Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Combinatorial Benders' Cuts for Mixed-Integer Linear Programming

Gianni Codato, Matteo Fischetti,

To cite this article:
Gianni Codato, Matteo Fischetti, (2006) Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. Operations Research 54(4):756-766. https://doi.org/10.1287/opre.1060.0286

OPERATIONS RESEARCH
Vol. 54, No. 4, July–August 2006, pp. 756–766
ISSN 0030-364X | EISSN 1526-5463 | 06 | 5404 | 0756

DOI 10.1287/opre.1060.0286

Combinatorial Benders' Cuts for Mixed-Integer Linear Programming

Gianni Codato, Matteo Fischetti
Department of Information Engineering, University of Padova, via Gradenigo 6/A, I-35100 Padova, Italy
{giannicod@gmx.it, matteo.fischetti@unipd.it}

Mixed-integer programs (MIPs) involving logical implications modeled through big-M coefficients are notoriously among the hardest to solve. In this paper, we propose and analyze computationally an automatic problem reformulation of quite general applicability, aimed at removing the model dependency on the big-M coefficients. Our solution scheme defines a *master* integer linear problem (ILP) with no continuous variables, which contains combinatorial information on the feasible integer variable combinations that can be "distilled" from the original MIP model. The master solutions are sent to a *slave* linear program (LP), which validates them and possibly returns combinatorial inequalities to be added to the current master ILP. The inequalities are associated to minimal (or irreducible) infeasible subsystems of a certain linear system, and can be separated efficiently in case the master solution is integer. The overall solution mechanism closely resembles the Benders' one, but the cuts we produce are purely combinatorial and do not depend on the big-M values used in the MIP formulation. This produces an LP relaxation of the master problem which can be considerably tighter than the one associated with original MIP formulation. Computational results on two specific classes of hard-to-solve MIPs indicate that the new method produces a reformulation which can be solved some orders of magnitude faster than the original MIP model.

Subject classifications: mixed-integer programs; Benders' decomposition; branch and cut; computational analysis.
Area of review: Optimization.
History: Received March 2004; revision received February 2005; accepted June 2005.

1. Introduction

We first introduce the basic idea underlying combinatorial Benders' cuts; more elaborated versions will be discussed in the sequel.

Suppose that one has a basic 0-1 integer linear problem (ILP) of the form

$$\min\{c^T x\colon Fx \leqslant g,\ x \in \{0,1\}^n\}, \tag{1}$$

amended by a set of "conditional" linear constraints involving additional continuous variables y, of the form

$$x_{j(i)} = 1 \Rightarrow a_i^T y \geqslant b_i \quad \text{for all } i \in I, \tag{2}$$

plus a (possibly empty) set of "unconditional" constraints on the continuous variables y, namely,

$$Dy \geqslant e. \tag{3}$$

Note that the continuous variables y do not appear in the objective function—they are only introduced to force some feasibility properties of the xs.

A familiar example of a problem of this type is the classical asymmetric travelling salesman problem with time windows. Here, the binary variables x_{ij} are the usual arc variables, and the continuous variables y_i give the arrival time at city i. Implications (2) are of the form

$$x_{ij} = 1 \Rightarrow y_j \geqslant y_i + \mathit{travel\_time}(i,j), \tag{4}$$

whereas (3) bounds the arrival time at each city i,

$$\mathit{early\_arrival\_time}(i) \leqslant y_i \leqslant \mathit{late\_arrival\_time}(i). \tag{5}$$

Another example is the map-labeling problem (Klau and Mützel 2003), where the binary variables are associated to the relative position of two labels to be placed on a map, the continuous variables give their placement coordinates, and the conditional constraints impose nonoverlapping conditions of the type "if label i is placed on the right of label j, then the placement coordinates of i and j must obey a certain linear inequality giving a suitable separation condition."

The usual way implications (2) are modeled within the mixed-integer programming (MIP) framework is to use the (in)famous *big-M method*, where large positive coefficients M_i are introduced to activate/deactivate the conditional constraints as in

$$a_i^T y \geqslant b_i - M_i(1 - x_{j(i)}) \quad \text{for all } i \in I. \tag{6}$$

This yields a (often large) mixed-integer model involving both x and y variables, whereas, in principle, y variables are just artificial variables. Even more importantly, due to the presence of the big-M coefficients, the linear programming (LP) relaxation of the MIP model is typically very poor. As a matter of fact, the x solutions of the LP relaxation are only marginally affected by the addition of the y variables and of the associated constraints. In a sense, the

MIP solver is "carrying on its shoulders" the burden of *all* additional constraints and variables in (2)–(3) at *all* branch-decision nodes, while they become relevant only when the corresponding $x_{j(i)}$ attains value 1 (typically, because of branching).

Of course, we can get rid of the y variables by using Benders' decomposition (Benders 1962), but the resulting cuts are weak and still depend on the big-M values. As a matter of fact, the classical Benders' approach can be viewed as a tool to accelerate the solution of the LP relaxation, but not to improve its quality.

The idea behind "combinatorial" Benders' cuts is to work on the space of the x-variables only, as in the classical Benders' approach. However, we model the additional constraints (2)–(3) through the following *combinatorial Benders'* (CB) cuts:

$$\sum_{i\in C} x_{j(i)} \leqslant |C| - 1, \tag{7}$$

where $C \subseteq I$ induces a *minimal* (*or irreducible*) *infeasible subsystem* (MIS, or IIS, for short) of (2)–(3), i.e., any inclusion-minimal set of row indices of system (2) such that the linear subsystem

$$SLAVE(C) := \begin{cases} a_i^T y \geqslant b_i & \text{for all } i \in C, \\ Dy \geqslant e, \end{cases}$$

has no feasible (continuous) solution y.

A CB cut is violated by a given $x^* \in [0,1]^n$ if and only if $\sum_{i\in C}(1 - x^*_{j(i)}) < 1$. Hence, the corresponding separation problem essentially consists of the following steps: (i) weigh each conditional constraint $a_i^T y \leqslant b_i$ in (2) by $1 - x^*_{j(i)}$; (ii) weigh each unconditional constraint in (3) by 0; and (iii) look for a minimum-weight MIS of the resulting weighted system—an NP-hard problem (Amaldi et al. 2003, Gleeson and Ryan 1990).

A simple polynomial-time heuristic for CB-cut separation is as follows: Given the (fractional or integer) point x^* to be separated, start with $C := \{i \in I\colon\ x^*_{j(i)} = 1\}$, verify the infeasibility of the corresponding linear subsystem $SLAVE(C)$ by classical LP tools, and then make C inclusion minimal in a greedy way. Although extremely simple, this efficient separation turns out to be exact when x^* is integer.

The discussion above suggests the following exact branch-and-cut solution scheme. We work explicitly with the integer variables x only. At each branching node, the LP relaxation of a *master problem* (namely, problem (1) amended by the CB cuts generated so far) is solved, and the heuristic CB separation is called so as to generate new violated CB cuts (and to assert the feasibility of x^*, if integer).

The new approach automatically produces a sequence of CB cuts, which try to express in a purely combinatorial way the feasibility requirement in the x space—the CB cut generator acting as an automatic device to distill more and more combinatorial information from the input model. As a consequence of the interaction among the generated CB cuts, other classes of combinatorial cuts are likely to be violated, hence allowing other cut separators to obtain a further improvement. We found that the $\{0, 1/2\}$-cuts addressed in Caprara and Fischetti (1996) and Andreello et al. (2003) fit particularly well in this framework, and contribute substantially to the overall efficacy of the approach.

It is worth noting that using the new technique, the role of the big-M terms in the MIP model vanishes—only implications (2) are relevant, no matter the way they are modeled. Actually, the approach suggests an extension of the MIP modeling language where logical implications of type (2) can be stated explicitly in the model, as in Hooker and Osorio (1999).

In this paper, we aim at investigating whether the above method can be useful to approach certain types of MIPs which are notoriously very hard to solve. As shown in the computational section, this is indeed the case: even in its simplest implementation, on some classes of instances the new approach allows for a speed-up of some orders of magnitude with respect to ILOG-Cplex 8.1, one of the best MIP solvers on the market.

Our technique is based on Hooker's (2000) idea of deriving Benders' cuts from minimal sets of inconsistencies. In this respect, our main contributions have been (a) to present a separation heuristic that finds minimal Benders' cuts for the special case of conditional constraints with linear implications, and (b) to test these cuts computationally on some hard MIP problems. This is an important special case because conditional constraints of this form are a very useful modeling device.

This paper is organized as follows. In §2, we present the new approach in a more general context, whereas previous literature on the subject is reviewed in §3. As already stated, our CB cut separator requires a fast determination of MISs; this important topic is addressed in §4, where an approach particularly suited to our application is described. Computational results are presented in §5, with the aim of verifying whether a simple implementation of the new method can already produce improved performance with respect to the application of a sophisticated MIP solver such as ILOG-Cplex 8.1 (at least, on some problem classes which fit particularly well in our scheme). Finally, some conclusions are drawn in §6.

This paper is based on the master's thesis of the first author (Codato 2003), which was awarded the 2003 Camerini-Carraresi prize by the Italian Operations Research Association (AIRO). Also, this paper was presented at the Integer Programming and Combinatorial Optimization (IPCO) X meeting held in New York, June 2004.

2. Combinatorial Benders' Cuts

Let P be a MIP problem with the following structure:

$$P\colon \quad z^* := \min\ c^T x + d^T y \tag{8}$$

$$\text{s.t. } Fx \leqslant g, \tag{9}$$

$$Mx + Ay \geqslant b, \tag{10}$$

$$Dy \geqslant e, \tag{11}$$

$$x_j \in \{0, 1\} \quad \text{for } j \in B, \tag{12}$$

$$x_j \text{ integer} \quad \text{for } j \in G, \tag{13}$$

where x is a vector of integer variables, y is a vector of continuous variables, G and B are the (possibly empty) index sets of the general-integer and binary variables, respectively, and M is a matrix with exactly one nonzero element for every row i, namely, the one indexed by column $j(i) \in B$. In other words, we assume that the linking between the integer variables x and the continuous variables y is only due to a set of constraints of the type

$$m_{i,\,j(i)} x_{j(i)} + a_i^T y \geqslant b_i \quad \text{for all } i \in I, \tag{14}$$

where the variables $x_{j(i)}$ are binary for all $i \in I$.

We consider the case $d = 0$ first, i.e., we assume that the MIP objective function does not depend on the continuous variables, and leave case $d \neq 0$ for a later analysis. In this situation, we can split problem P into two subproblems:

- *MASTER*:

$$z^* = \min \quad c^T x \tag{15}$$

$$\text{s.t.} \quad Fx \leqslant g, \tag{16}$$

$$x_j \in \{0, 1\} \quad \text{for } j \in B, \tag{17}$$

$$x_j \text{ integer} \quad \text{for } j \in G. \tag{18}$$

- *SLAVE*($\tilde{x}$), a linear system parametrized by $\tilde{x}$:

$$Ay \geqslant b - M\tilde{x}, \tag{19}$$

$$Dy \geqslant e. \tag{20}$$

Let us solve the master problem at integrality. If this problem turns out to be infeasible, then P also is. Otherwise, let x^* be an optimal solution (we exclude the unbounded case here under the mild assumption that, e.g., the general-integer variables are bounded).

If the linear system *SLAVE*(x^*) has a solution, say y^*, then clearly (x^*, y^*) is an optimal solution of P. If the slave is infeasible, instead, x^* itself is infeasible for problem P. We therefore look for a MIS of *SLAVE*(x^*), involving the rows of A indexed by C (say), and observe that at least one binary variable $x_{j(i)}$ has to be changed to break the infeasibility. This condition can be translated by the following linear inequality in the x space, that we call the Combinatorial Benders' (CB) cut:

$$\sum_{i \in C:\, x^*_{j(i)} = 0} x_j + \sum_{i \in C:\, x^*_{j(i)} = 1} (1 - x_j) \geqslant 1. \tag{21}$$

One or more CB cuts of this type are generated in correspondence of a given infeasible x^*, and added to the master problem. Iterating the procedure produces an exact solution method in the spirit of Benders' decomposition.

Of course, it is advisable to exploit the CB cuts within a modern branch-and-cut scheme which hopefully produces violated cuts at each node of the branching tree—and not just in correspondence of an integer optimal solution of the master. Note that the correctness of the branch-and-cut method only requires the generation of a violated CB cut (if any) before each updating of the incumbent solution of the master, i.e., any heuristic CB separator that guarantees to be exact for an integer x^* already suffices to get a valid solution method.

We now address the case $c = 0$ and $d \neq 0$, arising when the objective function only depends on the continuous variables y. In this situation, we cannot accommodate the objective function into the master problem. Instead, we can add the bound inequality $d^T y \leqslant UB - \epsilon$ to the slave system, where UB is the value of the incumbent solution, and ϵ is a sufficiently small positive value. In this way, the CB cuts will translate both the feasibility and the optimality requirements. More specifically, at the iterations where *SLAVE*(x^*) (amended by the bound inequality) is infeasible, we can generate one or more violated CB cuts, as required. At the iterations where this system is feasible, instead, we can find an optimal solution y^* of the LP problem $\min\{d^T y\colon Ay \geqslant b - Mx^*,\ Dy \geqslant e\}$ and update the best incumbent solution by (x^*, y^*). The overall method will stop when the current master (that now looks for a feasible x improving the incumbent) becomes infeasible. Finally, we observe that the case $c \neq 0$ and $d \neq 0$ cannot be dealt with effectively by our method. A possible approach is to make an external binary search of the (unknown) value of $d^T y^*$ in an optimal solution (x^*, y^*) of P, and exploit this information by introducing bound constraints of the type $d^T y = d^T y^*$ into the slave. However, this naive approach would imply the solution of a (possibly long) series of MIPs, hence its practical effectiveness should be investigated computationally (this topic is left to future research).

At first glance, the required assumptions—in particular, (14)—restrict the range of applicability of our method considerably. However, there are several important (and difficult) MIPs which fit naturally in our scheme. Some of them will be addressed in the computational section. In addition, there is a simple reformulation method that can extend its applicability significantly. The idea is to introduce a (continuous) copy x_j^c of each binary variable x_j, $j \in B$, and to link the two copies through the constraints

$$x_j = x_j^c \quad \text{for } j \in B. \tag{22}$$

By construction, the above constraints can play the role of (14), thus linking the master problem to the slave.

In particular, one can always reformulate a generic MIP with no general-integer variables (i.e., involving only binary and continuous variables) as follows. Introduce the continuous copy x^c of the entire vector x. The initial master takes the binary variable vector x, plus the constraints

$Fx \leqslant g$, along with the obvious constraints $x \in \{0,1\}^n$; the master objective function is zero (or $c^T x$ if $d = 0$). The slave keeps the continuous variable vectors x^c and y, the constraints $Mx^c + Ay \geqslant b$, $Dy \geqslant e$, along with the linking equations $x = x^c$, plus the objective function bound $c^T x^c + d^T y \leqslant UB - \epsilon$, where UB is the value of the incumbent solution.

Actually, one can even decide to remove $Fx \leqslant g$ from the master, and to introduce $Fx^c \leqslant g$ into the slave (the master objective function being zero). With this choice, at each iteration the master only contains the distilled combinatorial information (notably, CB cuts) that excludes certain configurations of the binary variables because they are infeasible or cannot lead to an improved solution. The master then iteratively detects new binary solutions x^* which fulfill the master constraints generated so far, and invokes the slave for validation. The slave verifies the feasibility of the proposed x^* (with respect to the LP relaxation of the original MIP, amended by the upper bound constraint), possibly updates the incumbent, and then returns one or more CB cuts related to some forbidden *minimal* configurations of the binary variables.

3. Related Work

CB cuts have their roots in the seminal work of Hooker (2000) on logic-based methods for optimization, where the problem under investigation has the very general form

$$\min \quad f(x) \tag{23}$$

$$\text{s.t.} \quad p_i(x), \quad i \in I_1, \tag{24}$$

$$g_i(y), \quad i \in I_2, \tag{25}$$

$$q_i(x) \Rightarrow h_i(y), \quad i \in I_3. \tag{26}$$

If C is a minimal conflict set, in the sense that C is an inclusion-minimal subset of I_3 such that conditions $(h_i(y)\colon i \in C)$ and $(g_i(y)\colon i \in I_2)$ are inconsistent, then one can write the *Benders' cut condition*

$$\bigvee_{i \in C} \neg q_i(x). \tag{27}$$

In this context, our approach deals with the special case where conditions $p_i(x)$ are linear inequalities in the integer variables x_j, the $g_i(y)$ and $h_i(y)$ are linear inequalities, and the $q_i(x)$ have the form $x_{j(i)} = 1$. Indeed, in this case, condition (27) can be rephrased as the CB cut $\sum_{i \in C}(1 - x_{j(i)}) \geqslant 1$.

Apparently, the first practical use of this kind of Benders' cuts was to solve circuit verification problems (Hooker and Yan 1995). In this application, the subproblem constraints $h_i(y)$ are again linear inequalities (actually, systems of linear inequalities), but they have a special structure that permits the rapid identification of minimal conflict sets. Hooker (2000) studied logic-based Benders' cuts and proposed applying them to multiple machine scheduling, in which constraints $h_i(y)$ are scheduling constraints rather than linear inequalities. Jain and Grossmann (2001) implemented this idea and obtained very good results, even without insisting on finding minimal conflict sets. Similar results were obtained for a broader class of problems in Hooker (2004), but these results required a more sophisticated form of Benders' cut than (27). Cambazard et al. (2004) returned to the idea of min-conflict Benders' cuts to solve real-time scheduling problems on multiple machines. They relied on a min-conflict algorithm of Junker (2001), which improves on an earlier algorithm of De Siqueira and Puget (1988).

Kim and Hooker (2002) considered fixed-charge network flow problems, and proposed a scheme that is very similar to the approach proposed in this paper. The main difference is that Kim and Hooker find minimal infeasible configurations of the binary variables (the so-called *nogoods*) specifically in a min-cost network flow relaxation, while this paper finds them in a more general context. The same comment applies to the map-labeling problem studied by Klau and Mützel (2003); see §5 for details.

Rather than alternating between solving a master problem and subproblem, as in the classical Benders' method, our approach solves a single master problem and generates Benders' cuts on the fly. This approach was proposed in Hooker (2000), and later named "branch and check" by Thorsteinsson (2001), who successfully applied it to the Jain and Grossmann (2001) problems. Meanwhile, Hooker and Ottosson (2003) used logic Benders' cuts to solve SAT problems and 0-1 programming problems.

Our solution method also has interesting connections with Chvátal's (1997) *resolution search*. Roughly speaking, resolution search can be viewed as an attempt to get rid of the rigid tree paradigm used within enumeration schemes. Convergence of generic enumerative methods for combinatorial optimization problems requires one to record information about the subsets of the solution space that have been already explored, so as to avoid considering a same subset twice, and to abort the search when no unexplored subset exists.

For each subset, an *oracle* (borrowing Chvátal's terminology) is invoked to answer the basic question: Is there any hope that this subset contains a solution improving the incumbent? If (possibly after having updated the incumbent) the answer is "no," the subset is discarded. If the answer is "yes," the set is subdivided further. For MIP problems, the subsets are typically defined by a *branching strategy*, i.e., by fixing the value of certain subsets of the integer variables. Resolution search can be viewed as a way to control the solution process. Here, the standard branching tree is replaced by a pool of logical conditions that summarize the previous computation. At each iteration, a partial assignment of the binary variables is found, which fulfills the logical conditions in the current pool. If no such assignment exists, then the enumeration is over and the incumbent is a provable optimal solution. Otherwise, the oracle

is invoked in two distinct greedy phases. In the *waxing phase*, the current partial assignment is iteratively extended up to a point where the oracle returns a "no" answer (with the incumbent possibly updated). In the subsequent *waning phase*, the current partial assignment is iteratively reduced as long as the oracle answer remains "no." At the end of the waning phase, the current partial assignment corresponds to an *obstacle*, i.e., to a minimal set of variables that have to be changed to get a "yes" answer from the oracle. The logical condition "change the value at least one of the variables in the obstacle" is then added to the pool of logical conditions, so as to avoid facing the same obstacle in the later exploration, and the process is iterated. The analogy with our approach is now evident: our master problem plays the role of a 0-1 ILP model for the logical conditions in the pool, whereas the slave is our implementation of the oracle function to detect one or more obstacles efficiently.

There is, however, an important difference between our method and resolution search. Indeed, in resolution search the nogoods are structured so that one can find a feasible solution for the accumulated nogoods in polynomial time. In Chvátal's method, in particular, the nogoods have a proper pathlike structure; other variations on this theme include dynamic backtracking, dependency-directed backtracking, partial-order dynamic backtracking, and generalized partial-order dynamic backtracking (Bliek 1998, Ginsberg 1993, Ginsberg et al. 1996, Ginsberg and Mc Allester 1994, Hooker 2000). This nice feature is not preserved in our method, where the master problem is a (somehow general) NP-hard ILP.

CB cuts can be generated within a standard branch-and-cut solution framework not based explicitly on the master/slave decomposition we propose. To simplify notation, let us consider a pure 0-1 ILP of the form $\min\{c^T x\colon Ax \geqslant b, x \in \{0,1\}^n\}$. Take a generic branching node, $NODE_h$, which is fathomed by the classical lower bound criterion, possibly after the updating of the incumbent solution. The node corresponds (say) to the fixing $x_j = x_j^*$ for some $j \in J_h$, where $x_j^* \in \{0,1\}$ is the known branching value for variable x_j at $NODE_h$. The fathoming condition then implies that the slave linear system made by the "linking equations" $x_j = x_j^*$ for $j \in J_h$ and by the inequalities $Ax \geqslant b$, $x \in [0,1]^n$, $c^T x \leqslant UB - \epsilon$, is infeasible. Finding a MIS of this system then yields an inclusion-minimal $C \subseteq J_h$, whose associate CB cut (21) can be added to the current formulation. (Of course, in this context the new cut is only useful if C is a proper subset of J_h, so as to exclude a solution subset strictly larger than the one associated with the current node $NODE_h$.) Our approach can therefore be viewed as a simple (yet hopefully effective) method for "branching resequencing" in the spirit of the *dynamic branch and bound* of Glover and Tangedhal (1976).

4. Fast MIS Search

In this section, we describe an efficient algorithm to find a MIS of an infeasible linear system that fits particularly well within our solution approach. The method is in the spirit of the one discussed by Parker and Ryan (1996).

MIS search can be formulated as follows: given an infeasible system of inequalities, say $\tilde{A}y \geqslant \tilde{b}$, find an inclusion-minimal set of its rows yielding an infeasible system.

We therefore construct the LP

$$\min \quad 0^T y \tag{28}$$

$$\text{s.t.} \quad \tilde{A}y \geqslant \tilde{b}, \tag{29}$$

and its dual

$$\max \quad u^T \tilde{b} \tag{30}$$

$$\text{s.t.} \quad u^T \tilde{A} = 0^T, \tag{31}$$

$$u \geqslant 0. \tag{32}$$

It is known that if the primal problem is infeasible, then the corresponding dual can be either unbounded or infeasible. Now, dual infeasibility is excluded by the fact that $u = 0$ is always dual feasible, hence primal infeasibility corresponds to dual unboundedness, i.e., to the existence of a dual solution u^* such that $u^{*T}\tilde{b} > 0$ (hence, ku^* for a sufficiently large $k > 0$ is a feasible dual solution with an arbitrarily large objective value). Therefore, we can replace the dual objective function by the following constraint:

$$u^T \tilde{b} = 1. \tag{33}$$

This modified dual problem is directed at finding a linear combination of the rows of $\tilde{A}y \geqslant \tilde{b}$ leading to a valid inequality $u^{*T}\tilde{A}y \geqslant u^{*T}b$ with all-zero left-hand-side coefficients and strictly positive right-hand side, thus proving the infeasibility of $\tilde{A}y \geqslant \tilde{b}$—the existence of u^* is guaranteed for infeasible systems by Farkas' lemma.

It is known (Gleeson and Ryan 1990) that each *vertex* of the dual polyhedron defined by (31)–(33) has a support defining a MIS (whereas minimality is not guaranteed for an arbitrary point u of the same polyhedron). In our applications, we look for a solution u^* having a small support in the set of the linking constraints (19), whereas we do not actually care on the minimality with respect to (20). This suggests the use of the heuristic dual objective function $\sum_i w_i u_i$, where weights w_is are used to drive the LP solver to select a solution u^* with the desired characteristics. Moreover, by iteratively setting to zero some of the u variables, we can easily detect alternative MISs, which is very important to generate several violated CB cuts at each call of the separation heuristic.

According to our computational experience, the described algorithm is very effective and outperforms (for our applications) an analogous MIS algorithm based on the ILOG-Cplex function `IloCplex::getIIS()`.

5. Computational Results

To evaluate its effectiveness, we implemented our method in C++ and embedded it within the ILOG-Cplex Concert

Technology 1.2 (2003) framework, based on ILOG-Cplex 8.1 (2003).

In our implementation, we read an input MIP in the form (8)–(13), and verify that (a) the linking constraints are of the form (14) with binary $x_{j(i)}$, and (b) the objective function only depends on the integer variables. We then construct (automatically) the master/slave decomposition, and invoke the ILOG-Cplex solver on the master. During the ILOG-Cplex branch-and-cut execution, we call our separation procedures for CB and $\{0, 1/2\}$-cuts. In the CB separation, the integer components of the current master LP solutions x^* are sent to the slave, in the attempt to generate one or more CB cuts through the MIS search described in §3. As to $\{0, 1/2\}$-cuts, we use the separation code of Andreello et al. (2003). This separation proved quite effective, and allowed for a reduction of up to 50% of the computing time of our method for some instances of the test bed. (Observe that effective $\{0, 1/2\}$-cuts cannot be derived from the original MIP formulation of the instances in our test bed, due to the presence of the continuous variables.)

In order not to overload the LP formulation, we avoided calling our CB and $\{0, 1/2\}$-cut separators at each node of the branching tree, but applied them (a) before each updating of the incumbent solution; (b) at each node with a tree depth not greater than 10; and (c) after each backtracking step (see Andreello et al. 2003 for a discussion on similar strategies).

Due to the heuristic nature of our separation procedures for CB and $\{0, 1/2\}$-cuts, and because the number of generated cuts tends to increase steeply, all cuts are stored in a constraint pool, which is purged from time to time.

It should be observed that ILOG-Cplex 8.1 does not implement an internal cut pool: once a (globally-valid) cut has been generated at run time, it is added statically to the LP formulation and never removed. An important exception (that we exploit in our code) arises for locally-valid cuts, which are automatically removed from the formulation the first time they are no longer guaranteed to be valid (because of a backtracking step); removed cuts are not saved.

The lack of a native pool within ILOG-Cplex 8.1 is not really an issue in itself, in that one can easily implement an external data structure to store the CB and $\{0, 1/2\}$-cuts. A possible issue is, however, the impossibility of removing a (globally-valid) cut from the current LP. This makes it impossible to purge the current LP by removing some of its constraints, with the risk it becomes too large. A possible work-around is to define the CB and $\{0, 1/2\}$-cuts as *local* (as opposed to global) cuts, so they are automatically removed from time to time from the LP. Although this approach does not allow us to have complete control on the cut removal mechanism (nor on the size of the LP), it works reasonably well in practice but implies a certain memory overhead.

5.1. The Test Bed

Realistically, one cannot expect our approach to work well in all applications. Indeed, CB cuts can sometimes be very weak, as they do not capture in an adequate way the complexity of the problem at hand. In particular, this is the case for the asymmetric travelling salesman problem with time windows (the problem we used as an introductory example), where CB cuts can only state the single infeasibility of certain paths and are dominated by much stronger classes of problem-specific inequalities, including the *tournament inequalities* studied in Ascheuer et al. (2000, 2001). However, our method proved to have some merits in handling difficult MIP problems where CB cuts play a role in describing in a strong polyhedral way the underlying combinatorial structure. Examples of these problems are described next.

Map Labeling. This problem consists of placing as many rectangular labels as possible (without overlap) in a given map. If placed, a label has a limited degree of movement around its associated "reference point" (a prespecified point of the map corresponding to, e.g., the city whose name is in the label). This problem has been formulated as a MIP model involving big-M coefficients by Klau and Mütze1 (2003), who report very poor results when trying to solve the model directly. Much better results are in fact obtained by the same authors when using a different (purely combinatorial) 0-1 ILP model, where the binary variables are associated with arcs of a suitable digraph, and the nonoverlapping conditions are translated by rank inequalities forbidding the choice of all the arcs in a circuit with certain properties. Actually, it was precisely the nice correspondence between the MIP model and its graph reformulation that motivated us to generalize the Klau-Mütze1 construction, thus originating the work presented in this paper. As a matter of fact, applying our method to their MIP model automatically produces a slave problem with a network structure, hence it can be associated with a digraph whose circuits (under appropriate conditions) correspond to minimal infeasible subsystems, and hence to CB cuts.

Our test bed contains 18 map-labeling instances of the so-called 4-slider (4S) and 4-position (4P) type, kindly provided by G. W. Klau.

As shown in the computational section, the results we obtained on map-labeling problems are comparable with those reported in Klau and Mütze1 (2003), even though our method is more general and does not exploit the specific network structure of the slave. This is an indication that the overhead introduced in our implementation (in particular, for computing MISs via LP methods rather than using fast graph-search algorithms) is acceptable.

Two-Group Statistical Classification (Discriminant Analysis). This problem can be described briefly as follow; see, e.g., Rubin (1997) for more details. We have a population whose members can be divided into two distinct classes, for example, people affected or not by a certain disease. We can measure a number of characteristics that are related to the partition above, e.g, biological parameters that show the presence or absence of

the disease. Based on the available measures, we want to obtain a linear function which allows us to decide whether a new member belongs to the first or second class. More specifically, we are looking for a linear function that minimizes the number of misclassified members in the known sample. (Designing optimal linear classifiers consists of minimizing (respectively, maximizing) the number of misclassified (respectively, correctly classified) members. Given the infeasible inequality system imposing correct classification of all members, the problem amounts to deleting the minimum number of inequalities to make the system feasible, or equivalently, to finding a maximum feasible subsystem (MaxFS); see, e.g., Chinneck (2001) and Amaldi et al. (2003).) This problem can be modeled as a MIP problem with big-M coefficients: the unknowns are the coefficients of the linear function, and for every member there is a binary variable used to deactivate the inequality in case of misclassification. The purpose is to minimize the number of deactivations, so the objective function is just the sum of the binary variables. (In Rubin (1997) a slightly different MIP model is used, involving real-valued variables in the objective; this model can easily be transformed into the one we use in this paper.)

The raw data from which we have generated our instances has been taken from a public archive maintained at the University of California, Irvine (Murphy and Aha 1994), and converted to the final MIP model with a slightly modified version of a program kindly provided to us by P. A. Rubin. This resulted in 38 instances coming from different real situations (i.e., disease classification, interpretation of physical phenomena, animal behavior, etc.).

5.2. Results

Our experiments have been performed on a PC AMD Athlon 2100+ with 1 GByte RAM, and GNU/Linux (kernel 2.4) operating system.

All the instances of our test bed have been processed twice: the first time by solving the original MIP model through the commercial ILOG-Cplex 8.1 solver (with default settings), and the second by using our implementation of the master/slave decomposition based on CB cuts and $\{0, 1/2\}$-cuts (still based on the ILOG-Cplex 8.1 library). A time limit of three CPU hours was imposed for

Table 1. Problems solved to proven optimality by both Cplex and CBC.

		Subset 1				
		Execution times			Nodes	
File name	Opt.	Cplex h:m:s	CBC h:m:s	Ratio Cplex/CBC	Cplex	CBC
Statistical analysis						
Chorales-116	24	1:24:52	0:10:18	8.2	10,329,312	20,382
Balloons76	10	0:00:10	0:00:14	71.4	40,481	4
BCW-367	8	0:08:33	0:00:13	39.4	79,980	463
BCW-683	10	2:02:29	0:00:32	229.7	399,304	671
WPBC-194	5	0:57:17	0:03:32	16.2	806,188	26,439
Breast-Cancer-400	6	0:02:50	0:00:16	1,062	181,990	1
Glass-163	13	0:56:17	0:00:05	675.4	3,412,702	64
Horse-colic-151	5	0:04:50	0:00:23	12.6	135,018	2,184
Iris-150	18	0:09:29	0:01:10	8.1	970,659	1,290
Credit-300	8	0:19:35	0:00:02	587.5	176,956	66
Lymphography-142	5	0:00:11	0:00:01	11.0	8,157	106
Mech-analysis-107	7	0:00:05	0:00:01	5.0	11,101	68
Mech-analysis-137	18	0:07:44	0:00:27	17.2	938,088	1,888
Monks-tr-122	13	0:02:05	0:00:05	25.0	262,431	357
Pb-gr-txt-198	11	0:04:21	0:00:05	52.2	135,980	110
Pb-pict-txt-444	7	0:02:07	0:00:02	63.5	71,031	1,026
Pb-hl-pict-277	10	0:04:17	0:00:27	9.5	22,047	115
Postoperative-88	16	0:15:16	0:00:01	916.0	2,282,109	171
BV-OS-282	6	0:05:13	0:00:24	13.0	56,652	1,044
Opel-Saab-80	6	0:01:03	0:00:13	4.8	87,542	7,314
Bus-Van-437	6	0:09:17	0:00:28	19.9	55,224	6,795
HouseVotes84-435	6	0:04:59	0:00:11	27.2	42,928	734
Water-treat-206	4	0:01:10	0:00:06	11.7	12,860	482
Water-treat-213	5	0:17:00	0:00:51	20.0	168,656	4,036
Map labeling						
CMS 600-1	600	1:08:41	0:04:34	15.0	110,138	14
Total/Mean	—	8:29:51	0:24:11	21.1	20,797,534	64,373

Table 2. Problems solved to proven optimality by CBC but not by Cplex.

	Subset 2						
	Cplex					CBC	
File name	Time h:m:s	Best solution	Best bound	Gap (%)	Mem. MB	Opt.	Time h:m:s
Statistical analysis							
Chorales-134	0:36:23	33	16.0	51	371	30	0:36:23
	3:00:58	30	21.1	30	992		
Chorales-107	0:04:19	28	12.1	57	61	27	0:04:19
	3:01:27	27	22.2	18	711		
Breast-Cancer-600	0:00:13	108	1.5	99	9	16	0:00:13
	3:00:11	16	13.2	18	45		
Bridges-132	0:03:39	33	5.1	85	44	23	0:03:39
	3:01:09	23	10.0	56	1,406		
Mech-analysis-152	0:34:12	22	12.1	45	328	21	0:34:12
	3:00:50	21	16.1	24	865		
Monks-tr-124	0:01:55	27	8.1	70	25	24	0:01:55
	3:00:35	24	20.0	17	381		
Monks-tr-115	0:04:16	29	9.1	69	67	27	0:04:16
	3:01:07	27	19.0	30	1,131		
Solar-flare-323	0:00:04	51	5.0	90	18	38	0:00:04
	3:00:45	43	17.0	61	977		
BV-OS-376	0:09:04	9	3.1	65	9	9	0:09:04
	3:00:10	9	6.0	33	56		
BusVan-445	0:10:31	13	3.0	77	11	8	0:10:31
	3:00:06	9	5.1	43	56		
Total/Mean	30:07:18	Gaps: same t. = 71%, end = 33%					1:44:36
Map labeling							
CMS 600-0 (4S)	0:04:27	592	600	1.35	18	600	0:04:27
	3:03:00	594	600	1.01	770		
CMS 650-0 (4S)	0:06:26	638	650	1.88	20	649	0:06:26
	3:02:34	646	650	0.62	480		
CMS 650-1 (4S)	0:04:50	647	650	0.46	7	649	0:04:50
	3:03:13	648	650	0.31	904		
CMS 700-1 (4S)	0:13:06	686	700	2.04	58	699	0:13:06
	3:03:00	691	700	1.30	1,045		
CMS 750-1 (4S)	0:07:53	738	750	1.63	28	750	0:07:53
	3:02:19	741	750	1.21	521		
CMS 750-4 (4S)	0:07:05	736	750	1.90	28	748	0:07:05
	3:00:24	743	750	0.94	417		
CMS 800-0 (4S)	0:19:15	773	800	3.49	55	798	0:19:15
	3:02:16	773	800	3.49	533		
CMS 800-1 (4S)	0:22:24	784	800	2.04	92	800	0:22:24
	3:02:30	786	800	1.78	761		
Railway	0:00:31	95	103	8.42	1	100	0:00:31
	3:00:02	100	101	1.00	19		
CMS 600-0 (4P)	0:00:01	543	600	10.5	2	600	0:00:04
	3:02:57	574	600	4.53	782		
CMS 600-1 (4P)	0:39:07	565	600	6.19	184	597	0:39:07
	3:02:55	568	600	5.63	831		
Totals	33:25:10	Gaps: same t. = 3.6%, end = 2.0%					2:05:4.1

each run. In addition, we set the `IloCplex::NodeFileInd` parameter to 3: this forces Cplex to store the active-node data into the hard disk when physical memory is exhausted (due to the efficiency of this mechanism, we had no memory problems even when handling branching trees much larger than 1 GB).

In Tables 1–3, CBC refers to our code (based on CB cuts), whereas Cplex refers to the standard ILOG-Cplex 8.1 solver. We compared the two codes in terms of execution times and/or integrality gaps computed with respect to the best integer solution found, on the following three instance subclasses:

1. Instances solved to proven optimality by both CBC and Cplex.

2. Instances solved to proven optimality by CBC but not by Cplex.

3. Instances not solved (to proven optimality) by CBC nor by Cplex.

Note that there was no instance in our test bed that was solved by Cplex but not by CBC.

For the first subset, Table 1 reports the instance names, the optimal objective values (Opt), the Cplex and CBC computing times, their ratios, and the number of branching nodes enumerated by Cplex and by CBC for the exact solution of the instances. On these instances, CBC turns out to be up to three orders of magnitude faster than Cplex. According to the final row of the table, Cplex ran for almost 8.5 hours to solve all the instances of this subset, while CBC requires about 24 minutes, i.e., the latter code was 21 times faster in processing the whole subset. As to the number of nodes enumerated by the two approaches, we note that a direct comparison of these figures is not immediate in that the two methods actually work on different formulations, hence processing a single node may require different computing times. In particular, for CBC the large number of generated cuts tends to slow down the node processing significantly.

Figure 1 shows the graph of the best bound and of the incumbent solution values for Cplex and CBC executions on a sample instance in this subset.

The results for the second subset of instances are given in Table 2. As in Table 1, for CBC we report the optimization time and the optimal objective value, while for Cplex we give the value of the incumbent solution (Best solution) and of the best bound (Best bound), their percentage gap (Gap (%)), computed as $100|\text{best\_sol} - \text{best\_bound}|/\text{best\_sol}$), as well as the MBytes of memory used (Mem. MB). For each instance, we report two distinct rows for Cplex execution: the first row refers to the time instant when CBC finished its computation, and the second row gives the situation when Cplex is aborted (time limit expired). Finally, in the last row of the two sections of the table we report the total computing time and the average Cplex gaps, computed both

Figure 1. Optimizing the statistical analysis instance Chorales-116 (minimization problem).

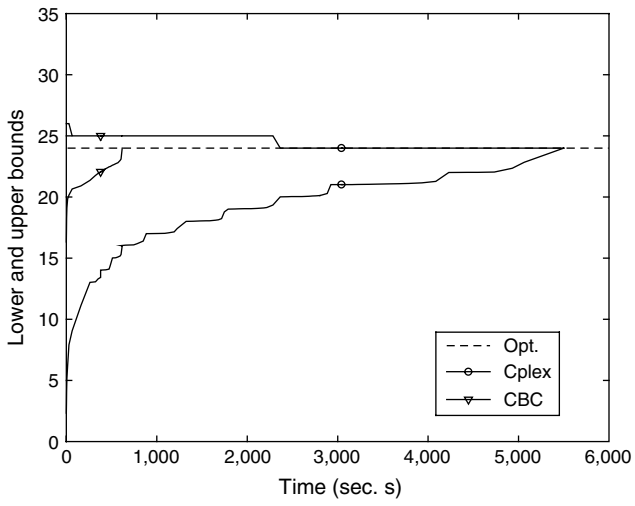
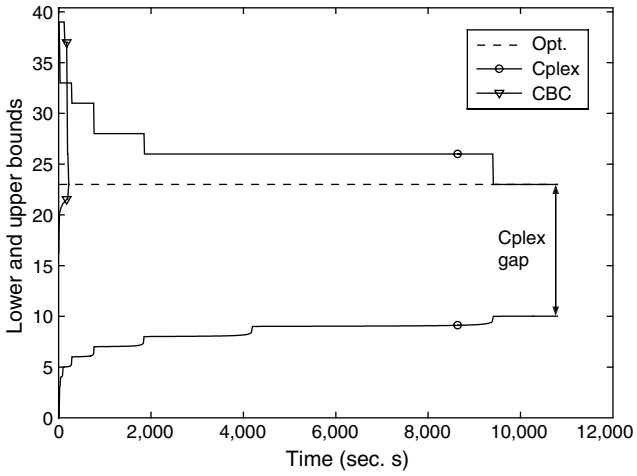


Figure 2. Optimizing the statistical analysis instance Bridges-132 (minimization problem).



when CBC finished its execution (same t.), and at the end of its own computation (end).

Figure 2 gives an illustration of the Cplex and CBC behavior on two sample instances of this second subset.

For this class, the difference between CBC and Cplex is even more evident: not only was Cplex not able to solve any of these problems within the imposed time limit, but the best-bound trend suggests that it is very unlikely it will converge even allowing for an extremely large computational effort. On the whole, CBC solved all the statistical analysis instances in less than two hours while, after the same time, Cplex had an average gap of more than 70%. Moreover, we found that after 28 additional hours of computation, the gap would have been only halved. An analogous behavior was observed on map-labeling instances, for which the gap is not even halved after more than 31 additional hours of computation.

The results for the third subset are summarized in Table 3, where we report both for Cplex and CBC the same information given for Cplex in Table 2, excluding only the time column (as all instances reached the three-hour time limit). In addition, in the last column of the table we report the difference between the Cplex and CBC gaps (ΔGap); in all cases, CBC obtained significantly smaller gaps.

6. Conclusions

We have proposed and analyzed computationally an automatic MIP reformulation method, aimed at removing the model dependency on the big-M coefficients often used to model logical implications.

Our technique is based on Hooker's (2000) idea of deriving Benders' cuts from minimal sets of inconsistencies. In this respect, our main contributions have been (a) to present a separation heuristic that finds minimal Benders' cuts for the special case of conditional constraints with linear implications, and (b) to test these cuts computationally

Table 3. Problems solved to proven optimality by neither code.

	Subset 3								
	Cplex				CBC				
File name	Best solution	Best bound	Gap (%)	Mem. MB	Best solution	Best bound	Gap (%)	Mem. MB	ΔGap (%)
Statistical analysis									
Flags-169	10	5.0	49.8	290	10	6.50	35.0	4,052	14.8
Horse-colic-253	13	5.0	61.5	279	13	8.91	31.5	3,394	30.0
Horse-colic-185	11	5.0	54.4	265	12	6.33	47.3	4,494	7.1
Solar-flare-1066	273	7.6	97.3	787	284	201.30	29.1	1,423	68.2
Total/Mean		—	65.5	1,621		—	35.7	13,363	30.0
Map labeling									
Berlin	37	47.8	29.1	1,063	38	43.0	13.1	1,952	16.0
CMS 900-0 (4S)	881	900	2.2	676	897	898.5	0.2	283	2.0
CMS 1000-0 (4S)	945	1,000	5.8	566	978	998.3	2.1	509	3.7
US-Abbrv	73	104.8	43.6	740	77	99.7	29.5	428	14.1
CMS 650-0 (4P)	611	650	6.4	764	633	646.9	2.2	1,658	4.2
CMS 650-1 (4P)	604	650	7.6	798	638	648.0	1.6	706	6.0
Total/Mean		—	15.8	3,261		—	8.12	5, 536	7.7

on some hard MIP problems. This is an important special case because conditional constraints of this form are a very useful modeling device.

Our method appears particularly suited for the MIPs whose objective function only depends on the integer variables, and the continuous variables are linked to the integer ones through linear constraints involving a single binary variable each typically multiplied by a large coefficient. This is precisely the situation arising in many important combinatorial problems modeled by the big-M technique, where the continuous variables are only used to certify a certain property of the feasible solutions (e.g., time windows in scheduling problems, piece nonoverlapping in nesting problems, etc.).

Computational results on two specific classes of hard-to-solve MIPs (namely, *statistical analysis* and *map-labeling problems*) show that the new method automatically produces a reformulation which can be solved some orders of magnitude faster than the original MIP model.

Future direction of work should address the more general case where the MIP objective function depends on both the continuous and the integer variables, and analyze computationally the merits of the resulting technique. Some preliminary results in this direction are encouraging.

Finally, one could ask whether it is indeed convenient to use our approach as a decomposition method, rather than simply as a cut-generation strategy. To this end, one could simply leave the continuous variables (and the associated constraints) in the master problem, and use the generated CB cuts as cutting planes. According to our computational experience, however, this approach has the main drawback of producing fractional points that are very difficult to separate by means of our simple CB-cut separation heuristics. Instead, preliminary computational tests seem to indicate that for some problems, a better strategy would be to use our decomposition approach as a preprocessing tool. To be more specific, one could first run our decomposition approach for a while, so as to generate a large number of globally-valid CB cuts, and then restart the optimization of the original model with these cuts used to initialize the internal cut pool. The applicability of this two-step approach is left for future investigation.

Acknowledgments

This work was supported by MIUR and CNR (Italy), by the EU project ADONET, and by the Future and Emerging Technologies unit of the EC (IST priority), under contract FP6-021235-2 (project Arrival). The authors thank two anonymous referees whose suggestions helped the authors in setting their contribution in the appropriate constraint programming context.

References

Amaldi, E., M. E. Pfetsch, L. E. Trotter. 2003. On the maximum feasible subsystem problem, IISs and IIS-hypergraphs. *Math. Programming* **95**(3) 533–554.

Andreello, G., A. Caprara, M. Fischetti. 2003. Embedding cuts within a branch and cut framework: A computational study with {0, 1/2}-cuts. Technical report, DEI, University of Padova, Padova, Italy.

Ascheuer, N., M. Fischetti, M. Groetschel. 2000. A polyhedral study of the asymmetric travelling salesman problem with time windows. *Networks* **36**(2) 69–79.

Ascheuer, N., M. Fischetti, M. Groetschel. 2001. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Math. Programming* **90** 475–506.

Benders, J. F. 1962. Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik* **4** 238–252.

Bliek, C. 1998. Generalizing partial order and dynamic backtracking. *Proc. 15th National Conf. on Artificial Intelligence* (*AAAI-98*). AAAI Press/MIT Press, Boston, MA, 319–325.

Cambazard, H., P.-E. Hladik, A.-M. Déplanche, N. Jussien, Y. Trinquet. 2004. Decomposition and learning for a hard real time task alloca-

tion algorithm. M. Wallace, ed. *Principles and Practice of Constraint Programming* (*CP 2004*). Lecture Notes in Computer Science 3258. Springer, Berlin, Germany, 153–158.

Caprara, A., M. Fischetti. 1996. $\{0, 1/2\}$-Chvátal-Gomory cuts. *Math. Programming* **74** 221–235.

Chinneck, J. 2001. Fast heuristics for the maximum feasible subsystem problem. *INFORMS J. Comput.* **13**(3) 210–223.

Chvátal, V. 1997. Resolution search. *Discrete Appl. Math.* **73** 81–99.

Codato, G. 2003. A combinatorial approach to Benders' decomposition. Master's dissertation, University of Padova, Padova, Italy (in Italian).

De Siqueira N., J. L., J.-F. Puget. 1988. Explanation-based generalisation of failures. *Proc. Eur. Conf. on Artificial Intelligence* (*ECAI-88*). Pitman Publishers, Munich, Germany, 339–344.

Ginsberg, M. L. 1993. Dynamic backtracking. *J. Artificial Intelligence Res.* **1** 25–46.

Ginsberg, M. L., D. A. McAllester, 1994. GSAT and dynamic backtracking. *Proc. Second Workshop on Principles and Practice of Constraint Programming*. Springer, Berlin, Germany, 216–225.

Ginsberg, M. L., J. M. Crawford, D. W. Etherington. 1996. Dynamic backtracking. Technical report, CIRL, University of Oregon, Eugene, OR.

Gleeson, J., J. Ryan. 1990. Identifying minimally infeasible subsystems of inequalities. *ORSA J. Comput.* **2**(1) 61–63.

Glover, F., L. Tangedhal. 1976. Dynamic strategies for branch and bound. *Omega* **4**(5) 571–576.

Hooker, J. N. 2000. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. John Wiley and Sons, New York.

Hooker, J. N. 2004. A hybrid method for planning and scheduling. M. Wallace, ed. *Principles and Practice of Constraint Programming* (*CP 2004*). Lecture Notes in Computer Science 3258. Springer, Berlin, Germany, 305–316.

Hooker, J. N., M. A. Osorio. 1999. Mixed logical/linear programming. *Discrete Appl. Math.* 96–97, 395–442.

Hooker, J. N., G. Ottosson. 2003. Logic-based Benders decomposition. *Math. Programming* **96** 33–60.

Hooker, J. N., H. Yan. 1995. Logic circuit verification by Benders decomposition. V. Saraswat, P. Van Hentenryck, eds. *Principles and Practice of Constraint Programming: The Newport Papers*. MIT Press, Cambridge, MA, 267–288.

ILOG Cplex 8.1. 2003. *User's Manual and Reference Manual, ILOG, S.A.* http://www.ilog.com/.

ILOG-Cplex Concert Technology 1.2. 2003. *User's Manual and Reference Manual, ILOG, S.A.* http://www.ilog.com/.

Jain, V., I. E. Grossmann. 2001. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS J. Comput.* **13** 258–276.

Junker, U. 2001. Quickxplain, conflict detection for arbitrary constraint propagation algorithms. IJCAI'01 Workshop on Modeling and Solving Problems with Constraints (CONS-1), Seattle, WA.

Kim, H.-J., J. N. Hooker. 2002. Solving fixed-charge network flow problems with a hybrid optimization and constraint programming approach. *Ann. Oper. Res.* **115**(1–4) 95–124.

Klau, G. W., P. Mütze. 2003. Optimal labelling of point features in rectangular labelling models. *Math. Programming Ser. B* **94**(2–3) 435–458.

Murphy, P. M., D. W. Aha. 1994. UCI repository of machine learning databases. Department of Information and Computer Science, University of California, Irvine, CA. http://www.ics.uci.edu/˜mlearn/MLRepository.html.

Parker, M., J. Ryan. 1996. Finding the minimum weight IIS cover of an infeasible system of linear inequalities. *Ann. Math. Artificial Intelligence* **17** 107–126.

Rubin, P. A. 1997. Solving mixed integer classification problem by decomposition. *Ann. Oper. Res.* **74** 51–64.

Thorsteinsson, E. S. 2001. Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. T. Walsh, ed. *Principles and Practice of Constraint Programming* (*CP 2001*). Lecture Notes in Computer Science. Springer, Berlin, Germany, 16–30.