Logic-Based Benders Decomposition

J. N. Hooker

Keywords Benders decomposition, Combinatorial Benders cuts, Logic-based Benders decomposition, Inference duality, Branch and check, Conflict-directed clause generation

1 Introduction

Benders decomposition, introduced in 1962 by Jacques Benders [2], is a popular and highly successful optimization tool. While it has seen countless applications, it contains a problem-solving idea that, when recognized and exploited, opens the door to even broader application. Specifically, its derivation of Benders cuts from the dual of the linear programming (LP) subproblem can be interpreted as a special case of logical inference. This insight allows extension of the classical method to *logic-based Benders decomposition* (LBBD), in which the subproblem can in principle be any optimization problem, thus resulting in a substantial generalization [12, 15]. This extension has enabled hundreds of new applications, ranging from supply chain logistics to organ transplantation to search-and-rescue operations [14]. The method is "logic-based" in the sense that logical inference plays a key role in its conception, and not because there is any need for a problem to involve formal logic.

J.N. Hooker
Tepper School of Business, Carnegie Mellon University, Pittsburgh, USA

2 The Basic Idea

Benders decomposition partitions the problem variables into *master problem* variables and *subproblem* variables. The master problem variables are chosen so that when they are fixed to some set of values, the original problem reduces to a substantially simpler subproblem. The subproblem and its dual are then solved. The dual solution is interpreted as a proof that demonstrates optimality by providing a bound on the optimal value of the subproblem, and therefore on the optimal value of the original problem. The key insight of Benders decomposition is that *this same proof* yields a bound on the optimal value that results from other values of the master problem variables. This allows one to formulate an inequality constraint, known as a *Benders cut*, that imposes this bound. A new set of trial values for the master problem variables is then obtained by solving the master problem, which consists of Benders cuts generated so far. The process repeats until the optimal value of the master problem is equal to the best previous subproblem value, in which case the original problem has been solved, or until the master problem is infeasible, in which case the original problem is shown to be infeasible.

In the classical method, the subproblem is a linear programming problem. Dual multipliers obtained from solving the subproblem prove optimality (or infeasibility) and yield a Benders cut by specifying a linear combination of the subproblem constraints. LBBD generalizes this strategy by deriving a cut from the *inference dual* of the subproblem. The inference dual is the problem of finding the tightest bound on the optimal value that can be logically deduced from the constraints. The dual solution is a proof of this bound. The LP dual is a special case of inference duality in which the inference method is nonnegative linear combination, and the proof is encoded by dual multipliers.

One advantage of LBBD is that the subproblem can be solved by combinatorial optimization methods, such as constraint programming algorithms, branch-and-cut procedures, or such problem-specific methods as traveling salesman algorithms. On the other hand, it is often necessary to formulate problem-specific Benders cuts, since one cannot routinely obtain cuts from the LP dual as in the classical method. This exercise may require some ingenuity, but it also allows one to exploit the special structure of a given problem class.

3 An Example

LBBD can be illustrated by a simple machine assignment and scheduling problem [14]. Four jobs are to be scheduled on two machines. The jobs have release times $(r_1, \dots, r_4) = (3, 3, 0, 3)$ and deadlines $(d_1, \dots, d_4) = (6, 5, 5, 5)$. They have processing times $(p_{11}, \dots, p_{14}) = (3, 1, 3, 1)$ on machine 1 and $(p_{21}, \dots, p_{24}) = (2, 1, 2, 1)$ on machine 2. The problem is to assign jobs to

machines, and schedule the jobs on each machine, so as to minimize makespan (the latest finish time). There is a natural decomposition: the master problem assigns jobs to machines, whereupon the scheduling subproblem decouples into a separate problem for each machine.

Let binary variable $x_{ij} = 1$ when job j is assigned to machine i, and suppose we initially set $x_{11} = x_{12} = x_{23} = x_{24} = 1$. The two scheduling problems that result cannot be solved as LP problems, but an *edge finding* algorithm [7] proves that the assignment of jobs 1 and 2 to machine 1 has no feasible schedule (i.e., the optimal makespan is infinite). This proof is the solution of the inference dual for machine 1. As a Benders cut we get the rather uninteresting *nogood cut* $x_{11} + x_{12} \leq 1$, which forbids this particular assignment. However, the edge finding algorithm provides a lower bound of 4 on machine 2's makespan, and this proof relies only on the premise that job 4 is assigned to the machine. The same proof therefore establishes the bound when (x_{23}, x_{24}) is either $(0, 1)$ or $(1, 1)$. We therefore get the more interesting Benders cut $z \geq 4x_{24}$, where z denotes makespan. Further analysis of the subproblem yields an *analytical* Benders cut $z \geq 2x_{23} + 3x_{24} - 1$.

The next trial assignment is obtained by solving a master problem that minimizes z subject to these three cuts. The process continues until the optimal value of the master problem equals the best subproblem value obtained so far, at which point a makespan of 5 is obtained by assigning jobs 2 and 4 to machine 1 and the others to machine 2.

4 The LBBD Algorithm

Logic-based Benders decomposition is applied to a problem of the form

$$\min_{\boldsymbol{x},\boldsymbol{y}} \left\{ f(\boldsymbol{x},\boldsymbol{y}) \;\middle|\; \mathcal{C}(\boldsymbol{x},\boldsymbol{y}),\ \mathcal{C}'(\boldsymbol{x}),\ \boldsymbol{x} \in \mathcal{D}_{\boldsymbol{x}},\ \boldsymbol{y} \in \mathcal{D}_{\boldsymbol{y}} \right\} \tag{1}$$

where variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ have been selected as master problem variables and $\boldsymbol{y} = (y_1, \ldots, y_m)$ as subproblem variables. $\mathcal{D}_{\boldsymbol{x}}$ and $\mathcal{D}_{\boldsymbol{y}}$ are the variable domains (e.g., tuples of reals, integers, etc.). $\mathcal{C}(\boldsymbol{x},\boldsymbol{y})$ is a set of constraints that contain variables in $\boldsymbol{x}$ and $\boldsymbol{y}$, while the constraints in $\mathcal{C}'(\boldsymbol{x})$ contain variables in $\boldsymbol{x}$. Fixing $\boldsymbol{x}$ to a given value $\bar{\boldsymbol{x}}$ yields the subproblem

$$\min_{\boldsymbol{y}} \left\{ f(\bar{\boldsymbol{x}},\boldsymbol{y}) \;\middle|\; \mathcal{C}(\bar{\boldsymbol{x}},\boldsymbol{y}),\ \boldsymbol{y} \in \mathcal{D}_{\boldsymbol{y}} \right\} \tag{2}$$

which we denote $\mathrm{SP}(\bar{\boldsymbol{x}})$. The inference dual of $\mathrm{SP}(\bar{\boldsymbol{x}})$ is

$$\max_{v,P} \left\{ v \;\middle|\; \mathcal{C}(\bar{\boldsymbol{x}},\boldsymbol{y}) \overset{P}{\vdash} \big(f(\bar{\boldsymbol{x}},\boldsymbol{y}) \geq v\big),\ v \in \mathbb{R},\ P \in \mathcal{P} \right\} \tag{3}$$

The notation $A \overset{P}{\vdash} B$ means that B can be deduced from A using proof P. The dual seeks the tightest lower bound v on $f(\bar{\boldsymbol{x}}, \boldsymbol{y})$ that can be deduced from the constraints $\mathcal{C}(\bar{\boldsymbol{x}}, \boldsymbol{y})$.

The dual is therefore defined with respect to an inference method indicated by the family $\mathcal{P}$ of possible proofs. For purposes of LBBD, the inference method is assumed to be complete in the sense it can deduce any valid bound. This results in a strong dual, meaning that (2) and (3) necessarily have the same optimal value (possibly ∞ or $-\infty$ in the case of an infeasible or unbounded subproblem). Any exact method for solving the subproblem defines a strong inference dual, and the optimality proof supplied by the method serves as a dual solution. For example, a branch-and-bound search tree can be regarded as a dual solution. The edge finding procedure mentioned earlier is not complete but can be augmented by branching to obtain a strong inference dual, as occurs in constraint programming solvers.

An LBBD algorithm now proceeds as follows. Let $v^*(\bar{\boldsymbol{x}})$ be the optimal value of $\text{SP}(\bar{\boldsymbol{x}})$, and suppose that proof P^* solves the inference dual by deducing the lower bound $v = v^*(\bar{\boldsymbol{x}})$ on $f(\bar{\boldsymbol{x}}, \boldsymbol{y})$ for any feasible solution $\boldsymbol{y}$ of $\text{SP}(\bar{\boldsymbol{x}})$. The essence of LBBD is that *this same proof* P^* may deduce for any $\boldsymbol{x}$ a useful lower bound on $f(\boldsymbol{x}, \boldsymbol{y})$ that is valid for any feasible solution $\boldsymbol{y}$ of $\text{SP}(\boldsymbol{x})$, and therefore for any feasible solution $(\boldsymbol{x}, \boldsymbol{y})$ of the original problem (1). The bound is captured in a Benders cut of the form $z \geq B_{\bar{\boldsymbol{x}}}(\boldsymbol{x})$, where z represents the value of $f(\boldsymbol{x}, \boldsymbol{y})$ in the master problem.

The Benders cut is added to the master problem, which contains all previously generated Benders cuts as constraints. The master problem in iteration k is

$$z_k = \min_{z,\boldsymbol{x}} \left\{ z \;\middle|\; \mathcal{C}'(\boldsymbol{x});\ z \geq B_{\boldsymbol{x}^\ell}(\boldsymbol{x}),\ \ell = 1, \ldots, k-1;\ \boldsymbol{x} \in \mathcal{D}_{\boldsymbol{x}} \right\} \tag{4}$$

where $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{k-1}$ are the solutions of the master problem in previous iterations. It is common in practice to add multiple Benders cuts in each iteration.

The next step is to solve the master problem and use its optimal solution $\boldsymbol{x}^k$ to define the next subproblem $\text{SP}(\boldsymbol{x}^k)$. At this point, the process repeats, and it continues until the master problem has the same value as the best optimal subproblem value obtained so far; that is, until

$$z_k = \min \left\{ v^*(\boldsymbol{x}^\ell) \;\middle|\; \ell = 1, \ldots, k-1 \right\}$$

If the master problem becomes infeasible at any point, the procedure terminates with the conclusion that (1) is infeasible. Otherwise, the procedure yields an optimal solution $(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ of (1), where ℓ is an iteration in which the best subproblem value was obtained, and $\boldsymbol{y}^\ell$ is the optimal solution of $\text{SP}(\boldsymbol{x}^\ell)$.

If $z \geq B_{\bar{\boldsymbol{x}}}(\boldsymbol{x})$ is to qualify as a valid Benders cut, the bound $B_{\bar{\boldsymbol{x}}}(\boldsymbol{x})$ must satisfy two properties:

(B1) $B_{\bar{\boldsymbol{x}}}(\boldsymbol{x}) \leq f(\boldsymbol{x}, \boldsymbol{y})$ for any $(\boldsymbol{x}, \boldsymbol{y})$ feasible in (1).
(B2) $B_{\bar{\boldsymbol{x}}}(\bar{\boldsymbol{x}}) = v^*(\bar{\boldsymbol{x}})$.

Property (B1) says that the cut indeed provides a valid lower bound on $f(\boldsymbol{x}, \boldsymbol{y})$. Property (B2) ensures finite convergence when the domain $\mathcal{D}_{\boldsymbol{x}}$ is finite.

5 Classical Benders Decomposition

The classical Benders method is a special case of the procedure just described. It is applied to problems of the form

$$\min_{\boldsymbol{x},\boldsymbol{y}} \left\{ f(\boldsymbol{x}) + \boldsymbol{c}\boldsymbol{y} \;\middle|\; \boldsymbol{g}(\boldsymbol{x}) + A\boldsymbol{y} \geq \boldsymbol{b},\ \boldsymbol{y} \geq \boldsymbol{0},\ \boldsymbol{x} \in \mathcal{D}_{\boldsymbol{x}} \right\} \tag{5}$$

For a given $\bar{\boldsymbol{x}}$, the subproblem $\mathrm{SP}(\bar{\boldsymbol{x}})$ is the LP problem

$$\min_{\boldsymbol{y}} \left\{ f(\bar{\boldsymbol{x}}) + \boldsymbol{c}\boldsymbol{y} \;\middle|\; A\boldsymbol{y} \geq \boldsymbol{b} - \boldsymbol{g}(\bar{\boldsymbol{x}}),\ \boldsymbol{y} \geq \boldsymbol{0} \right\} \tag{6}$$

We will show that cuts derived from an inference dual of (6) are classical Benders cuts, where the dual is based on nonnegative linear combination. If (6) is feasible, we say that $\boldsymbol{c}\boldsymbol{y} \geq v$ can be inferred from $A\boldsymbol{y} \geq \boldsymbol{b} - \boldsymbol{g}(\bar{\boldsymbol{x}}),\ \boldsymbol{y} \geq \boldsymbol{0}$ when the linear combination $\boldsymbol{u}A\boldsymbol{y} \geq \boldsymbol{u}(\boldsymbol{b} - \boldsymbol{g}(\bar{\boldsymbol{x}}))$ dominates $\boldsymbol{c}\boldsymbol{y} \geq v$ for some $\boldsymbol{u} \geq \boldsymbol{0}$, where domination means that $\boldsymbol{u}A \leq \boldsymbol{c}$ and $\boldsymbol{u}(\boldsymbol{b} - \boldsymbol{g}(\bar{\boldsymbol{x}})) \geq v$. Then the inference dual of (6) becomes

$$\max_{v,\boldsymbol{u}} \left\{ v \;\middle|\; \boldsymbol{u}A \leq \boldsymbol{c},\ \boldsymbol{u}(\boldsymbol{b} - \boldsymbol{g}(\bar{\boldsymbol{x}})) \geq v - f(\bar{\boldsymbol{x}}),\ \boldsymbol{u} \geq \boldsymbol{0} \right\}$$

But this is equivalent to the classical LP dual of (6) if we note that $f(\bar{\boldsymbol{x}})$ is simply a constant added to the objective function:

$$\max_{\boldsymbol{u}} \left\{ f(\bar{\boldsymbol{x}}) + \boldsymbol{u}(\boldsymbol{b} - \boldsymbol{g}(\bar{\boldsymbol{x}})) \;\middle|\; \boldsymbol{u}A \leq \boldsymbol{c},\ \boldsymbol{u} \geq \boldsymbol{0} \right\} \tag{7}$$

Thus if we let $B_{\bar{\boldsymbol{x}}}(\boldsymbol{x}) = f(\boldsymbol{x}) + \bar{\boldsymbol{u}}(\boldsymbol{b} - \boldsymbol{g}(\boldsymbol{x}))$, where $\bar{\boldsymbol{u}}$ is the solution of (7), $B_{\bar{\boldsymbol{x}}}(\boldsymbol{x})$ satisfies condition (B2) because the LP dual is a strong dual. The key observation for classical Benders decomposition is that $\bar{\boldsymbol{u}}$ remains feasible in the dual for any $\boldsymbol{x}$, since $\boldsymbol{x}$ occurs only in the objective function of (7). Thus by weak LP duality, $\bar{\boldsymbol{u}}(\boldsymbol{b} - \boldsymbol{g}(\boldsymbol{x})) \leq \boldsymbol{c}\boldsymbol{y}$ for any $\boldsymbol{x}$ and any feasible $\boldsymbol{y}$, and we can write

$$f(\boldsymbol{x}) + \bar{\boldsymbol{u}}\big(\boldsymbol{b} - \boldsymbol{g}(\boldsymbol{x})\big) \leq f(\boldsymbol{x}) + \boldsymbol{c}\boldsymbol{y} \tag{8}$$

for any $(\boldsymbol{x}, \boldsymbol{y})$ feasible in (5). That is, the same proof that establishes optimality of the subproblem $\mathrm{SP}(\bar{\boldsymbol{x}})$, namely the linear combination based on multipliers $\bar{\boldsymbol{u}}$, proves a valid bound for other values of $\boldsymbol{x}$. Due to (8), $B_{\bar{\boldsymbol{x}}}(\boldsymbol{x})$ satisfies (B1) as well as (B2), which means $z \geq B_{\bar{\boldsymbol{x}}}(\boldsymbol{x})$ is a valid logic-based

Benders cut. But this cut is identical to the classical cut

$$z \geq f(\boldsymbol{x}) + \bar{\boldsymbol{u}}\big(\boldsymbol{b} - \boldsymbol{g}(\boldsymbol{x})\big)$$

The classical cut $\bar{\boldsymbol{u}}(\boldsymbol{b} - \boldsymbol{g}(\boldsymbol{x})) \leq 0$ for infeasible (6) is similarly derived.

6 Deriving Logic-Based Cuts

Three broad categories of logic-based Benders cuts appear in the literature: strengthened nogood cuts, analytical cuts, and explanation-based cuts. Cuts can also be classified as *optimality cuts* $z \geq B_{\bar{\boldsymbol{x}}}(\boldsymbol{x})$, which place a bound on the optimal value, and *feasibility cuts*, which arise from an infeasible subproblem. We focus on optimality cuts, since feasibility cuts can be viewed as optimality cuts for which $B_{\bar{\boldsymbol{x}}}(\bar{\boldsymbol{x}}) = \infty$, although more straightforward cuts are used in practice. For instance, $x_{11} + x_{12} \leq 1$ is a feasibility cut in the scheduling example presented earlier.

To simplify exposition, we assume the common situation in which $\boldsymbol{x}$ is binary and the cuts are linear inequalities. We also suppose that $v^*(\boldsymbol{x})$ is a monotone nondecreasing function of $\boldsymbol{x}$. This occurs, for instance, when the master problem assigns tasks to a processing facility, as in the scheduling example.

Strenthened nogood cuts are obtained by strengthening a simple nogood cut of the form

$$z \geq v^*(\bar{\boldsymbol{x}}) - \big(v^*(\bar{\boldsymbol{x}}) - \underline{v}\big) \sum_{j \in J} (1 - x_j) \tag{9}$$

where $\bar{\boldsymbol{x}}$ is the solution of the current master problem and $J = \{j \mid \bar{x}_j = 1\}$. The cut (9) states that if all the tasks in J are assigned to a facility (possibly among other tasks), then the resulting subproblem cost will be at least $v^*(\bar{\boldsymbol{x}})$. If not all tasks in J are assigned, the cut imposes a known lower bound $\underline{v}$ on cost. The cut is strengthened by assigning various subsets J' of the tasks in J to the facility, and checking whether the optimal value changes. If not, J' can replace J in (9) while preserving validity. It is often practical to solve the subproblem repeatedly in this manner due to the speed of solution. The cut is stronger when J' is small, and it is *irreducible* if no task can be removed from J' without destroying validity.

Techniques for reducing J include a greedy procedure that removes one task at a time, a deletion filter [8], depth-first binary search (DFBS), a faster heuristic version of DFBS [3], and the QuickXplain procedure [16]. The deletion filter, DFBS, and QuickXplain yield irreducible cuts.

Analytical cuts are obtained by combining the current subproblem solution with an analysis of the subproblem structure. They are often more effective than strengthened nogood cuts because they represent a deeper understanding of problem characteristics. An analytical cut $z \geq 2x_{23} + 3x_{24} - 1$ was given

in the above scheduling example. In general, this cut for machine i consists of two inequalities

$$z \geq v^*(\bar{\boldsymbol{x}}) - \sum_{j \in J} (1 - x_j)\big(p_{ij} + \max\{0,\ r_j - r_{\min} - p_{\min}\}\big) - (d_{\max} - d_{\min})$$

$$z \geq v^*(\bar{\boldsymbol{x}}) - \sum_{j \in J} (1 - x_j)\big(p_{ij} + \max\{0,\ r_j - r_{\min} - p_{\min}\} + (d_{\max} - d_{\min})\big)$$

that happen to be identical in this case. Here $r_{\min} = \min\{r_j \mid j \in J\}$, and similarly for $d_{\max}$ and $d_{\min}$, while $p_{\min} = \min\{p_{ij} \mid j \in J\}$. Analytical cuts have been developed for a wide variety of applications [14].

Explanation-based cuts require an explanation from the subproblem solver as to how optimality or infeasibility was proved. That is, they require information about the solution of the inference dual, as opposed to inferring such information indirectly as in the case of nogood and analytical cuts. Solvers are beginning to supply this kind of information, particularly constraint programming and satisfiability solvers. Explanations typically identify constraints that are essential premises of the optimality or infeasibility proof [4, 6, 5]. This also enables automatic generation of logic-based cuts, which have been implemented for the traveling salesman problem with time windows [17], as part of a constraint modeling language [10], and in a general LBBD solver [18] that implements branch and check (a variation of LBBD described in the next section).

7 Variations and Special Cases of LBBD

Branch and check [12, 20] is a variation of LBBD that solves the master problem only once, by means of a branching search. Logic-based Benders cuts are generated at search tree nodes when a feasible master problem solution is identified, at which point the branching search continues with the new cuts included in the constraint set. Branch and check is distinct from a traditional branch-and-cut method, because (a) the Benders cuts are derived from an external constraint set (the subproblem), and (b) the cuts contain variables that have already been fixed in the branching process rather than variables that have not been fixed. Branch and check is often an attractive alternative to standard LBBD when the master problem is too hard to solve repeatedly.

Combinatorial Benders cuts [9] are strengthened nogood feasibility cuts used in the context of a specialized branch-and-check procedure designed for mixed integer/linear programming (MILP). Although the subproblem is an LP problem in this case, logic-based cuts rather than classical Benders cuts are generated. The method was designed to accelerate solution of MILP models that contain big-M constraints and has found many applications.

The *conflict clauses* [1] that are key to the success of most propositional satisfiability (SAT) solvers are strengthened nogood feasibility cuts that take the form of logical clauses. Cuts are obtained from a conflict graph that represents a solution of the inference dual of the subproblem, where the inference method is unit propagation. Thus most SAT algorithms can be viewed as variations of LBBD in which the variable partition is dynamic: the master problem variables in a given iteration are those that are currently fixed in the search process. Methods for solving SAT modulo theories can also frequently be viewed as implementations of LBBD.

8 Conclusion

Decomposition into smaller problems is frequently the only practical approach to solving difficult optimization problems. Benders decomposition is especially attractive because its mechanism of Benders cuts preserves optimality by enabling efficient communication between the master problem and subproblem. In addition, the subproblem can be selected so as to be much easier to solve than the original problem, perhaps by decoupling into still smaller problems. The classical Benders method, however, is defined only for a linear programming subproblem, or a continuous nonlinear subproblem in its 1972 extension by Geoffrion [11]. By removing this restriction, logic-based Benders decomposition permits extension of this problem-solving strategy to a much broader range of practical applications.

An excellent survey of developments in Benders decomposition methods, including LBBD, can be found in [19]. A brief survey of the LBBD literature is [13]. A recent book [14] contains an exposition of LBBD and its variations, as well as concise descriptions of 226 published applications.

See also

Benders decomposition
Generalized Benders decomposition
Integer programming duality

References

[1] Beame P, Kautz H, Sabharwal A (2003) Understanding the power of clause learning. In: International Joint Conference on Artificial Intelligence (IJCAI 2003), pp 1194–1201

[2] Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4:238–252
[3] Benini L, Lombardi M, Mantovani M, Milano M, Ruggiero M (2008) Multi-stage Benders decomposition for optimizing multicore architectures. In: Perron L, Trick MA (eds) CPAIOR 2008 Proceedings, Springer, Lecture Notes in Computer Science, vol 5015, pp 36–50
[4] Cambazard H, Jussien N (2005) Identifying and exploiting problem structures using explanation-based costraint programming. In: Barták R, Milano M (eds) CPAIOR 2005 Proceedings, Springer, Lecture Notes in Computer Science, vol 3524, pp 94–109
[5] Cambazard H, Jussien N (2005) Integrating Benders decomposition within constraint programming. In: van Beek P (ed) Principles and Practice of Constraint Programming (CP 2005), Springer, Lecture Notes in Computer Science, vol 3668, pp 752–756
[6] Cambazard H, Jussien N (2006) Identifying and exploiting problem structures using explanation-based constraint programming. Constraints 11:295–313
[7] Carlier J, Pinson E (1989) An algorithm for solving the job-shop problem. Management Science 35:164–176
[8] Chinneck JW, Dravnieks EW (1991) Locating minimal infeasible constraint sets in linear programs. ORSA Journal on Computing 3:157–168
[9] Codato G, Fischetti M (2006) Combinatorial Benders cuts for mixed-integer linear programming. Operations Research 54:756–766
[10] Davies TO, Gange G, Stuckey PJ (2017) Automatic logic-based Benders decomposition with MiniZinc. In: Lübbecke M, Koster A, Letmangthe P, Madlener R, Peis B, Walther G (eds) AAAI Conferenece on Artificiaal Intelligence, pp 787–793
[11] Geoffrion AM (1972) Generalized Benders decomposition. Journal of Optimization Theory and Applications 10:237–260
[12] Hooker JN (2000) Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction. Wiley, New York
[13] Hooker JN (2019) Logic-based Benders decomposition for large-scale optimization. In: Velásquez-Bermúdez J, Khakifirooz M, Fathi M (eds) Large Scale Optimization in Supply Chains and Smart Manufacturing, Springer Optimization and Its Applications, vol 149, Springer, Berlin, pp 1–26
[14] Hooker JN (2023) Logic-Based Benders Decomposition: Theory and Applications. Springer, Berlin
[15] Hooker JN, Ottosson G (2003) Logic-based Benders decomposition. Mathematical Programming 96:33–60
[16] Junker U (2001) QuickXplain: Conflict detection for arbitrary constraint propagation algorithms. In: IJCAI01 Workshop on Modeling and Solving Problems with Constraints (CONS-1), Seattle, USA

[17] Lam E, Van Hentenryck P (2017) Branch-and-check with explanations for the vehicle routing problem with time windows. In: Beck JC (ed) Principles and Practice of Constraint Programming (CP 2017), Springer, Lecture Notes in Computer Science, vol 10416, pp 579–595

[18] Lam E, Gange G, Stuckey PJ, Van Hentenryck P, Dekker JJ (2020) Nutmeg: a MIP and CP hybrid solver using branch-and-check. SN Operations Research Forum 1

[19] Rahmaniani R, Crainic TG, Gendreau M, Rei W (2017) The Benders decomposition algorithm: A literature review. European Journal of Operational Research 259:801–817

[20] Thorsteinsson E (2001) Branch and check: A hybrid framework integrating mixed integer programming and constraint logic programming. In: Walsh T (ed) Principles and Practice of Constraint Programming (CP 2001), Springer, Lecture Notes in Computer Science, vol 2239, pp 16–30