Logic-based Benders Decomposition for Large-scale Optimization

J. N. Hooker

Abstract Logic-based Benders decomposition (LBBD) is a substantial generalization of classical Benders decomposition that, in principle, allows the subproblem to be any optimization problem rather than specifically a linear or nonlinear programming problem. It is amenable to a wide variety large-scale problems that decouple or otherwise simplify when certain decision variables are fixed. This chapter presents the basic theory of LBBD and explains how classical Benders decomposition is a special case. It also describes branch and check, a variant of LBBD that solves the master problem only once. It illustrates in detail how Benders cuts and subproblem relaxations can be developed for some planning and scheduling problems. It then describes the role of LBBD in three large-scale case studies. The chapter concludes with an extensive survey of the LBBD literature, organized by problem domain, to allow the reader to explore how Benders cuts have been developed for a wide range of applications.

1 Introduction

The fundamental challenge of large-scale optimization is that its difficulty tends to increase superlinearly, even exponentially, with the size of the problem. The challenge can often be overcome by solving the problem with a heuristic method, but only if one is willing to sacrifice optimality, or at least a proof of optimality. If a provably optimal solution is desired, decomposition may be the only practical recourse.

The advantage of decomposition is that it breaks a problem into smaller subproblems that are easier to solve. Due to superlinear complexity growth, solving many small subproblems can require much less computational effort than solving one large problem. The disadvantage of decomposition is that to achieve optimality, the subproblems must somehow communicate with each other, and it may be necessary

J. N. Hooker
Carnegie Mellon University e-mail: jh38@andrew.cmu.edu

to solve them repeatedly to converge to a solution. Nonetheless, when a problem has suitable structure, an algorithm based on decomposition can transform an intractable problem into a tractable one.

One of the best known and most successful decomposition strategies is *Benders decomposition*, which dates from the early 1960s [12]. It was originally designed for problems that become linear programming (LP) problems, known as *subproblems*, when certain variables are fixed. The duals of the subproblems are solved to obtain *Benders cuts*, which are constraints written in terms of the variables that were fixed. A *master problem* is then solved, subject to Benders cuts generated so far, to find another set of values for these variables, whereupon the procedure repeats. The Benders cuts exclude undesirable solutions, and the algorithm converges to a provably optimal solution under weak conditions. The Benders approach is most attractive when the subproblem is not only linear but decouples into smaller subproblems that can be solved independently.

Although classical Benders decomposition has many successful applications, its basic strategy is substantially restricted by the fact that the subproblem must be an LP problem—or a continuous nonlinear programming (NLP) problem in a 1972 extension to "generalized" Benders decomposition [48]. There are a wide range of potential applications in which the subproblem simplifies without yielding an LP or NLP problem, often by decoupling into smaller problems. Classical Benders decomposition cannot exploit this kind of problem structure.

Logic-based Benders decomposition (LBBD), introduced in [60, 66], addresses this issue by recognizing that the classical Benders method is a actually a special case of a much more general method. LBBD extends the underlying Benders strategy to cases in which the subproblem is an arbitrary optimization problem. It obtains Benders cuts by solving an *inference dual* of the subproblem, which reduces to the LP dual when the subproblem is linear.

Due due its greater versatility, LBBD has a large and rapidly growing range of successful applications. In many cases it leads to computational speedups of several orders of magnitude over the previous state of the art. It introduces a complication, however, that is not present in classical Benders methods. Logic-based Benders cuts must be developed anew for each problem class, while classical Benders cuts are automatically given by the LP dual of the subproblem. This can be viewed as a drawback, but it can also be an advantage. It may allow one to exploit the special structure of a given problem class with specially crafted Benders cuts, resulting in a effective solution method.

Branch and check, also introduced in [60], is a variation of LBBD. Rather than generate Benders cuts after each master problem is solved, it solves the master problem only once, by a branching method. When a feasible solution is found in the course of branching, the resulting subproblem is solved to obtain a Benders cut that is enforced throughout the remainder of the branching search. This method was first compared computationally with standard LBBD in [116], which introduced the term "branch and check." A related approach was later proposed specifically for mixed integer/linear programming (MILP) in [29], where the cuts are called *combinatorial Benders cuts*.

An important advantage of LBBD is that it provides a natural means to combine different kinds of problem formulations and solvers. For example, the master problem might be amenable to an MILP formulation and solver, while the subproblem might be better suited for constraint programming (CP). The MILP/CP combination is probably the most popular, because problems frequently decompose into an assignment problem suitable for MILP and a scheduling problem on which CP methods tend to excel [8, 63].

There is an outmoded perception that Benders decomposition converges slowly and is therefore often unsuitable even for problems that have a natural decomposition. We first remark that this perception is based on experience with classical Benders methods, not LBBD. Even classical Benders methods have been substantially accelerated over the last two decades, using a number of devices. An excellent survey of these improvements can be found in [91], which covers both classical methods and LBBD. These authors also document an explosion of literature on Benders methods since 2000 or so, no doubt due to improvements in performance.

We begin below with an exposition of the theory behind LBBD and a precise statement of the LBBD algorithm, followed by an explanation of how classical Benders decomposition is a special case. We then discuss branch and check, a variant of LBBD in which the master problem is solved only once, and when it is likely to be preferable to standard LBBD. Following this is a detailed presentation of how LBBD applies to a job assignment and scheduling problem with various objective functions. This discussion illustrates how to formulate logic-based Benders cuts for a class of problems that have perhaps benefited most frequently from LBBD to date. It also shows how to create subproblem relaxations for this class of problems, since such relaxations are often essential to the success of LBBD. We then briefly describe three case studies in which LBBD performed successfully in the context of large-scale optimization. Finally, since logic-based Benders cuts are problem-specific, it can be helpful to examine previous LBBD applications in a similar problem domain, to learn how others have exploited problem structure. Fortunately, a wide variety of LBBD applications now appear in the literature, and the chapter concludes with an extensive survey of these.

2 Fundamentals of LBBD

We begin by defining the *inference dual*, which is a basic element of LBBD. Consider a general optimization problem $\min\{f(x) \mid C(x),\ x \in D\}$, in which $C(x)$ represents a constraint set containing variables in $x = (x_1, \ldots, x_n)$, and D is the domain of x (such as tuples of nonnegative reals or integers). The inference dual is the problem of finding the tightest lower bound v on the objective function that can be deduced from the constraints, or

$$\max \left\{ v \;\middle|\; C(x) \overset{P}{\vdash} \big(f(x) \geq v\big),\ v \in \mathbb{R},\ P \in \mathcal{P} \right\} \tag{1}$$

Here $C(x) \overset{P}{\vdash} (f(x) \geq v)$ 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 a solution of the dual is a proof of the tightest bound v. Thus the inference dual is always defined with respect to an inference method that determines the family of proofs in $\mathcal{P}$. For a feasibility problem with no objective function, the dual can be viewed as the problem finding a proof P of infeasibility.

In practical applications of LBBD, the dual is defined with respect to the inference method used to prove optimality (or infeasibility) when solving the subproblem. We therefore assume that the inference dual is a strong dual: its optimal value is equal to the optimal value of the original problem.[1] When the subproblem is an LP problem, the inference method is nonnegative linear combination of inequalities, and the inference dual becomes the LP dual, as we will see in the next section.

We now define LBBD, which is applied to a problem of the form

$$\min \left\{ f(x, y) \;\middle|\; C(x, y),\ C'(x),\ x \in D_x,\ y \in D_y \right\} \tag{2}$$

Fixing x to $\bar{x}$ defines the *subproblem*

$$\min \left\{ f(\bar{x}, y) \;\middle|\; C(\bar{x}, y),\ y \in D_y \right\} \tag{3}$$

The inference dual of the subproblem is

$$\max \left\{ v \;\middle|\; C(\bar{x}, y) \overset{P}{\vdash} \big(f(\bar{x}, y) \geq v\big),\ v \in \mathbb{R},\ P \in \mathcal{P} \right\} \tag{4}$$

Let v^* be the optimal value of the subproblem (∞ if the subproblem is infeasible), and let proof P^* solve the inference dual by deducing the bound $f(\bar{x}, y) \geq v^*$. The essence of LBBD is that *this same proof* may deduce useful bounds when x is fixed to values other than $\bar{x}$. The term "logic-based" refers to this pivotal role of logical deduction. A *Benders cut* $z \geq B_{\bar{x}}(x)$ is derived by identifying a bound $B_{\bar{x}}(x)$ that proof P^* deduces for a given x. Thus, in particular, $B_{\bar{x}}(\bar{x}) = v^*$. The cut is added to the *master problem*, which in iteration k of the Benders procedure is

$$z_k = \min \left\{ z \;\middle|\; C'(x);\ z \geq B_{x^i}(x),\ i = 1, \ldots, k;\ x \in D_x \right\} \tag{5}$$

where $x^1, \ldots, x^k$ are the solutions of the master problems in iterations $1, \ldots, k$, respectively.

In any iteration k, the optimal value z_k of the master problem is a lower bound on the optimal value of (2), and the optimal value $v^* = v_k$ of the subproblem is an upper bound. The master problem values z_k increase monotonically as the iterations progress, while the subproblem values v_k can move up or down. The Benders algorithm terminates when the optimal value of the master problem equals the optimal value of the subproblem in some previous iteration. More precisely,

[1] An infeasible problem is viewed as having optimal value ∞ (when minimizing) or $-\infty$ (when maximizing).

it terminates when $z_k = \min\{v_i \mid i = 1, \ldots, k\}$, or when $z_k = \infty$ (indicating an infeasible problem).

At any point in the procedure, the Benders cuts in the master problem partially describe the projection of the feasible set of (2) onto x. Even when the procedure is terminated early, it yields a lower bound z_k and upper bound $\min_i\{v_i\}$ on the optimal value, as well as the best feasible solution found so far.

A formal statement of the LBBD procedure appears as Algorithm 1. Since z is unconstrained in the initial master problem $\min\{z \mid C'(x),\ x \in D_x\}$, we have $z_0 = -\infty$, and any feasible x can be selected as the solution x^0. Alternatively, we can use a "warm start" by generating a few Benders cuts in advance for heuristically chosen values of $\bar{x}$.

$k \leftarrow 0,\ v_0 \leftarrow -\infty,\ v_{\min} \leftarrow \infty$;
repeat
  if *the master problem (5) is infeasible* **then**
    stop; the original problem (2) is infeasible;
  end
  let x^k be an optimal solution of the master problem (5) with optimal value z_k;
  solve the subproblem (3) with $\bar{x} = x^k$;
  if *the subproblem (3) is unbounded* **then**
    stop; the original problem (2) is unbounded;
  end
  $k \leftarrow k + 1$;
  let v_k be the optimal value of the subproblem (3), where $v_k = \infty$ if (3) is infeasible;
  generate a Benders cut $z \geq B_{x^k}(x)$ such that $B_{x^k}(x^k) = v_k$;
  if $v_k < \infty$ **then**
    let y^k be an optimal solution of the subproblem (3);
    if $v_k < v_{\min}$ **then**
      $v_{\min} \leftarrow v_k,\ \ y^{\text{best}} \leftarrow y^k$;
    end
  end
until $z_k = v_{\min}$;
an optimal solution of the original problem (2) is $(x, y) = (x^k, y^{\text{best}})$;

Algorithm 1: LBBD procedure when the subproblem is an optimization problem.

If the subproblem is a feasibility problem with no objective function, the procedure continues until a feasible solution of the subproblem is found. In this case, the original problem (2) has the form

$$\min\left\{f(x) \,\middle|\, C(x, y),\ C'(x),\ x \in D_x,\ y \in D_y\right\} \tag{6}$$

and the subproblem is a constraint set

$$\left\{C(\bar{x}, y),\ y \in D_y\right\} \tag{7}$$

An infeasible subproblem gives rise to a *feasibility cut*, which is a constraint $N_{x^k}(x)$ that is violated when $x = \bar{x}$. The feasibility cut is added to the master problem

$$z_k = \min\left\{ f(x) \;\middle|\; C'(x);\ N_{x^i}(x),\ i = 1, \dots, k;\ x \in D_x \right\} \tag{8}$$

This version of the Benders algorithm yields no feasible solution until it terminates, but it still provides a valid lower bound z_k on the optimal value in any iteration k. A statement of the procedure appears as Algorithm 2.

$k \leftarrow 0$, feasible $\leftarrow$ false;
repeat
  if *master problem (5) is infeasible* **then**
    stop; original problem (2) is infeasible;
  end
  let x^k be an optimal solution of (5) with value z_k;
  solve subproblem (7) with $\bar{x} = x^k$;
  $k \leftarrow k + 1$;
  if *(7) is infeasible* **then**
    generate a feasibility cut $N_{x^k}(x)$ that is violated when $x = x^k$;
  else
    let feasible $\leftarrow$ true, and let y^k be a feasible solution of (7);
  end
until feasible = true;
an optimal solution of the original problem (2) is $(x, y) = (x^k, y^k)$;

Algorithm 2: LBBD procedure when the subproblem is a feasibility problem.

The simplest sufficient condition for finite convergence of LBBD is that the master problem variables have finite domains. This is normally the case in practice, since continuous variables (if any) typically occur in the subproblem. The following is shown in [66].

Theorem 1 *If the domains of the master problem variables are finite, Algorithm 1 and Algorithm 2 terminate after a finite number of steps.*

3 Classical Benders Decomposition

LBBD reduces to the classical Benders method when the subproblem is an LP problem and the inference dual is based on nonnegative linear combination and domination. To see this, we first show that the inference dual is the classical LP dual. Consider an LP problem

$$\min\{cx \mid Ax \geq b,\ x \geq 0\} \tag{9}$$

We suppose that an inequality $cx \geq v$ is deduced from $Ax \geq b$ when some nonnegative linear combination (surrogate) $uAx \geq ub$ of the constraint set dominates $cx \geq v$, where domination means that $uA \leq c$ and $ub \geq v$. The inference dual maximizes v subject to the condition that $cx \geq v$ can be deduced from $Ax \geq b$ and can therefore be written

$$\max \left\{ v \;\middle|\; uA \leq c,\ ub \geq v,\ u \geq 0 \right\}$$

This is equivalent to the classical LP dual $\max\{ub \mid uA \leq c,\ u \geq 0\}$. If (9) has a finite optimal value v^* and $\bar{u}$ is an optimal dual solution, we have $v^* = \bar{u}b$ by classical duality theory. The tuple $\bar{u}$ of dual multipliers therefore encodes a proof of optimality by deducing the bound $cx \geq \bar{u}b$.

Classical Benders decomposition is applied to a problem of the form

$$\min\{f(x) + cy \mid g(x) + Ay \geq b,\ x \in D_x,\ y \geq 0\} \tag{10}$$

The subproblem is the LP problem

$$\min \left\{ f(\bar{x}) + cy \;\middle|\; Ay \geq b - g(\bar{x}),\ y \geq 0 \right\}$$

If the subproblem has a finite optimal value v^* and $\bar{u}$ is an optimal dual solution, classical duality theory implies that $v^* = h(\bar{x}) + \bar{u}(b - g(\bar{x}))$, and the tuple $\bar{u}$ of dual multipliers therefore encodes a proof of optimality. The essence of classical Benders decomposition is that this same tuple of multipliers (i.e., this same proof) yields a lower bound $h(x) + \bar{u}(b - g(x))$ on the optimal value of (10) for any x. We therefore have a Benders cut $z \geq h(x) + \bar{u}(b - g(x))$. If the subproblem is infeasible and its dual is feasible (and therefore unbounded), the Benders cut is $\bar{u}(b - g(x)) \leq 0$, where $\bar{u}$ is an extreme ray solution of the subproblem dual.

4 Branch and Check

Branch and check is a variation of LBBD that solves the master problem only once. It is most naturally applied when a branching procedure solves the master problem, and the subproblem is a feasibility problem. When a feasible solution of the master problem is encountered during the branching process, it is "checked" by solving the subproblem that results. If the subproblem is infeasible, a feasibility cut is added to the master problem and enforced during the remainder of the tree search. The algorithm terminates when the search is exhaustive.

Branch and check can be an attractive alternative when the master problem is significantly harder to solve than the subproblem. Under the right conditions, it can bring orders-of-magnitude speedups relative to standard LBBD. A computational comparison of the two methods is provided in [11].

Branch and check is applied to a problem in the form (6). In all applications to date, the initial master problem $\min\{f(x) \mid C'(x),\ x \in D_x\}$ is a mixed integer/linear

programming (MILP) problem that is solved by a branch-and-cut method. When a feasible solution $\bar{x}$ is discovered at a node of the branching tree, perhaps because $\bar{x}$ is an integral solution of the current LP relaxation, the corresponding subproblem (7) is solved. If (7) is infeasible, a cut $N_{\bar{x}}(x)$ is derived. Since the master problem is an MILP problem, the cut must take the form of a linear inequality.

If the subproblem is feasible, the tree search continues in the normal fashion. If a feasibility cut is generated, the current solution $\bar{x}$ is no longer feasible because it violates the cut. The current LP relaxation is re-solved after adding the feasibility cut, and the search again continues in the usual fashion. The stopping condition is the same as for normal branch and cut. At termination, the incumbent solution (if any) is optimal for (6) because it defines a feasible subproblem.

Branch and check is not a special case of branch and cut, because its feasibility cuts are obtained in a different fashion. Unlike the cuts used in branch-and-cut methods, they are not are valid for the MILP problem being solved. They are based on a subproblem constraint set that does not appear in the MILP problem. They intermingle with standard cuts during the tree search but have different origins.

MILP solvers typically use a primal heuristic to generate feasible solutions at the root node of the search tree, and perhaps at other nodes. These feasible solutions can be used to obtain additional feasibility cuts, sometimes to great advantage. Another practical consideration is that branch and check requires modification of the code that solves the MILP master problem. This contrasts with standard LBBD, which can use an off-the-shelf method. Branch and check can therefore take longer to implement.

In an extended form of branch and check, there is no separate subproblem, but partial solutions found during the branching process are checked for feasibility. At any given node of the branching tree, the variables have been fixed so far can be treated as the master problem variables. Their values are checked for feasibility by solving the subproblem that remains after they are fixed. If infeasibility is verified, the dual solution of the subproblem can form the basis of a Benders cut. Such a method can be regarded as a branch-and-check algorithm with a dynamic partition of the master problem and subproblem variables.

Interestingly, this is the most popular scheme used in state-of-the-art satisfiability (SAT) solvers, where it is known as conflict-directed clause learning [10]. Partial solutions are not necessarily obtained by straightforward branching, but their feasibility is nonetheless checked by solving a subproblem in the form of an implication graph. If infeasibility is detected, a dual solution is derived by identifying a unit resolution proof of infeasibility represented by a conflict graph within the implication graph. A conflict clause (Benders cut) is obtained from a certain kind of partition of the conflict graph. Modern SAT solvers can handle industrial instances with well over a million variables. Their extraordinary efficiency is due in large part to clause learning, which is basically a form of branch and check.

5 Example: Job Assignment and Scheduling

An initial example will illustrate several practical lessons for applying LBBD:

- The master problem and subproblem are often best solved by different methods that are suited to the structure of the two problems.
- Often the subproblem solver does not provide easy access to its proof of optimality (or infeasibility), and Benders cuts must be based on dual information that is obtained indirectly.
- It is usually important to include a relaxation of the subproblem in the master problem, expressed in terms of master problem variables.

The example problem is as follows [64]. Jobs $1, \ldots, n$ must be assigned to facilities $1, \ldots, m$, and the jobs assigned to each facility must be scheduled. Each job j has processing time p_{ij} on facility i, release time r_j, and due date d_j. The facilities allow *cumulative scheduling*, meaning that jobs can run in parallel so long as the total rate of resource consumption does not exceed capacity. Job j consumes resources at the rate c_j, and facility i has a resource capacity of C_i. If $c_j = C_i = 1$ for all j and i, we have a *disjunctive scheduling* problem in which jobs run one at a time without overlap. Various objectives are possible, such as minimizing makespan, processing cost, the number of late jobs, or total tardiness.

The problem decomposes naturally. If the master problem assigns jobs to processors, and the subproblem schedules jobs, the subproblem decouples into a separate scheduling problem for each facility. Given that the scheduling component of the problem is the most difficult to scale up, this is a substantial benefit because it breaks up the scheduling problem into smaller pieces. Such a decomposition also allows appropriate solution methods to be applied to the master problem and subproblem. MILP tends to be very effective for assignment-type problems, while constraint programming (CP) is often the method of choice for scheduling problems.

Since the master problem is to be solved as a MILP problem, we formulate it with 0–1 variables and linear inequality/equality constraints. Let x_{ij} take the value 1 when job j is assigned to facility i. If we choose to minimize makespan M (the time at which the last job finishes), the master problem (2) is

$$\min \left\{ M \;\middle|\; M \geq M_i, \text{ all } i;\ \sum_i x_{ij} = 1, \text{ all } j;\ \text{Benders cuts};\ x_{ij} \in \{0,1\}, \text{ all } i,j \right\} \tag{11}$$

The variable M_i is the makespan on facility i and will appear in the cuts. Let $\bar{x}_{ij}$ be the solution of the master problem, and $J_i = \{i \mid \bar{x}_{ij} = 1\}$ the set of jobs assigned to processor i. If variable s_j is the start time of job j, the subproblem for each facility i can be given the CP formulation

$$\min \left\{ M_i \;\middle|\; \begin{array}{l} M_i \geq s_j + p_{ij},\ r_j \leq s_j \leq d_j - p_{ij}, \text{ all } j \in J_i; \\ \qquad \text{cumulative}\big(s(J_i), p_i(J_i), c(J_i), C_i\big) \end{array} \right\} \tag{12}$$

where $s(J_i)$ is the tuple of variables s_j for $j \in J_i$, and similarly for $p_i(J_i)$ and $c(J_i)$. The *cumulative constraint*, a standard global constraint in CP, requires that the jobs running at any one time have resource consumption rates that sum to at most C_i.

Benders cuts can be obtained as follows. Let M_i^* be the optimal makespan obtained for facility i. We wish to obtain a Benders cut $M_i \geq B_{i\bar{x}}(x)$ for each facility i that bounds the makespan for any assignment x, where $B_{i\bar{x}}(\bar{x}) = M_i^*$. Ideally, we would examine the proof of the optimal value M_i^* obtained by the CP solver and determine what kind of bound this same proof deduces when a different set of jobs is assigned to facility i. However, the solver typically does not provide access to a proof of optimality.

We must therefore rely on information about the proof obtained indirectly. The most basic information is which job assignments appear as premises in the proof. If we could find a smaller set $J_i' \subset J_i$ that contains the jobs whose assignments serve as premises, we could write a cut

$$M_i \geq M_i^* \Big(1 - \sum_{j \in J_i'} (1 - x_j)\Big) \tag{13}$$

that imposes the bound M_i^* whenever the jobs in J_i' are all assigned to facility i. One way to obtain J_i' is to tease out the structure of the proof by removing jobs from J_i one at a time and re-solving the scheduling problem until the minimum makespan drops below M_i^*. The last set of jobs for which the makespan is M_i^* becomes J_i'. This simple procedure can be quite effective, because in many applications the individual scheduling problems can be re-solved very rapidly. We will refer to cuts like (13) as *strengthened nogood cuts*, a term that originates with analogous feasiblity cuts. A cut of the form (13) should be generated and added to the master problem for each facility i.

A weakness of strengthened nogood cuts is that they provide no useful bound when not all jobs in J_i' are assigned to facility i. This weakness can often be overcome by using *analytical Benders cuts* that are based on an analysis of the subproblem structure. For example, if all the release times are the same, we can prove a lemma that gives rise to more useful Benders cuts. We give the proof from [64] to illustrate the type of reasoning that is often employed in the derivation of Benders cuts.

Lemma 1 *Suppose all release times $r_j = 0$, and M_i^* is the optimal makespan on facility i when the jobs in J_i' are assigned to it. If the jobs in $S \subseteq J_i'$ are removed from facility i, the optimal makespan M_i of the resulting problem satisfies*

$$M_i \geq M_i^* - \max_{j \in S}\{d_j\} + \min_{j \in S}\{d_j\} - \sum_{j \in S} p_{ij} \tag{14}$$

Proof Starting with the optimal schedule that yields makespan M_i, we can create a schedule for J_i' with makespan $M_i + \sum_{j \in S} p_{ij}$ by scheduling the jobs in S consecutively and contiguously, beginning at time M_i. We consider two cases:

$$\text{(a) } M_i + \sum_{j \in S} p_{ij} \leq \min_{j \in S}\{d_j\}, \quad \text{(b) } M_i + \sum_{j \in S} p_{ij} > \min_{j \in S}\{d_j\}$$

In case (a), the schedule is feasible, and we have $M_i^* \leq M_i + \sum_{j \in S} p_{ij}$ because M_i^* is optimal. But this implies (14). In case (b), we add $\max_{j \in S}\{d_j\}$ to both sides of (b) and rearrange terms to obtain

$$M_i + \sum_{j \in S} p_{ij} + \max_{j \in S}\{d_j\} - \min_{j \in S}\{d_j\} > \max_{j \in S}\{d_j\} \geq M^*$$

where the second inequality is due to the fact that M_i^* results from a feasible solution. This again implies (14). □

To obtain an analytic Benders cut from Lemma 1, we interpret S as the set of jobs in J_i' that are no longer assigned to facility i in subsequent Benders iterations; that is, the jobs j for which $x_{ij} = 0$. Thus (14) implies the cut

$$M_i \geq M_i^* - \sum_{j \in J_i'} p_{ij}(1 - x_{ij}) + \max_{j \in J_i'}\{d_j\} - \min_{j \in J_i'}\{d_j\} \tag{15}$$

because $\max_{j \in J_i}\{d_j\} \geq \max_{j \in S}\{d_j\}$ and $\min_{j \in J_i}\{d_j\} \leq \max_{j \in S}\{d_j\}$. A cut of this form is generated for each facility i and added to the master problem. These cuts should be used alongside the strengthened nogood cuts (13), which impose a tighter bound M_i^* when no jobs are removed from facility i and the deadlines differ.

A similar line of argument establishes analogous cuts when the jobs have different release times but no deadlines, an assumption perhaps better suited to minimum makespan problems:

$$M_i \geq M_i^* - \sum_{j \in J_i'} p_{ij}(1 - x_{ij}) - \max_{j \in J_i'}\{r_j\} + \min_{j \in J_i'}\{r_j\} \tag{16}$$

These cuts should also be used alongside the strengthened nogood cuts (13).

The Benders cuts are similar for other objective functions. If the objective is to minimize assignment cost, we let c_{ij} be the cost of assigning job j to facility i. The master problem becomes

$$\min \left\{ \sum_j c_{ij} x_{ij} \;\middle|\; \sum_i x_{ij} = 1, \text{ all } j; \text{ Benders cuts; } x_{ij} \in \{0,1\}, \text{ all } i, j \right\}$$

and the subproblem for facility i is the feasibility problem

$$\left\{ r_j \leq s_j \leq d_j - p_{ij}, \text{ all } j \in J_i; \text{ cumulative}\big(s(J_i), p_i(J_i), c(J_i), C_i\big) \right\}$$

Strengthened nogood cuts take the form $\sum_{j \in J_i'}(1 - x_j) \geq 1$ and are derived in a similar fashion as the makespan cuts.

If the objective is to minimize total tardiness, the deadlines become due dates, and the master problem is

$$\min\left\{\sum_i T_i \;\middle|\; \sum_i x_{ij} = 1,\ \text{all } j;\ \text{Benders cuts};\ x_{ij} \in \{0,1\},\ \text{all } i,j\right\}$$

where the variable T_i is the tardiness on facility i and will appear in the Benders cuts. The subproblem for facility i is

$$\min\left\{\sum_{j\in J_i}(s_j + p_{ij} - d_j)^+ \;\middle|\; r_j \le s_j,\ \text{all } j \in J_i;\ \text{cumulative}\big(s(J_i), p_i(J_i), c(J_i), C_i\big)\right\}$$

where $(\alpha)^+ = \max\{0,\alpha\}$. There are various schemes for deriving strengthened nogood cuts. One that has been used successfully [64] goes as follows. Let $T_i^*(J)$ be the minimum tardiness on facility i when the jobs in J are assigned to it, so that $T_i^*(J_i)$ is the minimum tardiness under the current assignment J_i. Let Z_i be the set of jobs in J_i that can be removed one at a time, with all other jobs remaining, without reducing the minimum tardiness. Thus $Z_i = \{j \in J_i \mid T_i^*(J_i \setminus \{j\}) = T_i^*(J_i)\}$. Then we have the cut

$$T_i \ge T_i^*(J_i \setminus Z_i)\Big(1 - \sum_{j\in J_i\setminus Z_i}(1 - x_{ij})\Big),\quad T_i \ge 0$$

This cut should be used alongside the cut

$$T_i \ge T_i^*(J_i)\Big(1 - \sum_{j\in J_i}(1 - x_{ij})\Big),\quad T_i \ge 0$$

to obtain a tighter bound $T^*(J_i)$ when no jobs are removed from facility i.

Analytical Benders cuts for the minimum tardiness problem are analogous to those for the minimum makespan problem, although the derivation is somewhat more involved. They have the form

$$T_i \ge M_i^* - \Big(\sum_{j\in J_i} p_{ij}(1 - x_{ij}) + \max_{j\in J_i}\{d_j\} - \min_{j\in J_i}\{d_j\}\Big)$$

where M_i^* is the minimum makespan on facility i for assignment J_i, a quantity that must be computed separately from the minimum tardiness. For reasons explained in Section 6.15.5 of [65], the analytical cuts are weak for cumulative scheduling but are more effective for the special case of disjunctive scheduling.

6 Relaxing the Subproblem

Past experience with LBBD has shown that success often depends on the presence of a subproblem relaxation in the master problem. It is not the typical sort of relaxation, because it is expressed in terms of the master problem variables rather than the subproblem variables. Nonetheless, a suitable relaxation is often evident based on

the structure of the subproblem. This is illustrated here for the job assignment and scheduling problem of the previous section, using various objective functions [64].

When the objective is to minimize assignment cost, a simple *time window relaxation* can be very effective. Let the *energy* consumed by job j be $p_j c_j$. Then it is clear that the total energy consumed by jobs that run in a given time interval $[t_1, t_2]$ can be no greater than the energy $C_i(t_2 - t_1)$ that is available during that interval. This gives rise to a simple valid inequality for facility i:

$$\sum_{j \in J(t_1,t_2)} p_{ij} c_{ij} x_{ij} \leq C_i(t_2 - t_1)$$

where $J(t_1, t_2)$ is the set of jobs with time windows in the interval $[t_1, t_2]$. We will refer to this inequality as $R_i[t_1, t_2]$. We can add a relaxation of the subproblem to the master problem by augmenting the master problem with the inequalities $R_i[r_j, d_{j'}]$ for each i and each distinct pair $[r_j, d_{j'}]$ of release times and deadlines. Actually, we can omit many of these inequalities because they are dominated by others. Let the *tightness* of an inequality $R_i(t_1, t_2)$ be

$$\theta_i(t_1, t_2) = (1/C_i) \sum_{j \in J(t_1,t_2)} p_{ij} c_{ij} - t_2 + t_1$$

Then the following lemma can be used to eliminate redundant inequalities:

Lemma 2 *Inequality $R_i[t_1, t_2]$ dominates $R_i[u_1, u_2]$ whenever $[t_1, t_2] \subseteq [u_1, u_2]$ and $\theta_i(t_1, t_2) \geq \theta_i(u_1, u_2)$.*

In a minimum makespan problem, we can use similar reasoning to bound the makespan. Let $R_i(t)$ be the inequality

$$M_i \geq t + (1/C_i) \sum_{j \in J(t,\infty)} p_{ij} c_{ij} x_{ij}$$

We can add inequalities $R_i(r_j)$ to the master problem for each distinct release time r_j. Again, some of these may be redundant.

The minimum tardiness problem calls for less obvious relaxation schemes. Two have been derived, the simpler of which is a time-window relaxation based on the following.

Lemma 3 *If jobs $1, \ldots, n$ are scheduled on a single facility i, the total tardiness is bounded below by*

$$\left((1/C_i) \sum_{j \in J(0,d_k)} p_{ij} c_{ij} - d_k \right)^+$$

for each $k = 1, \ldots, n$.

This yields the following valid inequalities for each facility i:

$$T_i \geq (1/C_i) \sum_{j \in J(0,d_k)} p_{ij} c_{ij} x_{ij} - d_k, \; k = 1, \ldots, n$$

These inequalities can be added to the master problem, along with $T_i \geq 0$, for each i. To state the second relaxation, let π_i be a permutation that orders jobs by increasing energy on facility i, so that $p_{i\pi_i(1)}c_{i\pi_i(1)} \leq \cdots \leq p_{i\pi_i(n)}c_{i\pi_i(n)}$. We have

Lemma 4 *If jobs* $1, \ldots, n$ *are scheduled on a single facility* i *and are indexed so that* $d_1 \leq \cdots \leq d_n$*, the total tardiness is bounded below by* $\sum_{k=1}^{n} \hat{T}_k$*, where*

$$\hat{T}_k = \left((1/C_i)\sum_{j=1}^{k} p_{i\pi_i(j)}c_{i\pi_i(j)} - d_k\right)^+, \quad k = 1, \ldots, n$$

This leads to a relaxation consisting of the inequality $T_i \geq \sum_{k=1}^{n} \hat{T}_{ik}$ for each i, as well as the inequalities $\hat{T}_{ik} \geq 0$ and

$$\hat{T}_{ik} \geq (1/C_i)\sum_{j=1}^{k} p_{i\pi_i(j)}c_{i\pi_i(j)}x_{i\pi_i(j)} - d_k - U_{ik}(1 - x_{ik})$$

for each i and $k = 1, \ldots, n$. Here U_{ik} is a big-M term that can be defined

$$U_{ik} = \sum_{j=1}^{k} p_{i\pi_i(j)}c_{i\pi_i(j)} - d_k$$

The cuts are valid even when $U_{ik} < 0$.

7 Large-Scale Case Studies

In this section we briefly highlight three case studies that illustrate how LBBD can succeed in large-scale settings. One is a massive optimization problem associated with an incentive auction conducted by the U.S. Federal Communications Commission (FCC). The team that designed the solution procedure received the prestigious Franz Edelman Award from INFORMS (Institute for Operations Research and the Management Sciences) in 2018. A second case study illustrates how LBBD can scale up by using approximate solutions of the master problem and subproblem. A third shows how LBBD can be of value even when the problem does not naturally decompose. The reader is referred to the original papers for details regarding the models and solution methods.

Frequency Spectrum Allocation

The FCC incentive auction was designed to reallocate parts of the frequency spectrum to television broadcasters and wireless providers, due to growing demand from the latter. Wireless providers offered bids to TV stations for additional bandwidth. After the auction was conducted, an optimization problem was solved to reallocate the spectrum [59]. The smaller TV band that remained was reallocated to stations so as

to minimize interference, and successful wireless bidders were assigned frequencies in an enlarged wireless band. The problem was formulated for nearly 3000 U.S. and Canadian stations and initially contained some 2.7 million pairwise interference restrictions, as well as many additional constraints. Stations in congested areas were allocated portions of the wireless band when necessary to reduce interference.

The overall solution algorithm was an LBBD procedure in which the master problem allocated frequencies to wireless providers and certain stations in the wireless band, and the subproblem attempted to find a feasible packing of the TV band for the remaining stations. The problem was solved to optimality.

Suboptimal Solution of Master Problem and/or Subproblem

The performance of LBBD can often be accelerated by solving the master problem, or even the subproblem, only approximately. Suboptimal solution of the master problem is a well-known and often used strategy, because only feasible solutions of the master problem are required to obtain Benders cuts. Of course, the optimal values obtained from the master problem are no longer valid lower bounds. To obtain a provably optimal solution of the original problem, the master problem must be solved to optimality in the latter stages of the Benders procedure.

Supoptimal solution of the subproblem is a more difficult proposition, because it can result in non-valid Benders cuts. This possibility was investigated for classical Benders decomposition in [128], where it is assumed that a dual feasible solution is available for an LP subproblem that is not solved to optimality, as for example when using an interior-point method. More relevant here is an application to LBBD in [92, 93], where dramatic speedups were obtained for a vehicle routing problem by solving the subproblem and possibly the master problem with metaheuristics. This sacrifices optimality but yields significantly better solutions, in much less time, than terminating an exact LBBD algorithm prematurely. This study also found ways, based on specific problem structure, to improve the accuracy of previously-solved subproblems using information obtained from approximate solution of the current subproblem.

Another possible strategy, not employed in [92, 93], is to solve the inference dual of the subproblem directly by searching for a proof of optimality, and then terminating the search prematurely. The resulting bound is not optimal but can serve as the basis of a valid Benders cut. To guarantee convergence to an optimal solution, the subproblem dual must at some point be solve to optimality. One general approach to solving the inference dual directly is given in [43], where branching is interpreted as a solution method for the inference dual and is managed accordingly.

No Natural Decomposition

Finally, a problem need not decompose naturally to benefit from LBBD. This is demonstrated in [21, 22], which solves a simple single-machine scheduling problem with time windows, but with many jobs and long time horizons. To decompose the model, the time horizon is divided into segments. The master problem decides in which segment(s) a job is processed, and the subproblem decouples into a scheduling

problem for each segment. Because a job can overlap two or more segments, the decomposition might be viewed as unnatural, and in fact the master problem and analytic Benders cuts are quite complex and tedious to formulate.

However, modeling complexity does not necessarily imply computational complexity. It was found that LBBD is much faster than stand-alone CP and MILP on minimum cost and makespan instances. Nearly all instances were intractable for CP, and many for MILP, while only one was intractable for LBBD. The LBBD advantage was more modest on minimum tardiness instances. Interestingly, CP solved a few of the instances in practically zero time (presumably because the arrangement of time windows permitted effective propagation), but it timed out on the remaining instances.

8 Survey of Applications

As noted earlier, logic-based Benders cuts must be developed for each class of problems. This may require ingenuity but affords an opportunity to exploit problem structure and design a superior solution algorithm. Fortunately, there is a sizeable LBBD literature that describes how Benders cuts can be designed for particular problem classes. Examination of previous work in an application domain similar to one's own may suggest effective cuts as well as subproblem relaxations. To this end, we survey a variety of LBBD applications.

Task Assignment and Scheduling

The assignment and scheduling problem discussed above is further studied in [26, 27], where updated computational testing found that LBBD remains orders of magnitude faster than the latest MILP technology, with the advantage over CP even greater. Similar assignment and scheduling problems are solved in [24, 61, 62, 118]. LBBD models having basically the same structure have been applied to steel production [49, 54], concrete delivery [72], batch scheduling in chemical plants [55, 83, 84, 117], resource scheduling with sequence-dependent setups [119], and computer processor scheduling [13, 14, 15, 20, 38, 58, 78, 79, 80, 81, 103, 108].

Vehicle Routing

LBBD has been applied to a number of vehicle routing problems, most of which decompose into vehicle assignment and routing components. The latter include capacitated vehicle routing [92, 93, 97, 107], dispatching and routing of automated guided vehicles [30, 86], dial-a-ride problems [98], and a senior door-to-door transportation problem (on which pure CP "surprisingly" performs better than LBBD) [77]. In other solution approaches, the master problem selects markets to visit in the traveling purchaser problem [17], finds initial routes in a traffic diversion problem [125], and assigns jobs to cranes in yard crane dispatching and scheduling problems [88, 122]. Additional LBBD applications include search and

rescue [90], coordinating vessels for inter-terminal transport [75, 76], and maritime traffic management [1].

Shop, Factory, and Employee Scheduling

LBBD applications to shop and factory scheduling include aircraft repair shop scheduling [6], job shop scheduling with human resurce constraints [51], permutation flowshop scheduling with time lags [52, 53], one-machine scheduling problems [22, 104, 105], and flowshop planning and scheduling with deteriorating machines [7]. There is also an application to feature-based assembly planning [71]. Employee scheduling applications include shift selection and task sequencing [9], multi-activity shift scheduling [106], shift scheduling with fairness constraints [37], railway crew rostering with fairness constraints [87], and a multi-activity tour scheduling problem that integrates shift scheduling with days-off planning [96].

Other Scheduling and Logistics Problems

LBBD has been applied to a variety of additional scheduling and logistics problems. In the transportation logistics domain, they include food distribution [110], bicycle sharing [73, 74], lock scheduling [123], and supply chain scheduling [115]. Other applications are project scheduling [70], robust call center scheduling [28], task scheduling for satellites [129], course timetabling [19], wind turbine maintenance scheduling [42], queuing design and control [113, 114], service restoration planning for infrastructure networks [50], and sports scheduling [23, 94, 95, 120, 121].

Health-Related Applications

LBBD applications in the rapidly growing healthcare field include operating room scheduling [82, 100, 102, 101], outpatient scheduling [99], and home health care routing and scheduling [25, 56, 57]. The study reported in [57] is a case in which branch and check substantially outperforms standard LBBD due to rapid solution of the subproblems relative to the master problem.

Facility Location

Some location problems addressed by LBBD are plant location [40], inventory location [124], stochastic warehouse location [112], location-allocation problems [39], and facility location and fleet management [41].

Network Design

Network design applications include green wireless local area network design [45, 47], transport network planning for postal services [89], broadcast domination network design [109], and the edge partition problem in optimal networks [111]. Yet another employment of LBBD is to solve the minimum dominating set problem [46], which is a key element of a variety of network design problems.

Other Applications

LBBD has proved useful in a number of additional domains, both practical and algorithmic. Practical applications include capacity planning [44], logic circuit verification [67], template design [112], strip packing [31, 85], orthogonal stock cutting [36], robust scheduling [28], and robust optimization [2, 3]. Interestingly, LBBD can also play a role in the solution of abstract problem classes, such as optimal control [18], quadratic programming [5, 68, 69], chordal completion [16], linear complementarity [69], modular arithmetic [70], the operator count problem in automated planning algorithms [35], and propositional satisfiability (SAT) [4]. A hitting set method that has been successfully applied to the maximum satisfiability problem (MAXSAT) is a special case of LBBD [32, 33].

9 Concluding Remarks: Implementation

One impediment to the use of LBBD may be the lack of an implementation in off-the-shelf software. The Benders cuts must be designed by hand, and the communication between master problem and subproblem carried out by special-purpose code. Yet solution of a large-scale problem is typically far from straightforward by any method, even using a powerful MILP or SAT solver. An MILP model must often be carefully written or reformulated to make it tractable for a solver, and formulation of problems for a SAT solver is even more challenging. Of course, many problems are beyond the capability of a stand-alone solver, regardless of how they are formulated.

Actually, LBBD has recently been automated in the MiniZinc modeling system [34]. The system chooses the decomposition and Benders cuts rather than relying on the user to do so. This is a convenience but may result in a less effective realization of LBBD. The ability of LBBD to benefit from user insight is a substantial advantage, since humans are much better at pattern recognition, and therefore at discerning problem structure, than machines. There are also modeling systems that can facilitate the implementation of logic-based Benders, such as IBM's OPL Studio, the Mosel development environment, and the general-purpose solver SIMPL [127]. A survey of software tools for implementing LBBD and other hybrid methods, if somewhat dated, can be found in [126].

As large-scale applications proliferate in our age of big data, and decomposition methods are increasingly called upon, it is likely that tools for their implementation will become increasingly powerful and make the application of methods like LBBD more routine.

References

1. L. Agussurja, A. Kumar, and H. C. Lau. Resource-constrained scheduling for maritime traffic management. In *AAAI Conference on Artificial Intelligence*, pages 6086–6093, 2018.
2. L. Assunção, T. F. Noronha, A. C. Santos, and R. Andrade. A linear programming based heuristic framework for min-max regret combinatorial optimization problems with interval costs. *Computers and Operations Research*, 81:51–66, 2017.
3. L. Assunção, T. F. Noronha, A. C. Santos, and R. Andrade. On the finite optimal convergence of logic-based Benders decomposition in solving 0-1 min-max regret optimization problems with interval costs. In *International Symposium on Combinatorial Optimization (ISCO 2016)*, volume 9849 of *Lecture Notes in Computer Science*, pages 1–12, 2017.
4. F. Bacchus, S. Dalmao, T. Pitassi, and G. Katsirelos. Relaxation search: A simple way of managing optional clauses. In *AAAI Conference on Artificial Intelligence*, pages 835–841. 2014.
5. L. Bai, J. E. Mitchell, and J.-S. Pang. On convex quadratic programs with linear complementarity constraints. *Computational Optimization and Applications*, 54:517–554, 2012.
6. M. A. Bajestani and J. C. Beck. Scheduling a dynamic aircraft repair shop with limited repair resources. *Journal of Artificial Intelligence Research*, 47:35–70, 2013.
7. M. A. Bajestani and J. C. Beck. A two-stage coupled algorithm for an integrated planning and flowshop scheduling problem with deteriorating machines. *Journal of Scheduling*, 18:471–486, 2015.
8. P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer, Dordrecht, 2001.
9. A. Y. Barlatt, A. M. Cohn, and O. Gusikhin. A hybridization of mathematical programming and dominance-driven enumeration for solving shift-selection and task-sequencing problems. *Computers and Operations Research*, 37:1298–1307, 2010.
10. P. Beame, H. Kautz, and A. Sabharwal. Understanding the power of clause learning. In *International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 2003.
11. J. C. Beck. Checking up on branch-and-check. In D. Cohen, editor, *Principle and Practice of Constraint Programming (CP)*, volume 6308 of *Lecture Notes in Computer Science*, pages 84–98, 2010.
12. J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.
13. 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.
14. L. Benini, M. Lombardi, M. Mantovani, M. Milano, and M. Ruggiero. Multi-stage Benders decomposition for optimizing multicore architectures. In L. Perron and M. A. Trick, editors, *CPAIOR 2008 Proceedings*, volume 5015 of *Lecture Notes in Computer Science*, pages 36–50. Springer, 2008.
15. L. Benini, M. Lombardi, M. Milano, and M. Ruggiero. Optimal resource allocation and scheduling for the CELL BE platform. *Annals of Operations Research*, 184:51–77, 2011.
16. D. Bergman and A. U. Raghunathan. A Benders approach to the minimum chordal completion problem. In L. Michel, editor, *CPAIOR Proceedings*, volume 9075 of *Lecture Notes in Computer Science*, pages 47–64. Springer, 2015.
17. K. E. C. Booth, T. T. Tran, and J. C. Beck. Logic-based decomposition methods for the travelling purchaser problem. In C.-G. Quimper, editor, *CPAIOR 2016 Proceedings*, volume 9678 of *Lecture Notes in Computer Science*, pages 55–64. Springer, 2016.
18. A. H. Borzabadi and M. E. Sadjadi. Optimal control of hybrid systems by logic-based Benders decomposition. In A. Giua, C. Mahulea, M. Silva, and J. Zaytoon, editors, *Analysis and Design of Hybrid Systems*, volume 3, pages 104–107, 2009.

19. H. Cambazard, E. Hebrard, B. O'Sullivan, and A. Papadopoulos. Local search and constraint programming for the post enrolment-based course timetabling problem. *Annals of Operations Research*, 194:111–135, 2012.
20. 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.
21. E. Çoban and J. N. Hooker. Single-facility scheduling over long time horizons by logic-based Benders decomposition. In A. Lodi, M. Milano, and P. Toth, editors, *CPAIOR 2010 Proceedings*, volume 6140 of *Lecture Notes in Computer Science*, pages 87–91. Springer, 2010.
22. E. Çoban and J. N. Hooker. Single-facility scheduling by logic-based Benders decomposition. *Annals of Operations Research*, 210:245–272, 2013.
23. K. K. H. Cheung. A Benders approach for computing lower bounds for the mirrored traveling tournament problem. *Discrete Optimization*, 6:189–196, 2009.
24. Y. Chu and Q. Xia. A hybrid algorithm for a class of resource-constrained scheduling problems. In R. Barták and M. Milano, editors, *CPAIOR 2005 Proceedings*, volume 3524 of *Lecture Notes in Computer Science*, pages 110–124. Springer, 2005.
25. A. Ciré and J. N. Hooker. A heuristic logic-based Benders method for the home health care problem. Presented at Matheuristics 2012, Angra dos Reis, Brazil, 2012.
26. A. A. Ciré, E. Çoban, and J. N. Hooker. Mixed integer programming vs logic-based Benders decomposition for planning and scheduling. In C. Gomes and M. Sellmann, editors, *CPAIOR 2013 Proceedings*, pages 325–331, 2013.
27. A. A. Ciré, E. Çoban, and J. N. Hooker. Logic-based Benders decomposition for planning and scheduling: A computational analysis. *Knowledge Engineering Review*, 31:440–451, 2016.
28. E. Çoban, A. Heching, J. N. Hooker, and A. Scheller-Wolf. Robust scheduling with logic-based Benders decomosition. In M. Lübbecke, A. Koster, P. Letmangthe, R. Madlener, B. Peis, and G. Walther, editors, *Operations Research Proceedings 2014*, volume 4510, pages 99–105. Springer, 2014.
29. G. Codato and M. Fischetti. Combinatorial Benders cuts for mixed-integer linear programming. *Operations Research*, 54:756–766, 2006.
30. 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. Régin and M. Rueher, editors, *CPAIOR 2004 Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 370–378. Springer, 2004.
31. J.-F. Côté, M. Dell'Amico, and M. Iori. Combinatorial Benders cuts for the strip packing problem. *Operations Research*, 62:643–661, 2014.
32. J. Davies and F. Bacchus. Postponing optimization to speed up MAXSAT solving. In C. Schulte, editor, *Principles and Practice of Constraint Programming (CP 2013)*, volume 8124 of *Lecture Notes in Computer Science*, pages 247–262. Springer, 2013.
33. J. Davies and F. Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In J. Lee, editor, *Principles and Practice of Constraint Programming (CP 2011)*, volume 6876 of *Lecture Notes in Computer Science*. Springer, 2013.
34. T. O. Davies, G. Gange, and P. J. Stuckey. Automatic logic-based Benders decomposition with MiniZinc. In M. Lübbecke, A. Koster, P. Letmangthe, R. Madlener, B. Peis, and G. Walther, editors, *AAAI Conferenece on Artificiaal Intelligence*, pages 787–793, 2017.
35. T. O. Davies, A. R. Pearce, P. J. Stuckey, and N. Lipovetzky. Sequencing operator counts. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 61–69, 2015.
36. M. Delorme, M. Iori, and Martello S. Logic basic Benders' decomposition for orthogonal stock cutting problems. *Computers and Operations Research*, 78:290–298, 2017.
37. T. Doi and T. Nishui. Two-level decomposition algorithm for shift scheduling problems. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 3773–3778, 2014.

38. A. Emeretlis, G. Theodoridis, P. Alefragis, and N. Voros. Mapping DAGs on heterogeneous platforms using logic-based Benders decompostion. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 119–124. IEEE, 2015.
39. 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, New York, 2009. Springer.
40. M. M. Fazel-Zarandi and J. C. Beck. Using logic-based Benders decomposition to solve the capacity- and distance-constrained plant location problem. *INFORMS Journal on Computing*, 24:387–398, 2012.
41. M. M. Fazel-Zarandi, O. Berman, and J. C. Beck. Solving a stochastic facility location/fleet management problem with logic-based Benders decomposition. *IIE Transactions*, 45:896–911, 2013.
42. A. Froger, M. Gendreau, J. E. Mendoza, E. Pinson, and L.-M. Rousseau. A branch-and-check approach for a wind turbine maintenance scheduling problem. *Computers and Operations Research*, 88:117–136, 2017.
43. G. Benadé and J. N. Hooker. Optimization bounds from the branching dual. *INFORMS Journal on Computing*, to appear.
44. M. Gavanelli, M. Milano, B. O'Sullivan, and A. Holland. What-if analysis through simulation-optimization hybrids. In *European Conference on Modeling and Simulation*, 2012.
45. B. Gendron, R. G. Garroppo, G. Nencioni, M. G. Scutellà, and L. Tavanti. Benders decomposition for a location-design problem in green wireless local area networks. *Electronic Notes in Discrete Mathematics*, 41:367–374, 2013.
46. B. Gendron, A. Lucena, A. Salles da Cunha, and L. Simonetti. Benders decomposition, branch-and-cut, and hybrid algorithms for the minimum connected dominating set problem. *INFORMS Journal on Computing*, 26:645–657, 2014.
47. B. Gendron, M. G. Scutellà, R. G. Garroppo, G. Nencioni, and L. Tavanti. A branch-and-Benders-cut method for nonlinear power design in green wireless local area networks. *European Journal of Operational Research*, 255:151–162, 2016.
48. A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.
49. A. Goldwasser and A. Schutt. Optimal torpedo scheduling. *Journal of Artificial Intelligence Research*, 63:955–986, 2018.
50. J. Gong, E. E. Lee, J. E. Mitchell, and W. A. Wallace. Logic-based multiobjective optimization for restoration planning. In W. Chaovalitwongse, K. C. Furman, and P. M. Pardalos, editors, *Optimization and Logistics Challenges in the Enterprise*, pages 305–324. 2009.
51. O. Guyon, P. Lemaire, E. Pinson, and D. Rivreau. Solving an integrated job-shop problem with human resource constraints. *Annals of Operations Research*, 213:147–171, 2014.
52. I. Hamdi and T. Loukil. Logic-based Benders decomposition to solve the permutation flowshop scheduling problem with time lags. In *International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, pages 1–7. IEEE, 2013.
53. I. Hamdi and T. Loukil. Upper and lower bounds for the permutation flowshop scheduling problem with minimal time lags. *Optimization Letters*, 9:465–482, 2015.
54. 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.
55. 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.
56. A. Heching and J. N. Hooker. Scheduling home hospice care with logic-based Benders decomposition. In C.-G. Quimper, editor, *CPAIOR 2016 Proceedings*, pages 187–197, 2016.
57. A. Heching, J. N. Hooker, and R. Kimura. A logic-based Benders approach to home healthcare delivery. *Transportation Science*, to appear.
58. P.-E. Hladik, H. Cambazard, A.-M. Déplanche, and N. Jussien. Solving a real-time allocation problem with constraint programming. *Journal of Systems and Software*, 81:132–149, 2008.

59. K. Hoffmann. Using hybrid optimization algorithms for very-large graph problems and for small real-time problems. INFORMS Optimization Society Conference, plenary talk, 2018.
60. J. N. Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, New York, 2000.
61. J. N. Hooker. A hybrid method for planning and scheduling. *Constraints*, 10:385–401, 2005.
62. J. N. Hooker. An integrated method for planning and scheduling to minimize tardiness. *Constraints*, 11:139–157, 2006.
63. J. N. Hooker. *Integrated Methods for Optimization*. Springer, 2007.
64. J. N. Hooker. Planning and scheduling by logic-based Benders decomposition. *Operations Research*, 55:588–602, 2007.
65. J. N. Hooker. *Integrated Methods for Optimization, 2nd ed.* Springer, 2012.
66. J. N. Hooker and G. Ottosson. Logic-based Benders decomposition. *Mathematical Programming*, 96:33–60, 2003.
67. 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.
68. J. Hu, J. E. Mitchell, and J.-S. Pang. An LPCC approach to nonconvex quadratic programs. *Mathematical Programming*, 133:243–277, 2012.
69. J. Hu, J. E. Mitchell, J.-S. Pang, K. P. Bennett, and G. Kunapuli. On the global solution of linear programs with linear complementarity constraints. *SIAM Journal on Optimization*, 19:445–471, 2008.
70. B. Kafle, G. Gange, P. Schachte, H. Søndergaard, and P. J. Stuckey. A Benders decomposition approach to deciding modular linear integer arithmetic. In S. Gaspers and T. Walsh, editors, *International Conference on Theory and Applications of Satisfiability Testing*, pages 380–397, 2017.
71. C. Kardos, A. Kovács, and J. Váncza. Decomposition approach to optimal feature-based assembly planning. *CIRP Annals*, 66:417–420, 2017.
72. J. Kinable and M. Trick. A logic-based Benders approach to the concrete delivery problem. In H. Simonis, editor, *CPAIOR 2014 Proceedings*, volume 8451 of *Lecture Notes in Computer Science*, pages 176–192. Springer, 2014.
73. C. Kloimüllner, P. Papazek, B. Hu, and G. R. Raidl. A cluster-first route-second approach for balancing bicycle sharing systems. In *International Conference on Computer Aided Systems Theory (EUROCAST)*, volume 9520 of *Lecture Notes in Computer Science*, pages 439–446. Springer, 2015.
74. C. Kloimüllner and G. R. Raidl. Full-load route planning for balancing bike sharing systems by logic-based Benders decomposition. *Networks*, 69:439–446, 2015.
75. S. Li, R. R. Negenborn, and G. Lodewijks. A logic-based Benders decomposition approach to improve coordination of inland vessels for inter-terminal transport. In *International Conference on Computational Logistics*, volume 9855 of *Lecture Notes in Computer Science*, pages 96–115. Springer, 2016.
76. S. Li, R. R. Negenborn, and G. Lodewijks. Closed-loop coordination of inland vessels operations in large seaports using hybrid logic-based Benders decomposition. *Transportation Research Part E*, 97:1–21, 2017.
77. C. Liu, D. M. Aleman, and J. C. Beck. Modelling and solving the senior transportation problem. In W.-J. van Hoeve, editor, *CPAIOR 2018 Proceedings*, volume 10848 of *Lecture Notes in Computer Science*, pages 412–428. Springer, 2018.
78. W. Liu, Z. Gu, J. Xu, X. Wu, and Y. Ye. Satisfiability modulo graph theory for task mapping and scheduling on multiprocessor systems. *IEEE Transactions on Parallel and Distributed Systems*, 22:1382–1389, 2011.
79. W. Liu, M. Yuan, X. He, Z. Gu, and X. Liu. Efficient SAT-based mapping and scheduling of homogeneous synchronous dataflow graphs for throughput optimization. In *Real-Time Systems Symposium*, pages 492–504. IEEE, 2008.
80. M. Lombardi and M. Milano. Stochastic allocation and scheduling for conditional task graphs in MPSoCs. In F. Benhamou, editor, *Principles and Practice of Constraint Programming (CP 2006)*, volume 4204 of *Lecture Notes in Computer Science*, pages 299–313. Springer, 2006.

81. M. Lombardi, M. Milano, M. Ruggiero, and L. Benini. Stochastic allocation and scheduling for conditional task graphs in multi-processor systems-on-chip. *Journal of Scheduling*, 13:315–345, 2010.
82. C. Luong. *An Examination of Benders Decomposition Approaches in Large-scale Healthcare Optimization Problems*. PhD thesis, University of Toronto, 2015.
83. C. T. Maravelias. A decomposition framework for the scheduling of single- and multi-stage processes. *Computers and Chemical Engineering*, 30:407–420, 2006.
84. 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, *CPAIOR 2004 Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.
85. J. Maschler and G. Raidl. Logic-based Benders decomposition for the 3-staged strip packing problem. In *International Conference on Operations Research (German OR Society)*, 2015.
86. T. Nishi, Y. Hiranaka, and I. E. Grossmann. A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles. *Computers and Operations Research*, 38:876–888, 2011.
87. T. Nishi, T. Sugiyama, and M. Inuiguchi. Two-level decomposition algorithm for crew rostering problems with fair working condition. *European Journal of Operational Research*, 237:465–473, 2014.
88. J. Nossack, D. Briskorn, and E. Pesch. Container dispatching and conflict-free yard crane routing in an automated container terminal. *Transportation Science*, 52:1059–1076, 2018.
89. B. Peterson and M. Trick. A Benders' approach to a transportation network design problem. In W.-J. van Hoeve and J. N. Hooker, editors, *CPAIOR 2009 Proceedings*, volume 5547 of *Lecture Notes in Computer Science*, pages 326–327, New York, 2009. Springer.
90. M. Raap, M. Moll, M. Zsifkovits, and S. Pickl. Utilizing dual information for moving target search trajectory optimization. In B. Hardy, A. Qazi, and S. Ravizza, editors, *5th Student Conference on Operational Research (SCOR 2016)*, volume 50 of *OpenAccess Series in Informatics (OASIcs)*, pages 1:1–1:10, Dagstuhl, Germany, 2016.
91. R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259:801–817, 2017.
92. G. R. Raidl, T. Baumhauer, and B. Hu. Speeding up logic-based Benders decomposition by a metaheuristic for a bi-level capacitated vehicle routing problem. In *International Workshop on Hybrid Metaheuristics*, volume 8457 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2014.
93. G. R. Raidl, T. Baumhauer, and B. Hu. Boosting an exact logic-based Benders decomposition approach by variable neighborhood search. *Electronic Notes in Discrete Mathematics*, 47:149–156, 2015.
94. R. V. Rasmussen. Scheduling a triple round robin tournament for the best Danish soccer league. *European Journal of Operational Research*, 20:795–810, 2008.
95. R. V. Rasmussen and M. A. Trick. A Benders approach to the constrained minimum break problem. *European Journal of Operational Research*, 177:198–213, 2007.
96. M. I. Restrepo, B. Gendron, and L. M. Rousseau. Combining Benders decomposition and column generation for multi-activity tour scheduling. *Computers and Operations Research*, 93:151–165, 2018.
97. S. Riazi, C. Seatzu, O. Wigstrom, and B. Lennartson. Benders/gossip methods for heterogeneous multi-vehicle routing problems. In *IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–6, 2013.
98. M. Riedler and G. R. Raidl. Solving a selective dial-a-ride problem with logic-based Benders decomposition. *Computers and Operations Research*, 96:30–54, 2018.
99. A. Riise, C. Mannino, and L. Lamorgese. Recursive logic-based Benders' decomposition for multi-mode outpatient scheduling. *European Journal of Operational Research*, 257:439–455, 2017.
100. V. Roshanaei, D. M. Aleman, and D. Urbach. Logic-based Benders decomposition approaches with application to operating room scheduling. In *INFORMS National Meeting*, 2015.

101. V. Roshanaei, C. Luong, D. M. Aleman, and D. Urbach. Collaborative operating room planning and scheduling. *INFORMS Journal on Computing*, 29:558–580, 2017.
102. V. Roshanaei, C. Luong, D. M. Aleman, and D. Urbach. Propagating logic-based Benders decomposition approaches for distributed operating room scheduling. *European Journal of Operational Research*, 257:439–455, 2017.
103. M. Ruggiero, A. Guerri, D. Bertozzi, F. Poletti, and M. Milano. Communication-aware allocation and scheduling framework for stream-oriented multi-processor systems-on-chip. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 3–8. European Design and Automation Association, 2006.
104. R. Sadykov. A hybrid branch-and-cut algorithm for the one-machine scheduling problem. In J. C. Régin and M. Rueher, editors, *CPAIOR 2004 Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 409–415. Springer, 2004.
105. R. Sadykov. A branch-and-check algorithm for minimizing the weighted number of late jobs on a single machine with release dates. *European Journal of Operational Research*, 189:1283–1304, 2008.
106. D. Salvagnin and T. Walsh. A hybrid MIP/CP approach for multi-activity shift scheduling. In M. Milano, editor, *Principles and Practice of Constraint Programming*, volume 7514 of *Lecture Notes in Computer Science*, pages 633–646. Springer, 2012.
107. R. Sarmad, O. Wigström, and S. Carla. Benders/gossip methods for heterogeneous multi-vehicle routing problems. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–6. IEEE, 2013.
108. N. Satish, K. Ravindran, and K. Keutzer. A decomposition-based constraint optimization approach for statically scheduling task graphs with communication delays to multiprocessors. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 57–62. EDA Consortium, 2007.
109. S. Shen and J. C. Smith. A decomposition approach for solving a broadcast domination network design problem. *Annals of Operations Research*, 210:333–360, 2011.
110. S. Solak, C. Scherrer, and A. Ghoniem. The stop-and-drop problem in nonprofit food distribution networks. *Annals of Operations Research*, 221:407–426, 2014.
111. Z. C. Taşkın, J. C. Smith, S. Ahmed, and A. J. Schaefer. Cutting plane algorithms for solving a stochastic edge-partition problem. *Discrete Optimization*, 6:420–435, 2009.
112. S. Tarim, S. Armagan, 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, *International Workshop on Constraint Solving and Constraint Logic Programming (CSCLP)*, pages 133–148. Springer, 2006.
113. 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 2007)*, volume 1, pages 261–266. AAAI Press, 2007.
114. D. Terekhov, J. C. Beck, and K. N. Brown. A constraint programming approach for solving a queueing design and control problem. *INFORMS Journal on Computing*, 21:549–561, 2009.
115. D. Terekhov, M. K. Doğru, U. Özen, and J. C. Beck. Solving two-machine assembly scheduling problems with inventory constraints. *Computers and Industrial Engineering*, 63:120–134, 2012.
116. 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.
117. C. Timpe. Solving planning and scheduling problems with combined integer and constraint programming. *OR Spectrum*, 24:431–448, 2002.
118. T. Tran, A. Araujo, and J. C. Beck. Decomposition methods for the parallel machine scheduling problem with setups. *INFORMS Journal on Computing*, 28:83–95, 2016.
119. T. T. Tran and J. C. Beck. Logic-based Benders decomposition for alternative resource scheduling with sequence dependent setups. In *European Conference on Artificial Intelligence (ECAI)*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 774–779. IOS Press, 2012.

120. M. Trick and H. Yıldız. Benders cuts guided large neighborhood search for the traveling umpire problem. In P. Van Hentenryck and L. Wolsey, editors, *CPAIOR Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, pages 332–345. Springer, 2007.
121. M. Trick and H. Yıldız. Benders cuts guided large neighborhood search for the traveling umpire problem. *Naval Research Logistics*, pages 771–781, 2011.
122. S. van Dijk. *Decomposition methods and rolling horizon approach for the yard crane scheduling problem*. PhD thesis, Delft University of Technology, 2015.
123. J. Verstichel, J. Kinable, P. De Causmaecker, and G. Vanden Berghe. A combinatorial Benders decomposition for the lock scheduling problem. *Computers and Operations Research*, 54:117–128, 2015.
124. D. Wheatley, F. Gzara, and E. Jewkes. Logic-based Benders decomposition for an inventory-location problem with service constraints. *Omega*, 55:10–23, 2015.
125. Q. Xia, A. Eremin, and M. Wallace. Problem decomposition for traffic diversions. In J. C. Régin and M. Rueher, editors, *CPAIOR 2004 Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 348–363. Springer, 2004.
126. T. H. Yunes. Software tools supporting integration. In P. van Hentenryck and M. Milano, editors, *Hybrid Optimization: The Ten Years of CPAIOR*, pages 393–424. Springer, New York, 2011.
127. T. H. Yunes, I. Aron, and J. N. Hooker. An integrated solver for optimization problems. *Operations Research*, 58:342–356, 2010.
128. G. Zakeri, A. B. Philpott, and D. M. Ryan. Inexact cuts in Benders decomposition. *SIAM Journal of Optimization*, 10:643–657, 2000.
129. J. Zhu, L. Zhang, D. Qiu, and H. Li. Task scheduling for multi-electro-magnetic detection satellite with a combined algorithm. *Journal of Systems Engineering*, 23:88–98, 2012.