Logic-Based Benders Decomposition *

J. N. Hooker
Graduate School of Industrial Administration
Carnegie Mellon University, Pittsburgh, PA 15213 USA

G. Ottosson
Department of Information Technology
Computing Science, Uppsala University
Box 311, S-751 05 Uppsala, Sweden

November 1995, Revised November 2000

Abstract

Benders decomposition uses a strategy of "learning from one's mistakes." The aim of this paper is to extend this strategy to a much larger class of problems. The key is to generalize the linear programming dual used in the classical method to an "inference dual." Solution of the inference dual takes the form of a logical deduction that yields Benders cuts. The dual is therefore very different from other generalized duals that have been proposed. The approach is illustrated by working out the details for propositional satisfiability and 0-1 programming problems. Computational tests are carried out for the latter, but the most promising contribution of logic-based Benders may be to provide a framework for combining optimization and constraint programming methods.

*This research was partially supported by U.S. Office of Naval Research Grant N00014-95-1-0517 and by the Engineering Design Research Center at Carnegie Mellon University, an Engineering Research Center of the National Science Foundation, under grant EEC-8943164.

Benders decomposition [7, 17] uses a problem-solving strategy that can be generalized to a larger context. It assigns some of the variables trial values and finds the best solution consistent with these values. In the process it learns something about the quality of other trial solutions. It uses this information to reduce the number of solutions it must enumerate to find an optimal solution. The strategy might described as "learning from one's mistakes."

The central element of Benders decomposition is the derivation of *Benders cuts* that exclude superfluous solutions. Classical Benders cuts are formulated by solving the dual of the *subproblem* that remains when the trial values are fixed. The subproblem must therefore be one for which dual multiplers are defined, such as a linear or nonlinear programming problem.

The key to generalizing Benders decomposition is to extend the class of problems for which a suitable dual can be formulated. We depart from previous generalizations of duality by defining an *inference dual* for any optimization problem. The solution of the inference dual is a proof of optimality within an appropriate logical formalism. Generalized Benders cuts are obtained by determining under what conditions the proof remains valid. Classical Benders decomposition can be seen to be a special case of this approach if it is viewed in a different light than the usual.

Logic-based Benders decomposition can be applied to any class of optimization problems, but a proof scheme and a method of generating Benders cuts must be devised for each class. We illustrate the method by applying it to propositional satisfiability, 0-1 programming problems, and a machine scheduling problem. The subproblems in these cases are not the traditional linear or nonlinear programming problems. One can therefore take advantage of special structure in the subproblems that is inaccessible to the traditional Benders method.

For example, a satisfiability or 0-1 programming problem may decouple into several smaller problems when certain variables are fixed. The subproblem can therefore be solved rapidly by solving its individual components, even though the components are themselves general satisfiability or 0-1 problems.

We address the satisfiability problem because it illustrates logic-based Benders in a lucid way and is an important problem in its own right. We examine the 0-1 programming problem because of the attention it has historically received. None of this should suggest, however, that logic-based Benders is applicable only to these problem classes.

For instance, Jain and Grossmann [31] recently solved machine schedul-

ing problems using a technique that is in effect logic-based Benders decoposition. Because the subproblems are one-machine scheduling problems, classical Benders cuts are unavailable. Jain and Grossmann achieved dramatic speedups in computation by solving the subproblems with constraint technology. This experience suggests that logic-based Benders can provide a natural medium for combining optimization and constraint programming. This idea is discussed in [27, 29].

The first section below reviews related work. Section 2 presents the basic idea of logic-based Benders decomposition. Section 3 introduces inference duality, and Section 4 shows how linear programming duality is a special case. The next two sections present logic-based Benders decomposition in the abstract, followed by its classical realization. Sections 7 and 8 apply logic-based Benders to the propositional satisfiability problem and 0-1 programming, respectively. Section 9 presents computational results for 0-1 programming, and Section 10 describes Jain and Grossmann's work. The final section is reserved for concluding remarks.

1 Related Work

Various types of generalized duality have been proposed over the years. Tind and Wolsey provided a survey in 1981 [43]. Much of this and subsequent work [2, 3, 8, 9, 13, 48, 49] is related to the superadditive duality of Johnson [35]. In 1981 Wolsey [50] used such a notion of duality as the basis for a generalized Benders algorithm in which the classical dual prices are replaced by a price function. Several other duals have been suggested for integer programming [4, 6, 15, 16, 40, 41]. A recent paper of Williams [47] examines a wide variety of duality concepts. Still more recently, Lagrangean and surrogate duals are interpreted in [27, 29] as forms of a relaxation dual.

The inference dual proposed here is fundamentally different from these earlier duals, because it regards the dual as an inference problem. Its solution is in general a proof, rather than a set of prices or a price function. The resulting generalization of Benders decomposition is therefore unlike Wolsey's.

The idea of inference duality might be traced ultimately to Jeroslow and Wang [33]. They showed that when linear programming demonstrates the unsatisfiability of a set of Horn clauses in propositional logic, the dual solution contains information about a unit resolution proof of unsatisfiability. (Unit resolution is defined in Section 7 below.) This introduces the key idea

that the dual solution can be seen as encoding (or partially encoding) a proof. It does not, however, show how to generalize the idea beyond a linear programming context, and a subsequent generalization focused on another type of linear programming problem, namely gain-free Leontief flows [32].

Hooker and Yan [30] introduced the logic-based Benders scheme described here, or a special case of it, in the context of logic circuit verification, and they presented computational results. Their method is that of Section 5 below specialized to logic circuits. After the first draft of the present paper was written (1995), Hooker [25] proposed inference duality as a basis for postoptimality analysis, and Dawande and Hooker [14] specialized the approach to sensitivity analysis for mixed integer programming. These ideas are presented in [27] as part of a general theoretical framework.

2 The Basic Idea

As already noted, Benders decomposition learns from its mistakes. A similar idea has been developed in a general way in the constraint programming literature under the rubric of *nogoods* ([45], Section 5.4; [34]). When in the process of solving a problem one deduces that a certain partial solution cannot be completed to obtain a feasible solution, one can examine the reasons for this. Often the same reasons lead to a constraint that excludes a number of partial solutions. Such a constraint is a nogood.

Benders decomposition uses a more specific strategy. It begins by partitioning the variables of a problem into two vectors x and y. It fixes y to a trial value so as to define a subproblem that contains only x. If the solution of the subproblem reveals that the trial value of y is unacceptable, the solution of the subproblem's *dual* is used to identify a number of *other* values of y that are likewise unacceptable. The next trial value must be one that has not been excluded. Eventually only acceptable values remain, and if all goes well, the algorithm terminates after enumerating only a few of the possible values of y. (The method is presented in more detail below.)

Because Benders decomposition searches for an optimal as well as a feasible solution, it actually enumerates trial values of (z, y), where z is the objective function value. Its "nogoods" have the form $z \geq \beta(y)$, where $\beta(y)$ is a bound on the optimal value that depends on y. The constraints $z \geq \beta(y)$ are known as *Benders cuts* and can rule out a large number of values of (z, y).

The specialized context of Benders decomposition enhances the general

strategy in two ways.

- The pre-arranged partition of variables can exploit problem structure. When y is fixed, the resulting subproblem may simplify substantially or decouple into a number of small subproblems.
- The linearity of the subproblem allows one to obtain a Benders cut in an easy and systematic way, namely by solving the linear programming dual.

The intent here is to generalize a Benders-like strategy while retaining these two advantages to a large degree. The key to doing so is to generalize the notion of a dual. The dual must be definable for any type of subproblem, not just linear ones, and must provide an appropriate bound on the optimal value.

Such a dual can be formulated simply by observing that a valid bound β on the optimal value is obtained by *inferring* it from the constraints. A generalized dual can therefore defined as an *inference dual*, which is the problem of inferring a strongest possible bound from the constraint set. The classical linear programming dual is the inference dual problem for linear optimization. A solution of the inference dual takes the form of a proof that β is in fact a bound on the optimal value.

In the context of Benders decomposition, the proof that solves the subproblem dual provides a valid bound β on the assumption that y is fixed to some particular value. But the same reasoning may deliver valid bounds when y takes other values. A constraint that imposes these bounds as a function of y becomes the logic-based Benders cut. It plays the same role as the classical Benders cut, although it may not take the form of an inequality constraint.

3 Inference Duality

Consider a general optimization problem,

$$\begin{array}{ll} \min & f(x) \\ \text{subject to} & x \in S \\ & x \in D \end{array} \tag{1}$$

where f is a real-valued function. The *domain* D is distinguished from the *feasible set* S. The domain might be the set of real vectors, 0-1 vectors, etc.

The feasible set S is generally defined by a collection of constraints, and it need not be a subset of D.

To state the inference dual it is necessary to define a semantic form of implication with respect to a domain. Let P and Q be two propositions whose truth or falsehood is a function of x. Then P *implies* Q *with respect to* D (notated $P \xrightarrow{D} Q$) if Q is true for any $x \in D$ for which P is true.

The *inference dual* of (1) is

$$\begin{array}{ll} \max & \beta \\ \text{s.t.} & x \in S \xrightarrow{D} f(x) \geq \beta \end{array} \tag{2}$$

The dual seeks the largest β for which $f(x) \geq \beta$ is implied by the constraint set. In other words, the dual problem is to find a tighest possible bound on the objective function value.

It is convenient to let the optimal value of a minimization problem be respectively ∞ or $-\infty$ when the problem is infeasible or unbounded, and vice-versa for a maximization problem. A strong duality property obviously holds for the inference dual: an optimization problem (1) always has the same optimal value as its inference dual (2).

Although the inference dual is very simple and satisfies a trivial form of strong duality, it suggests a different way of thinking about duality. It leads to a new generalized Benders algorithm as well as the new approach to sensitivity analysis mentioned earlier [14, 25]. It also suggests a different interpretation of the nontrivial strong duality theorems that appear in the optimization literature. Each such theorem identifies a method of proof for a particular logic and asserts that the method is sound and complete. (A proof method is *sound* when it obtains only valid inferences, and it is *complete* when it obtains all valid inferences.) Classical linear programming duality is an example of this.

4 Linear Programming Duality

The inference dual of a linear programming problem,

$$\begin{array}{ll} \min & cx \\ \text{s.t.} & Ax \geq a \\ & x \geq 0 \end{array} \tag{3}$$

can be stated,

$$\begin{array}{ll} \max & \beta \\ \text{s.t.} & \left(\begin{array}{c} Ax \geq a \\ x \geq 0 \end{array} \right) \xrightarrow{R^n} cx \geq \beta \end{array} \tag{4}$$

Linear inequalities over R^n can be regarded as forming a type of logic in which the inequalities are propositions, and nonnegative linear combinations provide a sound and complete proof method. More precisely, a feasible system $Ax \geq a, x \geq 0$ semantically implies $cx \geq \beta$ with respect to R^n if and only if there is a real vector $u \geq 0$ for which $uA \leq c$ and $ua \geq \beta$; i.e., if and only if the surrogate inequality $uAx \geq ua$ dominates $cx \geq \beta$ for some $u \geq 0$.

Another way to state this is that $Ax \geq a, x \geq 0$ implies $cx \geq \beta$ if and only if the classical dual problem has a solution u for which $ua \geq \beta$:

$$\begin{array}{ll} \max & ua \\ \text{s.t.} & uA \leq c \\ & u \geq 0 \end{array} \tag{5}$$

The classical dual therefore has the same optimal value as the inference dual (4) when $Ax \geq a, x \geq 0$ is feasible. This leads immediately (because of strong inference duality) to the classical strong duality theorem for linear programming: (5) has the same optimal value as (3), unless both are infeasible. Classical strong duality is therefore a way of stating the completeness of nonnegative linear combination as a proof method for the logic of linear inequalities. A feasible solution u of the classical dual can be interpreted as encoding a proof within this logic.

5 Benders Decomposition in the Abstract

Benders decomposition views elements of the feasible set as pairs (x, y) of objects that belong respectively to domains D_x, D_y. So the optimization problem (1) becomes,

$$\begin{array}{ll} \min & f(x, y) \\ \text{s.t.} & (x, y) \in S \\ & x \in D_x, y \in D_y \end{array} \tag{6}$$

A general Benders algorithm begins by fixing y at some trial value $\bar{y} \in$

D_y. This results in the *subproblem*,

$$\begin{array}{ll} \min & f(x, \bar{y}) \\ \text{s.t.} & (x, \bar{y}) \in S \\ & x \in D_x \end{array} \tag{7}$$

The inference dual of the subproblem is

$$\begin{array}{ll} \max & \beta \\ \text{s.t.} & (x, \bar{y}) \in S \xrightarrow{D_x} f(x, \bar{y}) \geq \beta \end{array} \tag{8}$$

The dual problem is to find the best possible lower bound β^* on the optimal cost that can be inferred from the constraints, assuming y is fixed to $\bar{y}$.

The heart of Benders decomposition is somehow to derive a *function* $\beta_{\bar{y}}(y)$ that gives a valid lower bound on the optimal value of (6) for *any* fixed value of y.

Just how this is done varies from one context to another, but essentially the idea is this. Let β^* be the value obtained for (8), which means that β^* is a lower bound on the optimal value of (6), given that $y = \bar{y}$. The solution of (8) is a proof of this fact. One then notes what valid lower bound *this same line of argument* yields for *other* values of y. This bound is expressed as a function $\beta_{\bar{y}}(y)$ of y, yielding a *Benders cut* $z \geq \beta_{\bar{y}}(y)$ The subscript $\bar{y}$ reflects which value of y gave rise to the bounding function.

The algorithm proceeds as follows. At each iteration the Benders cuts so far generated comprise the constraints of a *master problem*,

$$\begin{array}{ll} \min & z \\ \text{s.t.} & z \geq \beta_{y^k}(y), \;\; k = 1, \ldots, K \\ & y \in D_y \end{array} \tag{9}$$

Here $y^1, \ldots, y^K$ are the trial values of y hitherto obtained. The next trial value $(\bar{z}, \bar{y})$ of (z, y) is obtained by solving the master problem. If the optimal value β^* of the resulting subproblem dual (8) is equal to $\bar{z}$, the algorithm terminates with optimal value $\bar{z}$. Otherwise a new Benders cut $z \geq \beta_{\bar{y}}(y)$ is added to the master problem. The subproblem dual need not be solved to optimality if a suboptimal β is found that is better than $\bar{z}$. At this point the procedure repeats.

A precise statement of the algorithm appears in Fig. 1. Note that if the subproblem is infeasible, the dual is unbounded, so that $\beta^* = \beta_{\bar{y}}(\bar{y}) = \infty$.

```
Choose an initial ȳ ∈ D_y in problem (6).
Set z̄ = −∞ and k = 0.
While the subproblem dual (8) has a feasible solution β > z̄:
   Formulate a lower bound function β_ȳ(y) with β_ȳ(ȳ) = β.
   Let k = k + 1, let y^k = ȳ, and add the Benders cut
      z ≥ β_{y^k}(y) to the master problem (9).
   If the master problem (9) is infeasible then
      Stop; (6) is infeasible.
   Else
      Let ȳ be an optimal solution of (9) with
         optimal value z̄.
The optimal value of (6) is z̄.
```

Figure 1: *The generic Benders algorithm.*

Theorem 1 *Suppose that in each iteration of the generic Benders algorithm, the bounding function $\beta_{\bar{y}}$ satisfies the following.*

(B1) *The Benders cut $z \geq \beta_{\bar{y}}(y)$ is valid; i.e., any feasible solution (x, y) of (6) satisfies $f(x, y) \geq \beta_{\bar{y}}(y)$.*

Then if the algorithm terminates with a finite optimal solution $(z, y) = (\bar{z}, \bar{y})$ in the master problem, (6) has an optimal solution $(x, y) = (\bar{x}, \bar{y})$ with value $f(\bar{x}, \bar{y}) = \bar{z}$. If it terminates with an infeasible master problem, then (6) is infeasible. If it terminates with an infeasible subproblem dual, then (6) is unbounded.

Proof. Suppose first that the algorithm terminates with a finite solution $(z, y) = (\bar{z}, \bar{y})$ that is optimal in the master problem. Let β^* be the optimal value of the last subproblem dual (8) solved. Then the subproblem primal (7) has optimal value β^* and some optimal solution $\bar{x}$. Clearly β^* is an upper bound on the optimal value of (6). But because the algorithm terminated with a finite solution, $\beta^* = \bar{z}$. Finally, due to (B1), $\bar{z}$ is a lower bound on optimal value of (6). It follows that $\bar{z}$ is the optimal value of (6). Because $(\bar{x}, \bar{y})$ is feasible in (6), it is also optimal.

Because all Benders cuts are valid, infeasibility of the master problem implies that (6) is infeasible.

Finally, if the subproblem dual is infeasible, then the subproblem is unbounded. This implies that (6) is unbounded. □

The algorithm need not terminate in general, even if the subproblem is always solved to optimality. Consider for example the problem,

$$\begin{array}{ll} \min & y \\ \text{s.t.} & y \geq x \\ & x \geq 1 \\ & y \geq 0 \\ & x, y \in R, \end{array}$$

which has the optimal solution $(x, y) = (1, 1)$. It is consistent with the algorithm to set

$$\beta_{\bar{y}}(y) = \begin{cases} \infty & \text{if } y < \frac{1}{2}(1 + \bar{y}), \\ 0 & \text{otherwise.} \end{cases}$$

Then initially $(\bar{z}, \bar{y}) = (0, 0)$, and the solutions y^k of the master problem follow the sequence $1 - (\frac{1}{2})^k$ for $k = 1, 2, \ldots$, so that the algorithm never terminates.

If the domain of y is finite, however, the algorithm must terminate, because only finitely many subproblems can be defined. Because $\bar{z}$ must increase in every iteration but the last, the optimal value is reached after finitely many steps.

Theorem 2 *Suppose that in each iteration of the generic Benders algorithm, the bounding function $\beta_{\bar{y}}$ satisfies (B1) and the following.*

(B2) *$\beta_{\bar{y}}(\bar{y}) = \beta$, where β is the solution value obtained in the subproblem dual (8).*

Then if D_y is finite and the subproblem dual is solved to optimality, the generic Benders algorithm terminates.

Proof. Due to (B2), the subproblem value β must increase in every iteration but the last. Because D_y is finite and the subproblem dual is solved to optimality, the latter can generate only finitely many values. So the algorithm must terminate. □

6 Classical Benders Decomposition

The classical Benders technique applies to problems in which the subproblem is linear.

$$\begin{array}{ll} \min & cx + f(y) \\ \text{s.t.} & Ax + g(y) \geq a \\ & x \geq 0 \\ & x \in R^n, \;\; y \in D_y \end{array} \tag{10}$$

where $g(y)$ is a vector of functions $g_i(y)$. The subproblem (7) becomes,

$$\begin{array}{ll} \min & cx + f(\bar{y}) \\ \text{s.t.} & Ax \geq a - g(\bar{y}) \\ & x \geq 0 \end{array} \tag{11}$$

Due to classical strong duality, the subproblem dual (8) can be written,

$$\begin{array}{ll} \max & u(a - g(\bar{y})) + f(\bar{y}) \\ \text{s.t.} & uA \leq c \\ & u \geq 0 \end{array} \tag{12}$$

provided either (11) or (12) is feasible.

If the dual (12) has a finite solution u, then u provides an inference procedure (a linear combination) for obtaining a valid bound on the objective function value z of (10):

$$z \geq u(a - g(\bar{y})) + f(\bar{y})$$

that is valid when $y = \bar{y}$. The key to obtaining a bound for any y is to observe that this same u remains feasible in the dual for any y. So the same procedure (i.e., the same u) provides a bound

$$z \geq \beta_{\bar{y}}(y) = u(a - g(y)) + f(\bar{y})$$

for any y. This is the classical Benders cut. It is straightforward to show that when the dual is infeasible or unbounded, one can obtain a cut

$$v(a - g(y)) \leq 0,$$

where v is a ray that solves

$$\begin{array}{ll} \max & v(a - g(\bar{y})) \\ \text{s.t.} & vA \leq 0 \\ & v \geq 0 \end{array}$$

7 Propositional Satisfiability

The propositional satisfiability problem is to find an assignment of truth values to logical variables (*atomic propositions*) x_j for which a given set of logical *clauses* are all true. A clause is a disjunction of *literals*, each of which is an atomic proposition or its negation. For instance, $x_1 \vee \neg x_2 \vee \neg x_3$ is a clause asserting that x_1 is true or x_2 is false or x_3 is false. One clause C implies another D if and only of C *absorbs* D; i.e., C contains all the literals in D.

A logical clause may be denoted $C_i(x)$, where $x = (x_1, \ldots, x_n)$ is a vector of logical variables, and the variables appearing in $C_i(x)$ belong to $\{x_1, \ldots, x_n\}$. The satisfiability problem can be formulated as an optimization problem by giving it an objective function of zero.

$$\begin{array}{ll} \min & 0 \\ \text{s.t.} & C_i(x), \;\; i \in I \end{array} \tag{13}$$

The optimal value of (13) is zero if the clauses $C_i(x)$ are jointly satisfiable, and otherwise it is infinite (by convention).

The inference dual is,

$$\begin{array}{ll} \max & \beta \\ \text{s.t.} & \{C_i(x) \mid i \in I\} \overset{\{T,F\}^n}{\longrightarrow} 0 \geq \beta \end{array} \tag{14}$$

where T and F denote true and false, respectively. When the clauses are satisfiable, the maximum value of β is zero. Otherwise the antecedent of the implication is necessarily false and implies $0 \geq \beta$ for arbitrarily large β (because a necessarily false proposition implies everything). In this case the dual problem is unbounded.

Benders decomposition is applied when the propositional variables are partitioned (x, y). The satisfiability problem (13) becomes

$$\begin{array}{ll} \min & 0 \\ \text{s.t.} & C_i(x) \vee D_i(y), \;\; i \in I \end{array} \tag{15}$$

$C_i(x)$ represents the part of clause i containing variables in x, and $D_i(y)$ the part containing variables in y. Either part may be empty. In practice the variables would be partitioned so that the subproblem has special structure; this is discussed further below.

For a fixed assignment $\bar{y}$ of truth values to y, the subproblem is,

$$\begin{array}{ll} \min & 0 \\ \text{s.t.} & C_i(x), \;\; i \in I(\bar{y}) \end{array} \tag{16}$$

where $I(\bar{y})$ is the set of i for which $D_i(\bar{y})$ is false. When $D_i(\bar{y})$ is true, the clause $C_i(x) \vee D_i(\bar{y})$ is already satisfied and need not appear in the subproblem. The subproblem dual is,

$$\begin{array}{ll} \max & \beta \\ \text{s.t.} & \{C_i(x) \mid i \in I(\bar{y})\} \stackrel{\{T,F\}^n}{\longrightarrow} 0 \geq \beta \end{array} \tag{17}$$

As an example, consider the satisfiability problem,

$$\begin{array}{lrcrcrcrcrcl} \min & & & 0 & & & & & & & & \\ \text{s.t.} & x_1 & \vee & x_2 & & & \vee & y_1 & \vee & y_2 & & (a) \\ & x_1 & \vee & \neg x_2 & & & \vee & y_1 & & & & (b) \\ & \neg x_1 & & & \vee & x_3 & & & \vee & y_2 & & (c) \\ & & & x_2 & \vee & \neg x_3 & \vee & y_1 & \vee & y_2 & & (d) \\ & & & \neg x_2 & \vee & \neg x_3 & & & \vee & y_2 & & (e) \\ & \neg x_1 & \vee & x_2 & \vee & x_3 & \vee & y_1 & & & & (f) \\ & x_1 & \vee & x_2 & \vee & x_3 & \vee & \neg y_1 & & & \vee y_3 & (g) \end{array}$$

Initially the master problem (9) contains no constraints, and the solution $\bar{y}$ may be chosen arbitrarily. If $\bar{y} = (F, F, F)$, clauses (a)-(f) appear in the subproblem, with the y_j terms removed. The subproblem is therefore

$$\begin{array}{lrcrcrcl} \min & & & 0 & & & & \\ \text{s.t.} & x_1 & \vee & x_2 & & & & (a) \\ & x_1 & \vee & \neg x_2 & & & & (b) \\ & \neg x_1 & & & \vee & x_3 & & (c) \\ & & & x_2 & \vee & \neg x_3 & & (d) \\ & & & \neg x_2 & \vee & \neg x_3 & & (e) \\ & \neg x_1 & \vee & x_2 & \vee & x_3 & & (f) \end{array} \tag{18}$$

A branching method known as the Davis-Putnam-Loveland algorithm appears to be among the most effective complete algorithms for solving a

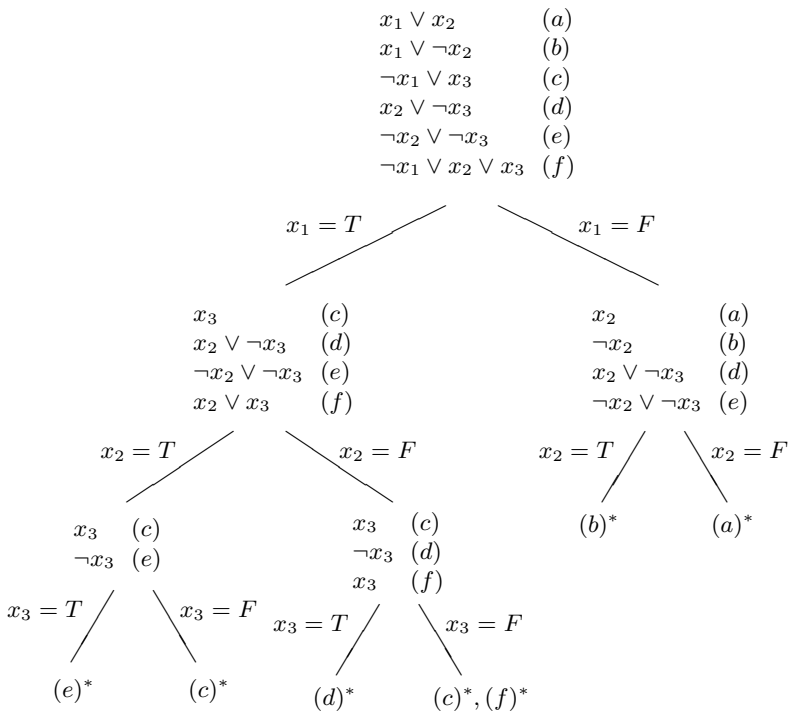

Figure 2: *A solution of the satisfiability subproblem by branching. The clauses remaining at each nonleaf node are indicated. The clauses falsified at each leaf node are indicated with an asterisk.*

satisfiability problem such as this (e.g., [12]). Figure 2 shows a straightforward branching tree that finds (18) to be unsatisfiable. The Davis-Putnam-Loveland algorithm adds a "unit resolution" step that will be discussed later.

The procedure in Figure 2 is simple. At the root node, the search branches on x_1 by setting x_1 to true and false. This generates two subproblems, corresponding to left and right child nodes respectively. Clauses (a) and (b) can be dropped in the left subproblem because $x_1 = T$ satisfies them. Also $\neg x_1$ is deleted from clause (f) because it cannot be true. The right subproblem is similarly simplified.

The search continues in recursive fashion until a feasible solution is found or the subproblem at every leaf node is shown to be unsatisfiable. A feasible

solution is found when the variables fixed so far at some node of the search tree satisfy every clause in the subproblem at that node. (A clause is satisfied when at least one literal in it is fixed to true.) A subproblem is shown to be unsatisfiable at a node when the variables fixed so far violate at least one clause of the subproblem (i.e., make every literal of the clause false).

If a feasible solution is found, the Benders algorithm immediately terminates with a feasible solution $(x, y) = (\bar{x}, \bar{y})$, where $\bar{x}_j$ is the truth value to which x_j is fixed at the node where a feasible solution is found. If x_j is not fixed, $\bar{x}_j$ may be chosen arbitrarily.

If no feasible solution is found, a Benders cut $z \geq \beta_{\bar{y}}(y)$ must be generated. This requires solution of the dual subproblem (17). Because the subproblem has an infinite optimal value, the dual solution must consist of a proof that $0 \geq \beta$ for arbitrarily large β, using the clauses $\{C_i(x) \mid i \in \bar{I}(\bar{y})\}$ as premises. In other words, it must show that $\{C_i(x) \mid i \in I(\bar{y})\}$ is unsatisfiable.

This can be done by applying a complete inference method for logical clauses, such as the *resolution* method discovered by W. V. Quine over 40 years ago [37, 38] (known as *consensus* when applied to formulas in disjunctive normal form). The name 'resolution' derives from Robinson [39], who developed the method for first-order predicate logic. However, this tends to be a very inefficient approach [18, 19].

It is more practical to observe that the implicit enumeration tree of Figure 2 already provides the desired proof of unsatisfiability, because at least one clause is falsified at every leaf node. In fact, if one associates a single falsified clause with each leaf node, the tree proves the unsatisfability of the subset of clauses associated with leaf nodes. (It is shown in [27] that this proof by enumeration is formally equivalent to a resolution proof, but this fact is not needed here.)

In the figure, it happens that only one clause is falsified at each leaf node, except for a node at which both (c) and (f) are falsified; suppose that (c) is associated with this node. Then the tree proves that clauses (a), (b), (c), (d) and (e) are jointly unsatisfiable. Let us denote the index set of these clauses by $\bar{I}(\bar{y})$, which is a subset of $I(\bar{y})$. Thus the dual solution uses only a subset of $\{C_i(x) \mid i \in I(\bar{y})\}$ as premises. This can result in a stronger Benders cut.

To find a Benders cut, it is necessary to identify all values of y for which this particular proof remains valid. This is easily done by observing which values of y falsify $D_i(y)$ for $i \in \bar{I}(\bar{y})$. So long as y falsifies $D_i(y)$, the clause $C_i(x)$ remains in the subproblem and is therefore falsified at its associated

nodes of the search tree in Figure 2. One can therefore write a Benders cut $z \geq \beta_{\bar{y}}(y)$ that states $z \geq \infty$ when all clauses $D_i(y)$ for $i \in \bar{I}(\bar{y})$ are false and states $z \geq 0$ otherwise. In the example, the cut states $z \geq \infty$ when y falsifies the following clauses.

$$
\begin{aligned}
& y_1 \vee y_2 \\
& y_1 \\
& y_2 \\
& y_1 \vee y_2 \\
& y_2
\end{aligned}
$$

It will be useful in the next section to use the following notation. If P is a proposition, define a function $v(P)$ to take the value 1 when P is true and 0 otherwise. Then the desired Benders cut can be written

$$z \geq \infty \cdot v\left(\bigwedge_{i \in \bar{I}(\bar{y})} \neg D_i(y)\right) \tag{19}$$

where $\wedge$ denotes a conjunction. Thus when all $D_i(y)$ for $i \in \bar{I}(\bar{y})$ are false, $z \geq \infty$. Otherwise $z \geq \infty \cdot 0 = 0$. In the example, the cut is

$$z \geq \infty \cdot v\left(\neg(y_1 \vee y_2) \wedge \neg y_1 \wedge \neg y_2\right)$$

This is equivalent to $z \geq \infty \cdot v\left(\neg y_1 \wedge \neg y_2\right)$, but it is not necessary to identify such reductions. In iteration K the master problem (9) is

$$
\begin{aligned}
\min \quad & z \\
\text{s.t.} \quad & z \geq \infty \cdot v\left(\bigwedge_{i \in \bar{I}(y^k)} \neg D_i(y)\right), \quad k = 1, \ldots, K \\
& y_j \in \{T, F\}, \text{ all } j
\end{aligned} \tag{20}
$$

In practice the master problem would be solved as a satisfiability problem in which each Benders cut (19) is written as a clause

$$\bigvee_{i \in \bar{I}(\bar{y})} D_i(y) \tag{21}$$

Clearly (21) is satisfied if and only if (19) does not force $z \geq \infty$. Thus (20) becomes

$$\bigvee_{i \in \bar{I}(y^k)} D_i(y), \quad k = 1, \ldots, K \tag{22}$$

```
Set k = 0.
While the master problem (22) has a feasible solution ȳ:
   Solve the subproblem (16) by branching.
   If (16) is infeasible then
      Associate with every leaf node t of the branching tree
         a falsified clause C_{i(t)}(x).
      Let Ī(ȳ) = {C_{i(t)} | all leaf nodes t}.
      Add the Benders cut (21) to the master problem (22).
      Set k = k + 1 and y^k = ȳ.
   Else stop; (15) is satisfiable with solution (x, y) = (x̄, ȳ),
      where x = x̄ solves the subproblem (16).
(15) is unsatisfiable.
```

Figure 3: *A Benders algorithm for the propositional satisfiability problem.*

In the example, (20) becomes the satisfiability problem

$$(y_1 \vee y_2) \vee y_1 \vee y_2 \tag{23}$$

which is of course equivalent to $y_1 \vee y_2$.

One solution of the master problem (23) is $(\bar{y}_1, \bar{y}_2, \bar{y}_3) = (F, T, F)$, for which the resulting subproblem is

$$\begin{aligned} &x_1 \vee \neg x_2 \\ &\neg x_1 \vee x_2 \vee x_3 \end{aligned}$$

The problem can be solved by setting $x_1 = x_2 = T$. Thus the algorithm terminates with solution $(x_1, x_2, x_3, y_1, y_2, y_3) = (T, T, x_3, F, T, F)$, where x_3 can be true or false.

The Benders algorithm for satisfiability appears in Figure 3. Because D_y is finite, Theorems 1 and 2 imply that the algorithm terminates with the correct answer if the Benders cuts have properties (B1) and (B2). It is easy to verify that they do.

The same approach can be used if the Davis-Putnam-Loveland algorithm is applied to the subproblems. This algorithm is simply the branching algorithm described earlier, except that *unit resolution* is applied at each node of the search tree. That is, if one of the clauses is a unit clause (contains exactly one literal), that literal is fixed to true. This in turn fixes the variable that occurs in the literal and allows the problem to be simplified. The process continues until no unit clauses remain.

Fixing a variable x_j to true or false in any step of unit resolution is equivalent to branching on x_j. Branching creates one subproblem in which a unit clause is violated, and a second subproblem that simplifies. The Davis-Putnam-Loveland algorithm is therefore formally equivalent to a pure branching algorithm, and Benders cuts can be generated accordingly.

There are various ways to accelerate the Benders algorithm. For example, it is advantageous to choose the same violated clause $C_{i(t)}(x)$ for several leaf nodes t. This makes the set $\{C_{i(t)}(x) \mid \text{all } t\}$ smaller and results in a stronger Benders cut.

Also, as noted earlier, the variables should be partitioned so that (a) the master problem is small and (b) the subproblem decouples into small problems or has some other special structure. For instance, a subproblem in which the clauses are renamable Horn can be checked for satisfiability in linear time [1, 10]. Finding a small subset of variables y for which the subproblem is always renamable Horn is a maximum clique problem [11], which can be solved by various heuristics and exact algorithms.

In practice the master problem as well as the subproblem might be solved by Davis-Putnam-Loveland. Whenever a feasible solution is found, the subproblem is solved, and the Benders cut (if any) is added to the master problem. At this point the master search tree can be updated to reflect the additional clause; an efficient algorithm for doing this is presented in [22]. Note that in the context of a branching algorithm, Benders cuts have a role complementary to that of traditional cutting planes. Whereas the latter contain variables that have not yet been fixed, Benders cuts contain variables that have already been fixed. They are also valid throughout the search tree, as are nogoods in general.

Interestingly, when the subproblem is renamable Horn, it is equivalent to a linear programming problem [33], so that the original problem could be solved by traditional Benders decomposition. But the linear-time methods for solving the subproblem are much faster than methods for solving the linear programming equivalent.

Furthermore, it is shown in [30] that if the variables y_j represent inputs to a logic circuit, and the variables x_j represent the outputs of gates in the circuit, then the problem of checking whether the circuit represents a tautology can be solved by Benders decomposition, where the subproblem is renamable Horn. Here again the subproblem is equivalent to a linear programming problem, and the specialized Benders algorithm developed in [30] yields Benders cuts that are in fact equivalent to those obtained by classical Benders—but again much more rapidly than in the classical case.

8 0-1 Programming

Benders decomposition has long been applied to 0-1 programming. The difference here is that the subproblem is a 0-1 programming problem, rather than a linear programming problem as in the classical case.

Because branching algorithms for integer programming problems usually solve a continuous relaxation of the problem at each node, this feature must be incorporated into the dual solution. In effect, the classical linear programming dual will be combined with a branch-and-bound search to obtain a dual solution.

A 0-1 programming problem may be stated,

$$\begin{array}{ll} \min & cx + c_0 \\ \text{s.t.} & Ax \geq a \\ & x \in \{0,1\}^n. \end{array} \tag{24}$$

The constant c_0 will be useful shortly. The inference dual is

$$\begin{array}{ll} \max & \beta \\ \text{s.t.} & Ax \geq a \overset{\{0,1\}^n}{\longrightarrow} cx + c_0 \geq \beta \end{array} \tag{25}$$

If β^* is the optimal value of (24), a solution of the dual (25) consists of a proof that $cx + c_0 \geq \beta^*$, using $Ax \geq a$ as premises.

If the variables are partitioned (x, y), a 0-1 problem can be written,

$$\begin{array}{ll} \min & cx + dy \\ \text{s.t.} & Ax + By \geq a \\ & x \in \{0,1\}^n,\ y \in \{0,1\}^p \end{array} \tag{26}$$

For a fixed $\bar{y}$ the subproblem is,

$$\begin{array}{ll} \min & cx + d\bar{y} \\ \text{s.t.} & Ax \geq a - B\bar{y} \\ & x \in \{0,1\}^n \end{array} \tag{27}$$

The subproblem dual is,

$$\begin{array}{ll} \max & \beta \\ \text{s.t.} & Ax \geq a - B\bar{y} \overset{\{0,1\}^n}{\longrightarrow} cx + d\bar{y} \geq \beta \end{array} \tag{28}$$

The subproblem (27) can be solved by branch and bound to obtain its optimal value β^*. Solution of the dual (28), however, is necessary to obtain a Benders cut. Its solution requires that one exhibit a proof of $cx + d\bar{y} \geq \beta^*$ using $Ax \geq a - B\bar{y}$ as premises. One approach is to use a complete inference method for 0-1 linear inequalities, such as the generalized resolution method of [21]. Such a method in fact serves as the basis for a logic-based solution method for 0-1 programming [5].

A more direct approach, however, is to interpret the branch-and-bound tree for the primal problem (27) as a proof of optimality. As in the satisfiability problem, one can associate a violated inequality I_t with each leaf node t of the tree. I_t implies a logical clause C_t that is falsified at that node. The optimality proof remains valid for any y such that I_t continues to imply C_t. This in turn provides the basis for a Benders cut.

This may be made precise as follows. At leaf node t, let J_1 be the set of all j for which branching has set x_j to 1, and similarly for J_0. Then the following clause C_t is violated by the fixed variables.

$$\bigvee_{j \in J_0} x_j \vee \bigvee_{j \in J_1} \neg x_j \tag{29}$$

The continuous relaxation of the problem at leaf node t may be written,

$$\begin{array}{lll} \min & cx + d\bar{y} & \\ \text{s.t.} & Ax \geq a - B\bar{y} & (u) \\ & Hx \geq h & (v) \\ & x \in \{0,1\}^n & \end{array} \tag{30}$$

The system $Hx \geq h$ contains upper bounds of the form $-x_j \geq -1$ as well as constraints $-x_j \geq 0$ that fix x_j to 0 for $j \in J_0$, and constraints $x_j \geq 1$ that fix x_j to 1 for $j \in J_1$. Dual variables u, v may be associated with the constraints as shown.

We will need the following lemma, whose proof is straightforward. Let $\alpha^+ = \max\{\alpha, 0\}$.

Lemma 3 *A 0-1 inequality $ax \geq \alpha$ implies C_t if and only if*

$$\sum_{j \in J_1} a_j + \sum_{j \notin J_0 \cup J_1} a_j^+ < \alpha$$

When the relaxation (30) is solved at node t, there are three possible outcomes.

(a) The relaxation is infeasible. In this case the dual solution (u, v) is such that

$$\begin{aligned} &uA + vH \leq 0 \\ &u(b - B\bar{y}) + vh > 0 \end{aligned}$$

Thus the following system is infeasible.

$$\begin{aligned} &uA \geq u(b - B\bar{y}) \\ &Hx \geq h \\ &x \geq 0 \end{aligned}$$

This means that the fixed variables violate the surrogate inequality

$$uAx \geq ua - uB\bar{y} \tag{31}$$

In fact, the fixed variables violate any inequality that implies C_t. They therefore violate $uAx \geq ua - uBy$ for any y such that this inequality implies C_t. By Lemma 3, $uAx \geq ua - uBy$ implies C_t if and only if

$$\sum_{j \in J_1} uA_j + \sum_{j \notin J_0 \cup J_1} (uA_j)^+ < ub - uBy$$

where A_j is column j of A. This can be written

$$uBy < ub - \sum_{j \in J_1} uA_j - \sum_{j \notin J_0 \cup J_1} (uA_j)^+ \tag{32}$$

Let this be inequality I_t.

(b) The relaxation is feasible and the solution is integral. Then if the optimal value is β_t^*, the dual solution satisfies

$$\begin{aligned} &uA + vH \leq c \\ &u(b - B\bar{y}) + vh \geq \beta_t^* \end{aligned}$$

Thus the following system is infeasible.

$$\begin{aligned} &(uA - c)x > ub - uB\bar{y} - \beta_t^* \\ &Hx \geq h \\ &x \geq 0 \end{aligned}$$

This means that the fixed variables violate the surrogate inequality

$$(uA - c)x > ub - uB\bar{y} - \beta_t^* \tag{33}$$

They also violate $(uA - c)x > \beta_t^* - ub + uBy$ for any y such that this inequality implies C_t. Using Lemma 3, C_t is implied if and only if

$$uBy \leq ub - \beta_t^* - \sum_{j \in J_1} (uA_j - c_j) - \sum_{j \notin J_0 \cup J_1} (uA_j - c_j)^+ \tag{34}$$

Let this be inequality I_t.

(c) The relaxation is feasible and has a nonintegral solution. Its optimal value $\bar{\beta}_t$ is greater than or equal to the value of the incumbent solution. The analysis is the same as in (b) with $\bar{\beta}_t$ replacing β_t^*. Thus the fixed variables again violate the surrogate (37) and the inequality I_t is

$$uBy \leq ub - \bar{\beta}_t - \sum_{j \in J_1} (uA_j - c_j) - \sum_{j \notin J_0 \cup J_1} (uA_j - c_j)^+ \tag{35}$$

The optimal value β^* of the subproblem is the minimum over all values β_t^* obtained at leaf nodes t. The dual solution exhibits, at every leaf node, a surrogate inequality (31) or (33) that is (a) violated, (b) violated when the value of the relaxation is assumed to be less than β_t^*, or (c) violated when the value of the relaxation is assumed to be less than $\bar{\beta}_t$. Given any y for which each leaf node's surrogate remains violated, the branch-and-bound tree proves optimality of β^*. But such a y is precisely one that satisfies the inequalities I_t for each t. A Benders cut may therefore be stated,

$$z \geq dy + \sum_j c_j^- + \left(\beta^* - \sum_j c_j^- \right) v \left(\bigwedge_{t \in T} I_t \right) \tag{36}$$

where T is the set of leaf nodes. The term $\sum_j c_j^-$ simply computes the smallest possible value of cx and therefore provides a default bound when y does not satisfy all of the I_t's.

A somewhat sharper analysis is possible when the subproblem (27) decouples into smaller problems. Suppose then that (27) can be written

$$\begin{aligned} \min \quad & \sum_{i=1}^{p} c^i x^i + d\bar{y} \\ \text{s.t.} \quad & A^i x^i \geq a^i - B^i \bar{y}, \quad i = 1, \ldots p \\ & x^i \in \{0,1\}^{n_i}, \quad i = 1, \ldots, p \end{aligned} \tag{37}$$

where the vectors x^i have no variables in common. The subproblem is solved by solving

$$\begin{array}{ll} \min & c^i x^i \\ \text{s.t.} & A^i x^i \geq a^i - B^i \bar{y} \\ & x^i \in \{0,1\}^{n_i} \end{array} \tag{38}$$

for $i = 1, \ldots, p$ to obtain optimal solutions $\bar{x}^i$ and optimal values $(\beta^*)^i$. An optimal solution of (37) can now be written $\bar{x} = (\bar{x}^1, \ldots, \bar{x}^p)$, and the optimal value is $\beta^* = \sum_{i=1}^p (\beta^*)^i + d\bar{y}$.

Let I_t^i be the inequality I_t obtained at leaf node t of the branch-and-bound tree that solves (38). Then the Benders cut (36) becomes

$$z \geq dy + \sum_j c_j^- + \sum_{i=1}^p \left((\beta^*)^i - \sum_{j \in J^i} c_j^- \right) v \left(\bigwedge_{t \in T_i} I_t^i \right) \tag{39}$$

where T_i is the set of leaf nodes of the search tree that obtained $(\beta^*)^i$. Also J^i is the set of indices of variables in x^i.

Consider the following example.

$$\begin{array}{lllllll} \min & 4x_1 + 2x_2 + 5x_3 + & x_4 + & y_1 + 20y_2 & & \\ \text{s.t.} & 2x_1 + & x_2 & & + 2y_1 + & y_2 \geq 5 & (a) \\ & & & 3x_3 + 2x_4 & + & y_2 \geq 4 & (b) \\ & x \in \{0,1\}^4, & y \in \{0,1\}^2 & & & & \end{array} \tag{40}$$

Initially the master problem has no constraints, and one may choose the optimal solution $(y_1, y_2) = (0, 0)$. Setting $\bar{y} = (0, 0)$ yields a subproblem of the form (37) that decouples into two smaller problems:

$$\begin{array}{lll} \min & 4x_1 + 2x_2 & \\ \text{s.t.} & 2x_1 + x_2 \geq 5 & \\ & x_1, x_2 \in \{0,1\} & \end{array} \qquad \begin{array}{lll} \min & 5x_3 + x_4 & \\ \text{s.t.} & 3x_3 + 2x_4 \geq 4 & \\ & x_3, x_4 \in \{0,1\} & \end{array} \tag{41}$$

The first subproblem has the following relaxation at the root node of a branch-and-bound tree.

$$\begin{array}{lll} \min & 4x_1 + 2x_2 & \\ \text{s.t.} & 2x_1 + x_2 \geq 5 & (u) \\ & -x_1 \geq -1 & (v_1) \\ & -x_2 \geq -1 & (v_2) \\ & x_1, x_2 \in \{0,1\} & \end{array}$$

The relaxation is infeasible, with dual solution $u = 1$, $v = (2, 1)$. The root node is therefore the only leaf node, at which the violated surrogate (31) is $2x_1 + x_2 \geq 5$. The Inequality I_1^1, given by (32), is $2y_1 + y_2 < 2$.

The second problem in (41) has a fractional solution $(x_3, x_4) = (\frac{2}{3}, 1)$ at the root node, and the search therefore branches on x_3. In the branch defined by $x_3 = 0$, the problem is infeasible and generates the inequality I_1^2 given by $y_2 < 2$. Because this inequality is necessarily satisfied, it can be replaced by the tautologous inequality $0 < 1$. The branch defined by $x_3 = 1$ has the fractional solution $(x_3, x_4) = (1, \frac{1}{2})$, which requires a further branch on x_4. This creates an infeasible node and a node at which the relaxation has an integral solution $(x_3, x_4) = (1, 1)$.

The complete algorithm is summarized in Table 1. After the first Benders cut is added to the master problem, the solution of the latter is $\bar{y} = (1, 1)$. This generates a second Benders cut, and the master problem that contains the first two Benders cuts has solution $\bar{y} = (1, 0)$. This generates a third Benders cut, which when added to the master problem yields $\bar{y} = (1, 1)$ with optimal value 11. The subproblem resulting from $\bar{y} = (1, 1)$ has already been solved, with optimal value 11. The algorithm therefore terminates with solution $(x, y) = (0, 1, 0, 1, 1, 1)$.

9 Computational Testing

The task of testing Benders decomposition computationally is somewhat problematic, since it is not a general-purpose method. It is intended for problems with special structure, and its performance depends on the degree to which decomposition can exploit the structure.

One approach to testing is to exhibit an important application area in which problems are structured so that Benders is more effective than other known techniques. One such application area is reported in the next section.

Another approach is to focus on a particular type of structure that a Benders method can exploit. The simplest case occurs when a subproblem separates into smaller problems. We investigated to what extent the subproblem of a 0-1 programming problem must decouple before a Benders approach becomes advantageous.

We randomly generated instances in which the subproblem decouples into m problems of size $s \times s$, and the master problem has s variables. The subproblem has no other special structure. Thus each A^i in (37) is $s \times s$, each B^i is $s \times s$, and the original problem is $ms \times (m + 1)s$. A larger m

```
Choose an initial ȳ ∈ {0,1}^n.
Set z̄ = −∞ and k = 0.
While the subproblem (27) has optimal value β* > z̄:
   For each leaf t of the search tree used to solve (27):
      If the relaxation (30) is infeasible then
         Let I_t be (32).
      Else if (30) has an integral optimal solution then
         Let I_t be (34).
      Else if (30) has a nonintegral optimal solution then
         Let I_t be (35).
   Add the Benders cut (36) to the master problem (9).
   Let k = k + 1 and y^k = ȳ.
   If the master problem (9) is infeasible then
      Stop; (26) is infeasible.
   Else let ȳ be an optimal solution of (9) with
      optimal value z̄.
(x, y, z) = (x̄, ȳ, z̄) is an optimal solution of (26), where
   x = x̄ is an optimal solution of the last subproblem solved.
```

Figure 4: *A Benders algorithm for 0-1 programming.*

implies a more highly decoupled subproblem, whereas $m = 1$ implies no decoupling at all.

The Benders approach requires that when the subproblem is solved, the dual solution be available at each leaf node of the search tree. Because commercial software available to us did not provide this information, we solved the subproblems with a straightforward branch-and-bound algorithm written in Java, in which CPLEX solved the linear programming relaxations.

We applied exactly the same branch-and-bound algorithm to the original problem (without decomposition) and compared the computation time to that of the Benders algorithm. This provided a controlled experiment in which the effect of decomposition could be isolated, a general approach advocated in [24, 26].

Because the master problem does not have the traditional inequality constraints, we solved it with a modified branch-and-bound algorithm. We branched on y_j's with fractional values as well as on the possible values of the right-hand side of each Benders cut (39).

More precisely, for $k = 1, \ldots K$ we wrote the kth Benders cut in the

```
Choose an initial ȳ ∈ {0,1}^n.
Set z̄ = −∞ and k = 0.
While the subproblem (27) has optimal value β* > z̄:
   For each component (37) of the subproblem (27):
      For each leaf t of the search tree used to solve (37):
         If the relaxation (30) is infeasible then
            Let I_t^i be (32).
         Else if (30) has an integral optimal solution then
            Let I_t^i be (34).
         Else if (30) has a nonintegral optimal solution then
            Let I_t^i be (35).
   Add the Benders cut (39) to the master problem (9).
   Let k = k + 1 and y^k = ȳ.
   If the master problem (9) is infeasible then
      Stop; (26) is infeasible.
   Else let ȳ be an optimal solution of (9) with
      optimal value z̄.
(x, y, z) = (x̄, ȳ, z̄) is an optimal solution of (26), where
   x = x̄ is an optimal solution of the last subproblem solved.
```

Figure 5: *A Benders algorithm for 0-1 programming in which the subproblem decouples into problems (37) for $i = 1, \ldots, p$.*

master problem (9) in the form

$$z \geq dy + \sum_j c_j^- + \sum_{i=1}^p z_{ik} \tag{42}$$

For each Benders cut k, we enforced the following p disjunctions.

$$\left(z_{ik} = (\beta^*)^i - \sum_{j \in J^i} c_j^- \right) \vee \bigvee_{t \in T_i} \left((z_{ik} = 0) \wedge \neg I_t^i \right), \quad i = 1, \ldots, p \tag{43}$$

To keep the problem small, we did not actually use the variables z_{ik} in (42). Rather, we replaced each z_{ik} with $(\beta^*)^i - \sum_{j \in J^i} c_j^-$ if the first disjunct of (43) was enforced and with zero otherwise. Thus the linear relaxation of the master problem minimizes z subject to the following inequality constraints.

(a) The Benders cuts (42), with each z_{ik} replaced by $(\beta^*)^i - \sum_{j \in J^i} c_j^-$ or zero.

(b) For each Benders cut and each $i = 1, \ldots, p$, the inequality constraint $\neg I_t^i$ if the disjunct $(z_{ik} = 0) \wedge \neg I_t^i$ of (43) is enforced.

(c) Constraints $y_j \leq 0$ or $y_j \geq 1$ for each y_j that has been fixed to 0 or 1 (respectively) by prior branching.

(d) The bounds $0 \leq y_j \leq 1$ for each y_j.

The inequality $\neg I_t^i$ is written $f \geq g$ if I_t^i has the form $f < g$, and it is written $f \geq g + \epsilon$ if I_t^i has the form $f \leq g$.

At each node of the branch-and-bound tree, we branched according to the following rules.

(i) If the the solution of the linear relaxation just described is noninteger, branch on a y_j with value closest to 1/2.

(ii) Otherwise, branch on one of the remaining disjunctions (43) in which the current values of the y_j's satisfy none of the last $|T_i|$ disjuncts, and remove this disjunction from the problem. Enforce one of the $|T_i| + 1$ disjuncts of (43) in each branch.

(iii) If no such disjunctions remain, the current solution is feasible and no branching is necessary.

Figures 6–9 show the results of the experiments. For subproblem sizes $s = 2, 3, 4$ we generated problem instances for several values of m. We solved each instance with both Benders (solid curve in the figure) and standard branch and bound (dashed curve). Each point plotted represents the average computation time over 10 instances. Note that the vertical axis has a logarithmic scale.

The crossover point m^* of the two curves is the value of m at which the Benders approach becomes superior to standard branch and bound. (For $s = 4$ we were unable to solve problems with $m > 25$.) The crossover point is rather large, with m^* equal to 20 or more. However, once the Benders approach becomes superior, its superiority rapidly becomes ovewhelming. With $s = 3$, for example, the Benders approach is on a par with the traditional approach for $m = 21$ but is already an order of magnitude faster for $m = 25$.

10 An Application to Machine Scheduling

A particularly interesting role for logic-based Benders decomposition is as a framework for combining optimization and constraint programming, as

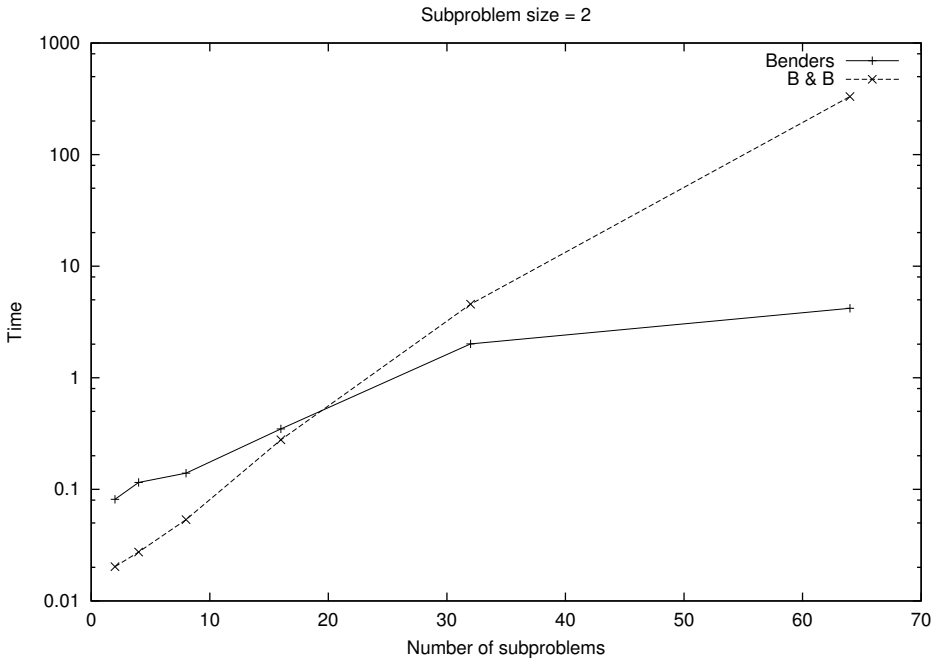

Figure 6: *Computation time in seconds versus number m of subproblem components for logic-based Benders decomposition and traditional branch and bound. Components have $s = 2$ variables and 2 constraints.*

proposed in [29]. Jain and Grossmann [31] illustrate how this might be done in their solution of a machine scheduling problem. Thorsteinsson and Hooker [42] use a similar framework to solve vehicle routing problems with time windows.

In the Jain and Grossmann application, the master problem assigns jobs to machines, and the subproblem tries to schedule jobs on their assigned machine. The master problem is given a mixed integer programming model, while the subproblem is attacked with the highly-developed scheduling technology of constraint programming (in particular, the ILOG Scheduler as invoked by OPL Studio [46]).

The problem may be stated as follows. Each job j is assigned to one of several machines i that operate at different speeds. Each assignment results in a processing time D_{ij} and incurs a processing cost C_{ij}. There is a release date R_j and a due date S_j for each job j. The objective is to minimize processing cost while observing release and due dates.

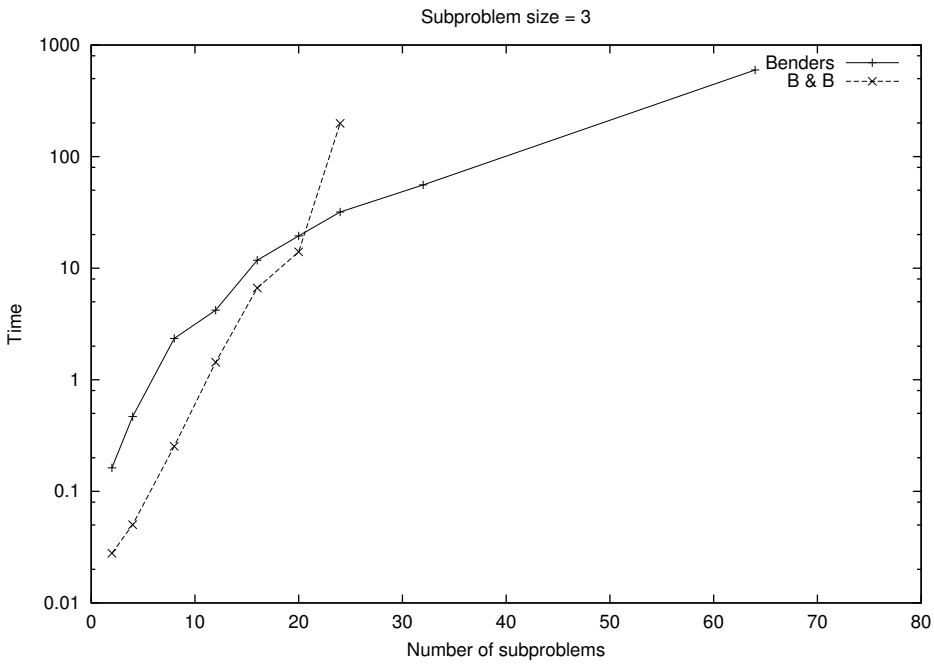

Figure 7: *Computation time in seconds versus number m of subproblem components for logic-based Benders decomposition and traditional branch and bound. Components have $s = 3$ variables and 3 constraints.*

Let t_j be the time at which job j begins processing, and let y_{ij} be 1 if job j is assigned to machine i and 0 otherwise. For a given machine i let $((t_j, D_{ij}) \mid y_{ij} = 1)$ be the list of pairs (t_j, D_{ij}) for which job j is assigned to machine i. The model is

$$\begin{array}{ll} \min & \sum_{ij} C_{ij} y_{ij} \\ \text{s.t.} & t_j \geq R_j, \ \text{ all } j \\ & t_j + \sum_i D_{ij} y_{ij} \leq S_j, \ \text{ all } j \\ & \text{nonoverlap}((t_j, D_{ij}) \mid y_{ij} = 1), \ \text{ all } i \\ & t \geq 0, \ \ y_{ij} \in \{0, 1\} \end{array}$$

The *nonoverlap* constraint requires that $t_j + D_{jk} \leq t_k$ for all jobs j, k assigned to a given machine i.

The master problem contains variables t_j, y_{ij}. To create a subproblem,

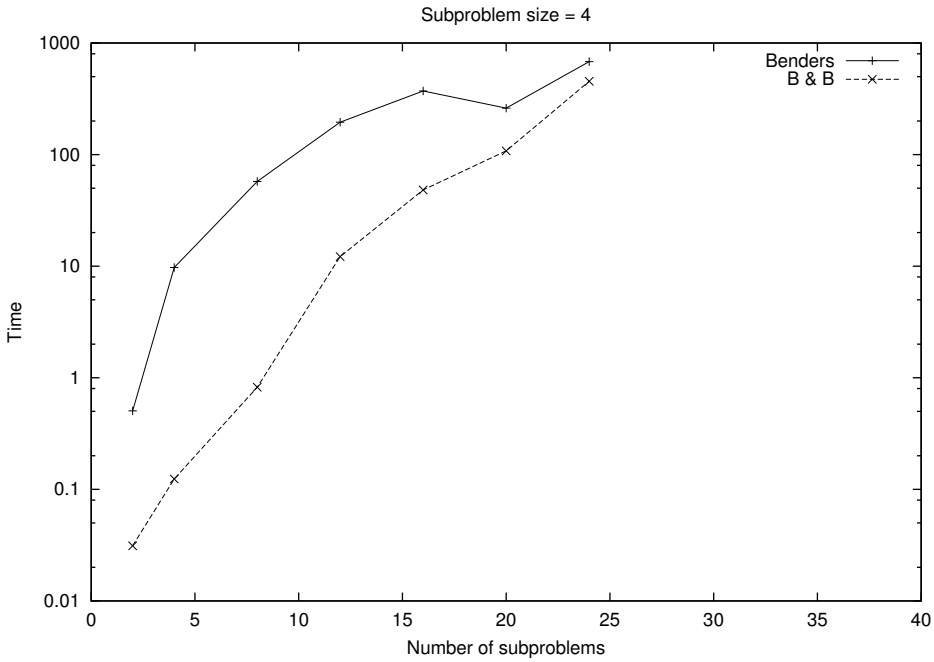

Figure 8: *Computation time in seconds versus number m of subproblem components for logic-based Benders decomposition and traditional branch and bound. Components have $s = 4$ variables and 4 constraints.*

introduce discrete variables t'_j that have the same meaning as t_j. The subproblem contains the release times and deadlines along with the nonoverlap constraint. So if the assignments are fixed to $\bar{y}$, the subproblem becomes the following scheduling problem:

$$\begin{array}{ll} \min & 0 \\ \text{s.t.} & t'_j \geq R_j, \ \text{ all } j \\ & t'_j + D_{ij} \leq S_j, \ \text{ all } j \text{ with } \bar{y}_{ij} = 1, \text{ all } i \\ & \text{nonoverlap}\Big((t'_j, D_{ij}) \mid \bar{y}_{ij} = 1\Big), \ \text{ all } i \end{array}$$

The problem separates into a feasibility problem for each machine i:

$$\begin{array}{ll} \min & 0 \\ \text{s.t.} & t'_j \geq R_j, \ \text{ all } j \text{ with } \bar{y}_{ij} = 1 \\ & t'_j + D_{ij} \leq S_j, \ \text{ all } j \text{ with } \bar{y}_{ij} = 1 \\ & \text{nonoverlap}\Big((t'_j, D_{ij}) \mid \bar{y}_{ij} = 1\Big) \end{array} \tag{44}$$

The nonoverlap constraint can implemented by standard "global" constraints available in constraint programming systems. In the ILOG scheduler, it is implemented by the *cumulative* constraint. The variables y'_j are discrete because the cumulative constraint is designed for variables with finite domains.

Let I_k be the set of machines i for which (44) is infeasible in the k-th iteration of the Benders algorithm, and let $J_{ik} = \{j \mid \bar{y}_{ij} = 1\}$ be the set of jobs assigned to machine i. For each $i \in I_k$, infeasibility implies that the jobs in J_{ki} cannot all be scheduled on machine i. This gives rise to a Benders cut $\sum_{j \in J_{ik}} (1 - y_{ij}) \geq 1$ for each $i \in I_k$. Going beyond Jain and Grossmann, one can strengthen the cut by identifying a proper subset J_{ik} of jobs assigned to machine i that cannot feasibly be scheduled on that machine. The proof of infeasibility obtained by the constraint programming algorithm may reveal such a smaller set.

The Benders cuts go into the master problem,

$$\begin{array}{ll} \min & \displaystyle\sum_{ij} C_{ij} y_{ij} \\ \text{s.t.} & t_j \geq R_j, \ \text{ all } j \\ & t_j + \displaystyle\sum_i D_{ij} y_{ij} \leq S_j, \ \text{ all } j \\ & \displaystyle\sum_{j \in J_{ik}} (1 - y_{jk}) \geq 1, \ \text{ all } i \in I_k, \text{ all } k \\ & t \geq 0, \ \ y_{ij} \in \{0, 1\} \end{array}$$

where k ranges over all the Benders iterations carried out so far. The algorithm terminates in the first iteration k for which I_k is empty.

Using this approach, Jain and Grossmann obtained substantial speedups relative to constraint programming and mixed integer programming.

11 Conclusion

An elementary theory of logic-based Benders decomposition has been developed and applied to three problems: the propositional satisfiability problem,

the 0-1 programming problem, and a machine scheduling problem. The last illustrates how Benders decomposition can provide a principle for combining optimization with constraint programming.

A logic-based Benders approach can in principle be applied to any optimization or feasibility problem, because inference duality is defined for any such problem. Its success depends on the extent to which

- an easily solved subproblem can be obtained by a judicious partitioning of the variables, and
- the solution of the subproblem dual with y fixed to a particular value is a proof whose line of reasoning yields a useful lower bound (Benders cut) for other values of y.

Logic-based Benders decomposition may also have potential in a branch-and-cut context, as mentioned in connection with the satisfiability problem above. Cuts presently used typically involve variables that have not yet been fixed in the enumeration tree and serve primarily to strengthen a linear or some other relaxation. Logic-based Benders cuts, by contrast, would involve variables that are already fixed and would apply throughout the tree. They would prune the tree in a way that is unrelated to any relaxation.

Table 1: Solution of a 0-1 problem by logic-based Benders decomposition.

$\bar{y}$	i	Leaf node defined by	Relaxation $0 \le x_j \le 1, j \in J^i$	Solution x of relaxation	u	Inequality I_t^i
$(0,0)$ $z=0$	1	root	min $4x_1+2x_2$ s.t. $2x_1+x_2 \ge 5$	infeasible	1	I_1^1 $2y_1+y_2<2$
	2	$x_3=0$	min $5x_3+x_4$ s.t. $3x_3+2x_4 \ge 4$ $-x_3 \ge 0$	infeasible	1	I_1^2 $0<1$
	2	$(x_3,x_4)=(1,0)$	min $5x_3+x_4$ s.t. $3x_3+2x_4 \ge 4$ $x_3 \ge 1, -x_4 \ge 0$	infeasible	1/2	I_2^2 $y_2<1$
	2	$(x_3,x_4)=(1,1)$	min $5x_3+x_4$ s.t. $3x_3+2x_4 \ge 4$ $x_3 \ge 1, x_4 \ge 1$	integral $(x_3,x_4)=(1,1)$ $z^*=6$	0	I_3^2 $0 \le 0$
	Value of relaxation: ∞ Benders cut: $z \ge y_1+y_2+\infty \cdot v(2y_1+y_2<2)+6v(y_2<1)$					
$(1,1)$ $z=2$	1	$x_1=0$	min $4x_1+2x_2$ s.t. $2x_1+x_2 \ge 2$ $-x_1 \ge 0$	infeasible	1/2	I_1^1 $0<1$
	1	$x_1=1$	min $4x_1+2x_2$ s.t. $2x_1+x_2 \ge 2$ $x_1 \ge 1$	integral $(x_1,x_2)=(1,0)$ $z^*=4$	2	I_2^1 $2y_1+y_2 \le 3$
	2	$x_3=0$	min $5x_3+x_4$ s.t. $3x_3+2x_4 \ge 3$ $-x_3 \ge 0$	infeasible	1/3	I_1^2 $0<1$
	2	$x_3=1$	min $5x_3+x_4$ s.t. $3x_3+2x_4 \ge 4$ $x_3 \ge 1$	integral $(x_3,x_4)=(1,0)$ $z^*=5$	1/2	I_2^2 $y_2 \le 1$
	Value of relaxation: 11 Benders cut: $z \ge y_1+y_2+4v(2y_1+y_2 \le 3)+5v(y_2 \le 1)=y_1+y_2+9$					
$(1,0)$ $z=10$	1	root	min $4x_1+2x_2$ s.t. $2x_1+x_2 \ge 3$	integral $(x_1,x_2)=(1,1)$ $z^*=6$	2	I_1^1 $2y_1+y_2 \le 2$
	2	$x_3=0$	min $5x_3+x_4$ s.t. $3x_3+2x_4 \ge 4$ $-x_3 \ge 0$	infeasible	1	I_1^2 $0<1$
	2	$(x_3,x_4)=(1,0)$	min $5x_3+x_4$ s.t. $3x_3+2x_4 \ge 4$ $x_3 \ge 1, -x_4 \ge 0$	infeasible	1/2	I_2^2 $y_2<1$
	2	$(x_3,x_4)=(1,1)$	min $5x_3+x_4$ s.t. $3x_3+2x_4 \ge 4$ $x_3 \ge 1, x_4 \ge 1$	integral $(x_3,x_4)=(1,1)$ $z^*=6$	0	I_3^2 $0 \le 0$
	Value of relaxation: 13 Benders cut: $z \ge y_1+y_2+6v(2y_1+y_2 \le 2)+6v(y_2<1)$					
$(1,1)$ $z=11$	Value of relaxation: 11 (solved before)					

References

[1] Aspvall, B., Recognizing disguised NR(1) instance of the satisfiability problem, *Journal of Algorithms* **1** (1980) 97-103.

[2] Bachem, A., and R. Schrader, A duality theorem and minimal inequalities in mixed integer programming, *Zeitschrift für angewandte Mathematik und Mechanik* **59** (1979) T88-T89.

[3] Bachem, A., and R. Schrader, Minimal inequalities and subadditive duality, *SIAM Journal on Control and Optimization* **18** (1980) 437-443.

[4] Balas, E., Duality in discrete programming, in H. W. Kuhn, ed., *Proceedings of the Princeton Symposium on Mathematical Programming*, Princeton University Press (1970) 179-197.

[5] Barth, P., *Logic-Based 0-1 Constraint Solving in Constraint Logic Programming*, Kluwer (Dordrecht, 1995).

[6] Bell, D. E., and J. F. Shapiro, A convergent duality theory for integer programming, *Operations Research* **25** (1977) 419-434.

[7] Benders, J. F., Partitioning procedures for solving mixed-variables programming problems, *Numerische Mathematik* **4** (1962) 238-252.

[8] Blair, C. E., and R. Jeroslow, The value function of a mixed integer program 1, *Discrete Mathematics* **19** (1977) 121-138.

[9] Blair, C. E., and R. Jeroslow, The value function of a mixed integer program, *Mathematical Programming* **23** (1982) 237-273.

[10] Chandru, V., C. R. Coullard, P. L. Hammer, M. Montañez, and X. Sun, On renamable Horn and generalized Horn functions, *Annals of Mathematics and AI* **1** (1990) 33-48.

[11] Chandru, V., and J. N. Hooker, Detecting embedded Horn structure in propositional logic, *Information Processing Letters* **42** (1992) 109-111.

[12] Chandru, V., and J. N. Hooker, *Optimization Methods for Logical Inference*, Wiley (New York, 1999).

[13] Cook, W., A. M. H. Gerards, A. Schrijver and E. Tardos, Sensitivity results in integer programming, *Mathematical Programming* **34** (1986) 251-264.

[14] Dawande, M., and J. N. Hooker, Inference-based sensitivity analysis for mixed integer/linear programming, to appear in *Operations Research*.

[15] Fisher, M. L., W. D. Northrup and J. F. Shapiro, Using duality to solve discrete optimization problems: Theory and computational experience, *Mathematical Programming Study* **3** (1975) 56-94.

[16] Fisher, M. L., and J. F. Shapiro, Constructive duality in integer programming, *SIAM Journal on Applied Mathematics* **27** (1974) 31-52.

[17] Geoffrion, A. M., Generalized Benders decomposition, *Journal of Optimization Theory and Applications* **10** (1972) 237-260.

[18] Harche, F., J. N. Hooker and G. L. Thompson, A computational study of satisfiability algorithms for propositional logic, *ORSA Journal on Computing* **6** (1994) 423-435.

[19] Hooker, J. N., Resolution vs. cutting plane solution of inference problems: Some computational experience, *Operations Research Letters* **7** (1988) 1-7.

[20] Hooker, J. N., Input proofs and rank one cutting planes, *ORSA Journal on Computing* **1** (1989) 137-145.

[21] Hooker, J. N., Generalized resolution for 0-1 linear inequalities, *Annals of Mathematics and Artificial Intelligence* **6** (1992) 271-286.

[22] Hooker, J. N., Solving the incremental satisfiability problem, *Journal of Logic Programming* **15** (1993) 177-186.

[23] Hooker, J. N., Logic-based methods for optimization, in A. Borning, ed., *Principles and Practice of Constraint Programming*, *Lecture Notes in Computer Science* **874** (1994) 336-349.

[24] Hooker, J. N., Needed: An empirical science of algorithms, *Operations Research* **42** (1994) 201-212.

[25] Hooker, J. N., Inference-based sensitivity analysis, *Constraints* **4** (1999) 104–112.

[26] Hooker, J. N., Testing heuristics: We have it all wrong, *Journal of Heuristics* **1** (1996) 33–42.

[27] Hooker, J. N., *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*, Wiley (2000).

[28] Hooker, J. N. and C. Fedjki, Branch-and-cut solution of inference problems in propositional logic, *Annals of Mathematics and AI* **1** (1990) 123-139.

[29] Hooker, J. N., G. Ottosson, E. S. Thorsteinsson and Hak-Jin Kim, *A scheme for unifying optimization and constraint satisfaction methods*, *Knowledge Engineering Review* **15** (2000) 11–30.

[30] Hooker, J. N., and Hong Yan, Logic circuit verification by Benders decomposition, in V. Saraswat and P. Van Hentenryck, eds., *Principles and Practice of Constraint Programming: The Newport Papers*, MIT Press (Cambridge, MA, 1995) 267-288.

[31] Jain, V., and I. E. Grossmann, Algorithms for hybrid MILP/CLP models for a class of optimization problems, to appear in *INFORMS Journal on Computing*.

[32] Jeroslow, R. G., R. K. Martin, R. L. Rardin and J. Wang, Gain-free Leontief flow problems, Graduate School of Business, University of Chicago, Chicago, IL USA (1989).

[33] Jeroslow, R. G. and J. Wang, Dynamic programming, integral polyhedra and Horn clause knowledge base, *ORSA Journal on Computing* **4** (1989) 7-19.

[34] Jiang, Y., T. Richards and B. Richards, No-good backmarking with min-conflict repair in constraint satisfaction and optimization, in A. Borning, ed., *Principles and Practice of Constraint Programming*, *Lecture Notes in Computer Science* **874** (1994) 21-39.

[35] Johnson, E., Cyclic groups, cutting planes and shortest paths, in T. C. Hu and S. Robinson, eds., *Mathematical Programming*, Academic Press (1973) 185-211.

[36] Mitterreiter, I., and F. J. Radermacher, Experiments on the running time behavior of some algorithms solving propositional logic problems, manuscript, Forschungsinstitut für anwendungsorientierte Wissensverarbeitung, Ulm, Germany, 1991

[37] Quine, W. V., The problem of simplifying truth functions, *American Mathematical Monthly* **59** (1952) 521-531.

[38] Quine, W. V., A way to simplify truth functions, *American Mathematical Monthly* **62** (1955) 627-631.

[39] Robinson, J. A., A machine-oriented logic based on the resolution principle, *Journal of the ACM* **12** (1965) 23-41.

[40] Shapiro, J. F., Generalized Lagrange multipliers in integer programming, *Operations Research* **19** (1971) 68-76.

[41] Schrage, L., and L. A. Wolsey, Sensitivity analysis for branch and bound integer programming, *Operations Research* **33** (1985) 1008-1023.

[42] Thorsteinsson, E. S., and J. N. Hooker, Solving vehicle routing problems with a hybrid of integer programming and constraint programming, manuscript, GSIA, Carnegie Mellon University, 2001.

[43] Tind, J., and L. A. Wolsey, An elementary survey of general duality theory in mathematical programming, *Mathematical Programming* **21** (1981)

[44] Trick, M., and D. S. Johnson, eds., *Second DIMACS Challenge: Cliques, Coloring and Satisfiability*, Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society (1995).

[45] Tsang, E., *Foundations of Constraint Satisfaction* (London, Academic Press) 1993.

[46] Van Hentenryck, P., *OPL Studio*.

[47] Williams, H. P., Duality in mathematics and linear and integer programming, *Journal of Optimization Theory and Applications* **90** (1996) 257-278.

[48] Wolsey, L. A., Integer programming duality: Price functions and sensitivity analysis, *Mathematical Programming* **20** (1981) 173-195.

[49] Wolsey, L. A., The b-hull of an integer program, *Discrete Applied Mathematics* **3** (1981) 193-201.

[50] Wolsey, L. A., A resource decomposition algorithm for general mathematical programs, *Mathematical Programming Study* **14** (1981) 244-257.