$bc - opt$: a Branch-and-Cut Code for Mixed Integer Programs *

Cécile Cordier[†]Hugues Marchand[‡], Richard Laundy[§], Laurence A. Wolsey[¶]

October 9, 1997

Abstract

A branch-and-cut mixed integer programming system, called $bc - opt$, is described, incorporating most of the valid inequalities that have been used or suggested for such systems, namely lifted 0-1 knapsack inequalities, 0-1 gub knapsack and integer knapsack inequalities, flow-cover and continuous knapsack inequalities, path inequalities for fixed charge network flow structure and Gomory mixed integer cuts. The principal development is a set of interface routines allowing these cut routines to generate cuts for new subsets or aggregations of constraints.

The system is built using the XPRESS Optimisation Subroutine Library (XOSL) which includes a cut manager that handles the tree and cut management, so that the user only essentially needs to develop the cut separation routines.

Results for the MIPLIB3.0 library are presented - 37 of the instances are solved very easily, optimal or near optimal solution are produced for 18 other instances, and of the 4 remaining instances, 3 have 0, +1, -1 matrices for which $bc - opt$ contains no special features.

*This work has been supported in part by the Esprit projects 8755 (PAMIPS) and 20118(MEMIPS).

[†]Cécile Cordier, CORE, Université Catholique de Louvain.

[‡]Hugues Marchand, CORE, Université Catholique de Louvain. Supported in part by a doctoral fellowship from Collège Interuniversitaire pour les Sciences du Management (CIM).

[§]Richard Laundy, Dash Associates, Leamington Spa.

[¶]Laurence A. Wolsey, CORE and INMA, Université Catholique de Louvain.

1 Introduction

Over the last twenty years a large number of specialised branch-and-bound codes based on strong cutting planes have been developed for a variety of combinatorial optimisation problems, such as the travelling salesman problem [18],[10], the max cut problem [3], the two-connected network problem [11], etc. The first more general code using strong cutting planes was that of Crowder, Johnson and Padberg [8] for pure 0-1 problems incorporating lifted cover inequalities. To handle mixed 0-1 programs, MPSARX [23] also included flow cover inequalities and some simple (fixed charge) path inequalities. Both these codes were cut-and-branch codes in which cuts were only generated at the root of the enumeration tree. MINTO [22] was the first branch-and-cut code incorporating cover and flow cover inequalities, and later gub-cover inequalities [12]. More recently MIPO [1] is a branch-and-cut system designed for mixed 0-1 problems using lift-and-project cuts, that has also been used to test Gomory mixed integer cuts [2].

The branch-and-cut system described here, called $bc-opt$, incorporates many features from the earlier codes such as lifted cover, flow cover, simple path, gub-cover inequalities and Gomory mixed integer cuts. It also includes new routines including integer knapsack inequalities and knapsacks with continuous variables. However the main new feature is a set of *model interface routines*, creating new model relaxations on which the existing cut routines can generate inequalities. The system is based on the XPRESS-MP system and is built using the corresponding subroutine library XOSL [25].

On the MIPLIB 3.0 library [5] of mixed integer test instances, $bc-opt$ solves 37 of the problems very easily, and produces provably optimal or near-optimal solutions to 18 other problems. 3 of the remaining 4 problems are set covering problems with 0,1 or 0,+1,-1 matrices for which $bc-opt$ contains no special features.

Our goal in the paper is to briefly describe the $bc-opt$ branch-and-cut software as it has existed for the last couple of years, and the results obtained with it. Thus in Section 2 we just give the idea of a branch-and-cut algorithm, and describe the components of a generic cut routine. In Section 3 we describe the canonical structures used for cut generation. The cutting plane and separation algorithms for these structures can all be found in the literature. In Section 4 we present the model interfaces which start from original model constraints and their classification and convert them to one or more of the canonical structures. In Section 5 further details of the

$bc-opt$ branch-and-cut system are given, and in Section 6 computational results are presented.

2 Branch-and-Cut for MIP

We consider the problem:

$$(IP) \qquad z = \max\{cx : x \in S\}$$

where an initial formulation $P = \{x \in R^n_+ : Ax \leq b\}$ of the set of feasible solutions $S = P \cap Z^n$ is given. For simplicity of notation, the algorithm is described for an integer program, but it applies just as well for a mixed integer program.

A branch-and-bound algorithm to solve problem IP consists of:

- breaking up unsolved subproblems into new subproblems by partitioning the set of feasible solutions. With subproblem j of the form $z^j = \max\{cx : x \in P^j \cap Z^n\}$, it is important to stress that its formulation is given by the polyhedron P^j, and not just by the feasible region $S^j = P^j \cap Z^n$.
- bounding the value of the objective function for each subproblem j. This phase consists of the solution of the linear program $\bar{z}^j = \max\{cx : x \in P^j\}$ with optimal solution $\bar{x}^j$. Thus $\bar{z}^j = c\bar{x}^j \geq z^j$,and if $\bar{x}^j \in Z^n$, then $z^j = c\bar{x}^j \leq z$.

The choice of the formulation is therefore crucial in generating good bounds on the objective value. In a Branch-and-Cut approach, the formulation P^j is progressively tightened by adding inequalities valid for the set S^j but violated by the solution of the current relaxation $\overline{x}^j$.

The main approach used in $bc-opt$ is to develop strong valid inequalities for well-defined structures and then generate cuts whenever these structures are found as part of a problem instance.

The theoretical development of such inequalities typically involves two steps:
the derivation of a *class of valid inequalities* for some *canonical structure*, and
a *separation algorithm* (exact or heuristic) that, given a point, tries to find a violated inequality from this class.

The implementation of a cut routine thus involves three layers:
a *cut generation routine* based on the separation algorithm for the canonical structure;
a *model interface routine* that converts the specific model instance into this canonical structure, and, if necessary, converts back a violated inequality for the structure into an inequality in the original variables of the instance. This conversion is done by examining the current solution and by using information about the general problem structure;
a *routine to recognise this general problem structure*: classifying the constraints of the problem, recognising variable upper (lower) bound as well as generalised upper bound constraints (explained in Section 3). This routine is just called once, after the instance is read in initially.

The classes of inequalities and the separation routines implemented in $bc-opt$ are quite standard, but an effort has been made to use these cut routines as extensively as possible, by refining the model interface routines.

In the next section we describe the structures on which $bc-opt$ tries to generate cuts, referring to the literature for the detailed description of the inequalities generated and the separation algorithm. In section 4, we give details about how our model interface routines recognise these canonical structures.

3 Canonical Structures

In this section, we describe six canonical sets. For each we give references for the valid inequalities and separation heuristics implemented in $bc-opt$, as well as any special features of our implementation.

3.1 The integer knapsack set

$$X^K = \{y \in Z^n : \sum_{j \in N} a_j y_j \leq b, y_j \leq u_j \text{ for } j \in N\}$$

with $a_j > 0$ for $j \in N$ and $b \geq 0$. For this set, separation routines for lifted cover inequalities are described in [8], [23], [12].

3.2 The knapsack set with a continuous variable

$$X^{KC} = \{(y, s) \in Z^n_+ \times R^1_+ : \sum_{j \in N} a_j y_j \leq b + s, y_j \leq u_j \text{ for } j \in N\}$$

where $a_j > 0$ for $j \in N$, and $b \geq 0$. Here, as described in [6], cover inequalities are derived for the integer knapsack set, and the continuous variable is then lifted.

3.3 The 0-1 knapsack set with gubs

$$X^{GK} = \{y \in Z^n_+ : \sum_{j \in N} a_j y_j \leq b, \sum_{j \in B_k} y_j \leq 1 \text{ for } k \in K\}$$

where $a_j > 0$ for $j \in N$, $B_k \cap B_{k'} = \emptyset$ if $k \neq k'$ and $\cup_{k \in K} B_k = N$. The constraints $\sum_{j \in B_k} y_j \leq 1$ are called *gubs* (generalised upper bound constraints). Cover inequalities for such sets are described in [24] and separation heuristics are tested extensively in [12].

3.4 The single node flow set

$$X^F = \{(x, y) \in R^n_+ \times B^n : \sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j \leq b \quad (1)$$

$$l_j y_j \leq x_j \leq u_j y_j \text{ for } j \in N\} \quad (2)$$

The constraints $l_j y_j \leq x_j$ and $x_j \leq u_j y_j$ are called *vlb* (variable lower bounds) and *vub* (variable upper bounds) respectively. The family of flow cover inequalities are described in [19]. Separation routines are described in [23],[17] and a computational study of lifted flow cover separation heuristics is presented in [13].

3.5 The fixed charge path set

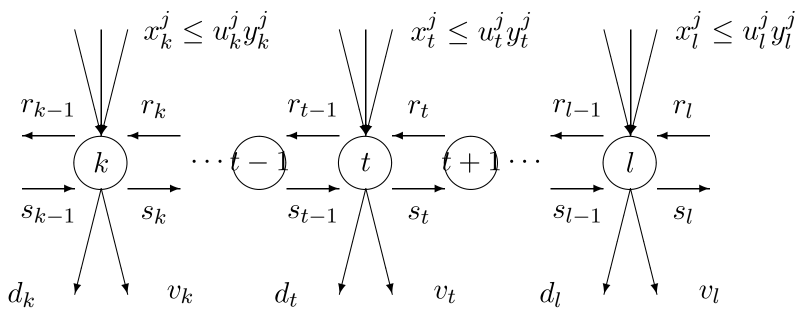

Figure 1: X^P

$$s_{t-1} - r_{t-1} + \sum_{j \in N_t} x_t^j = d_t + v_t + s_t - r_t \text{ for all } t \tag{3}$$

$$x_t^j \leq u_t^j y_t^j \text{ for } j \in N_t, v_t \leq h_t \text{ for all } t \tag{4}$$

$$s_t, r_t, v_t \geq 0 \; for \; all \; t, x_t^j, y_t^j \in \{0,1\} \text{ for all } j \in N_t \text{ and all } t. \tag{5}$$

Such a path models part of a fixed charge network flow problem. In particular for a lot-sizing problem, the variables s_t, r_t can be interpreted as stock and backlog variables in period t, the variables x_t^j as the amount produced by process j in period t, y_t^j is the associated fixed charge variable that takes the value 1 if the corresponding arc is used (process j is active in the period), v_t is the amount sold in period t , while d_t is the demand.

A family of path inequalities and their separation are presented in [23]. The inequalities are a generalization of inequalities for the uncapacitated lot-sizing problem.

3.6 The LP tableau row

$$X^G = \{(x, y, y_0) \in R_+^n \times Z_+^p \times Z_+^1 : y_0 + \sum_{j \in N_1} a_j y_j + \sum_{j \in N_2} a_j x_j = a_0\}$$

where the associated LP solution has $y_0 = a_0 \notin Z, y_j = 0$ for $j \in N_1, x_j = 0$ for $j \in N_2$. The Gomory mixed integer cut for such sets can be generated by inspection [9],[17]. See [2] for recent computational experience with such cuts. As the cuts are typically dense, care is taken in $bc - opt$ to limit the number of variables and not to generate too many cuts, because otherwise the linear programs quickly become very difficult to solve.

4 Model Interfaces

The model interface routines of $bc - opt$ reduce rows or sets of rows of some specific class into the canonical sets. In this section, we start by describing the row classification routines on which these interfaces are based and then we give examples of such reductions.

4.1 Row Classification

One of the first tasks of $bc - opt$ is to classify rows in preparation for the cut separation routines. Rows are classified, based on the types of variables occurring in the row, as:

- 0-1 rows
- integer rows
- variable lower/upper bound constraints
- gub constraints
- mixed integer rows if the row contains both continuous and discrete variables, or if some continuous variables have associated 0-1 variable lower and upper bound constraints
- continuous rows

This classification is similar to that employed in other codes such as [22],[23].

4.2 Reduction of Integer Rows

a) **Reduction of Integer Rows to the form** X^K

Suppose that the model contains a 0-1 or integer constraint, which together with simple bounds lead to a set of the form:

$$\{y \in Z^n_+ : \sum_{j \in N} a_j y_j \leq b, l_j \leq y_j \leq u_j \text{ for } j \in N\}$$

The substitution $y'_j = y_j - l_j$ for $j \in N$ with $a_j > 0$ and $y'_j = u_j - y_j$ if $a_j < 0$ leads directly to a set in the canonical form X^K.

The reduction of 0-1 rows and gub constraints to the form X^{GK} is similar, see [14].

4.3 Reduction of Mixed Integer Rows

a) **Reduction of Mixed Integer Rows to Flow Sets** X^F

Suppose the instance contains a mixed 0-1 row, plus variable lower and upper bounds of the form:

$$\sum_{j \in N_1} (a_j x_j + g_j y_j) + \sum_{j \in N_2} a_j x_j + \sum_{j \in N_3} g_j y_j \leq b$$

$$\tilde{l}_j y_j \leq x_j \leq \tilde{u}_j y_j \text{ for } j \in N_1, \tilde{l}_j \leq x_j \leq \tilde{u}_j \text{ for } j \in N_2, l'_j \leq y_j \leq u'_j \text{ for } j \in N_1 \cup N_3$$

$$y_j \in \{0,1\} \quad \text{for } j \in N_1 \cup N_3$$

where $a_j \neq 0, a_j g_j \geq 0$ for $j \in N_1, \tilde{l}_j \geq 0$ for $j \in N_1 \cup N_2, a_j \neq 0$ for $j \in N_2, g_j \neq 0$ for $j \in N_3$, and N_1, N_2, N_3 is a partition of N.

Let $N_i^+ = \{j \in N_i : a_j > 0\}$ $i = 1, 2$ and $N_3^+ = \{j \in N_3 : g_j > 0\}$.

Setting $z_j =\mid a_j x_j + g_j y_j \mid$ for $j \in N_1, z_j =\mid a_j x_j \mid$ for $j \in N_2, z_j =\mid g_j y_j \mid$ for $j \in N_3$, we obtain

$$\sum_{j \in N_1^+ \cup N_2^+ \cup N_3^+} z_j - \sum_{j \in N_1^- \cup N_2^- \cup N_3^-} z_j \leq b$$

$\mid a_j \tilde{l}_j + g_j \mid y_j \leq z_j \leq\mid a_j \tilde{u}_j + g_j \mid y_j$ for $j \in N_1$,

$\mid a_j \mid \tilde{l}_j \leq z_j \leq\mid a_j \mid \tilde{u}_j$ for $j \in N_2, z_j =\mid g_j \mid y_j$ for $j \in N_3$.

Finally introducing variables y_j for $j \in N_2$ with $l'_j = u'_j = 1$, we obtain a set of the from X^F.

b) **Reduction of Mixed Integer Rows to Knapsack Sets** X^K

Suppose for simplicity of exposition that the mixed integer row has been reduced as in 4.3a) to a flow set where the y_j are now general integer variables. The resulting set is of the form:

$$\begin{aligned} \{(x, y) \in R_+^{|N|} \times Z_+^{|N|} : \sum_{j \in N^+} x_j - \sum_{j \in N^-} x_j \leq b \\ l_j y_j \leq x_j \leq u_j y_j,\ l'_j \leq y_j \leq u'_j \text{ for } j \in N\} \end{aligned}$$

where (N^+, N^-) is a partition of N.

Replace x_j by $l_j y_j$ for $j \in N^+$ and by $u_j y_j$ for $j \in N^-$. The resulting relaxation is a pure integer row:

$$X' = \{y \in Z_+^{|N|} : \sum_{j \in N^+} l_j y_j - \sum_{j \in N^-} u_j y_j \leq b, l'_j \leq y_j \leq u'_j \text{ for } j \in N\}.$$

Now, by the reduction of Section 4.2a), this can be converted to the form X^K.

c) **Reduction of Mixed Integer Rows to the form** X^{KC}

Suppose again that the mixed integer row has been reduced to a flow set with general integer variables X^F of the form (1),(2), and that (x^*, y^*) is the current value of the variables occurring in X^F.

Let $N_l^+ = N^+ \cap \{j \in N : x_j^* - l_j y_j^* \leq u_j y_j^* - x_j^*\}$ and $N_u^+ = N^+ \setminus N_l^+$. Similarly define N_l^- and N_u^-. Replace x_j by $l_j y_j$ for $j \in N_l^+$, by $l_j y_j + s_j$ for $j \in N_l^-$, by $u_j y_j$ for $j \in N_u^-$ and by $u_j y_j - s_j$ for $j \in N_u^+$. Let $s = \sum_{j \in N_l^- \cup N_u^+} s_j$. The resulting relaxation is:

$$X' = \{y \in Z_+^{|N|} : \sum_{j \in N_l^+} l_j y_j + \sum_{j \in N_u^+} u_j y_j - \sum_{j \in N_l^-} l_j y_j - \sum_{j \in N_u^-} u_j y_j \leq b + s,$$

$$l_j' \leq y_j \leq u_j' \text{ for } j \in N\}.$$

Now using the same transformation as in Section 4.2a, this can be converted to the form X^{KC}.

Example 1.

We consider a set of constraints arising in *gesa3.mat* in the MIPLIB3.0 test library. The set involves the satisfaction of demand for electricity on island 2 in period 14. The set is:

$$x_2 + x_3 + x_4 - 0.13y_2 - 0.26y_3 - 0.35y_4 + 0.97v_1 - 1.5yv_1 = 42.64 + v_2$$

$$1.96y_2 \leq x_2 \leq 10.78y_2, 4.9y_3 \leq x_3 \leq 34.3y_3,$$

$$7.44y_4 \leq x_4 \leq 13.02y_4, 0 \leq v_i \leq 45yv_i \text{ for } i = 1,2$$

$$y_2, y_3, yv_1, yv_2 \in \{0,1\}, 0 \leq y_4 \leq 3 \text{ and integer.}$$

Here x_j represents the electricity produced by generators of type j, y_j is the number of generators active, v_1, v_2 are the shipments of electricity into and away from the island, and yv_i $i = 1,2$ the associated 0-1 variables.

The linear programming solution is $x_4^* = 0.514, y_4^* = 0.069, yv_1^* = 1, v_1^* = 45$ with all other variables zero.

Model Interface to X^{KC}. As $x_j^* = u_j y_j^*$ $j = 2,3$, $v_1^* = 45yv_1^*$ and $v_2^* = yv_2^* = 0$, and $s_4^* = x_4^* - l_4 y_4^* = 0$, we choose to relax x_j to $u_j y_j$ for $j = 2,3$, to replace x_4 by $l_4 y_4 + s_4$, $s_4 \geq 0$, to relax v_1 to $45yv_1$, and to relax v_2 to 0. The resulting relaxation

$$(10.78-0.13)y_2+(34.3-0.26)y_3+(7.44-0.35)y_4+s_4+(0.97\times45-1.5)yv_1 \geq 42.64,$$

is a canonical knapsack with continuous variable set X^{KC} (to keep the physical interpretation of the variables, we have not complemented the integer variables):

$$10.65y_2 + 34.04y_3 + 7.09y_4 + 42.15yv_1 + s_4 \geq 42.64$$

$$y_2, y_3, yv_1 \in \{0,1\}, 0 \leq y_4 \leq 3 \text{ and integer}, s_4 \geq 0$$

with linear programming solution $(y_2^*, y_3^*, y_4^*, yv_1^*, s_4^*) = (0, 0, 0.069, 1, 0)$.

The X^{KC} separation heuristic produces the inequality

$$y_2 + y_3 + y_4 + yv_1 + (42.64 - 42.15)^{-1} s_4 \geq 2.$$

Returning to the original space by eliminating s_4 gives the valid inequality

$$y_2 + y_3 + y_4 + yv_1 + (42.64 - 42.15)^{-1}(x_4 - 7.44y_4) \geq 2$$

cutting off the original point with violation of 0.93. ■

4.4 Reduction of Sets of Mixed integer Rows

a) **Reduction to simple paths** X^P

Starting from a set of mixed integer rows and variable upper bound constraints, a greedy row-by-row path augmenting procedure described in [23] either terminates with a set in the form X^P or decides that the set of rows does not correspond to a path in a fixed charge network.

b) **Reduction to an aggregated path set** $\bar{X}^P$

Suppose that a path X^P has been constructed, see (3)-(5), consisting of nodes $k, k+1, \ldots, l$ asin Figure 1. We sum up the flow balance constraints of X^P. Adding simple uncapacitated path inequalities for x_t^j, we then obtain the set $\bar{X}^P$

$$s_{k-1} - r_{k-1} + \sum_{t=k}^{l} \sum_{j \in N_t} x_t^j - \sum_{t=k}^{l} v_t - s_l + r_l = \sum_{t=k}^{l} d_t$$

$$x_t^j \leq u_t^j y_t^j \text{ for } j \in N_t, v_t \leq h_t \;\; t = k, \ldots, l$$

$$x_t^j \leq r_{p_t^j - 1} + (\sum_{\tau = p_t^j}^{q_t^j} d_\tau) y_j + s_{q_t^j} + \sum_{\tau = p_t^j}^{q_t^j} v_\tau \text{ for } j \in N_t, \quad t = k, \ldots, l$$

$$x, s, r, v \geq 0, y \in \{0,1\}$$

where for each $j \in N_t$ and all t, $[p_t^j, q_t^j]$ is an interval containing t, or is empty. The additional inequality simply says that if one considers the subpath $p_t^j, \ldots, q_t^j$, the inflow x_t^j either exits through a demand node, or by one of the outflow arcs.

This intermediate structure is used in the two reductions described below.

c) **Reduction to a flow set** X^F

Starting from an aggregated path set $\bar{X}^P$, temporarily set (project) the variables $r_{p_t^j - 1} = s_{q_t^j} = 0$ for $j \in N_t, t = k, \ldots, l$, and $v_t = 0$ for $t = k, \ldots, l$.

Eliminating the nonnegative variables s_{k-1}, r_l, we obtain a canonical flow set X^F in the form:

$$\sum_{t=k}^{l} \sum_{j \in N_t} x_t^j \leq \sum_{t=k}^{l} d_t + r_{k-1} + s_l$$

$$x_t^j \leq \min[u_t^j, \sum_{\tau = p_t^j}^{q_t^j} d_\tau] y_j \; j \in N_t,$$

$$x, s, r \geq 0, y \in \{0, 1\}$$

Note that if a flow cover inequality $\sum \pi_j z_j \leq \pi_0$ is generated, normalised so that the flow variables have unit coefficients, lifting back the projected variables gives an inequality

$$\sum \pi_j z_j \leq \pi_0 + \sum_{t=k}^{l} \sum_{j \in N_t} r_{p_t^j - 1} + \sum_{t=k}^{l} \sum_{j \in N_t} s_{q_t^j} + \sum_{t=k}^{l} v_t$$

valid for $\bar{X}^P$, X^P and the original instance. Variants of this inequality can be obtained by substituting $v_t = h_t - \bar{v}_t$ with $\bar{v}_t \geq 0$.

d) **Reduction to a Knapsack Set with Continuous Variable** X^{KC}

Starting from an aggregated path set $\bar{X}^P$, eliminating the nonnegative variables r_{k-1}, s_l, and replacing x_t^j directly by its variable upper bound $u_t^j y_t^j$, we obtain the knapsack with continuous variable set X^{KC}

$$s_{k-1} + r_l + \sum_{t=k}^{l} \sum_{j \in N_t^+} u_t^j y_t^j \geq \sum_{t=k}^{l} d_t$$

$$s_{k-1}, r_l \geq 0, y_t^j \in \{0, 1\} \text{ for } j \in N_t, t = k, \ldots, l$$

Here any valid inequality for X^{KC} is valid for $\bar{X}^P$, X^P and the original instance.

Example 2.

Here we consider a constant capacity lot-sizing problem without backlogging. For the item under consideration, the model is:

$$\begin{aligned} s_{t-1} + x_t = d_t + s_t \ t = 1, \ldots, 6 \\ x_t \le u y_t \ t = 1, \ldots, 6 \\ s_t, x_t \ge 0, y_t \in \{0,1\} \ t = 1, \ldots, 6 \end{aligned}$$

The data are $d = (3, 7, 6, 9, 4, 5), u = 10$, and the linear programming solution is $x^* = (3, 7, 6, 10, 8, 0), y^* = (1, 1, 1, 1, 0.8, 0), s^* = (0, 0, 0, 1, 5, 0)$.

The reduction routine generates the simple path shown in Figure 2.

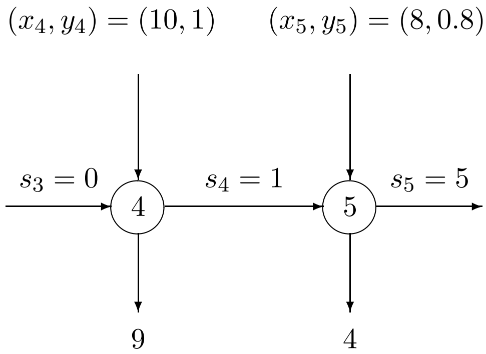

Figure 2: Constant Capacity Lot-Sizing Path

Reduction to an aggregate path set produces the set $\bar{X}^P$

$$s_3 + x_4 + x_5 = 13 + s_5$$

$x_4 \le 10y_4, x_5 \le 10y_5, x_4 \le 13y_4 + s_5, x_5 \le 4y_5 + s_5, x_4, x_5, s_3, s_5 \ge 0, y_4, y_5 \in \{0, 1\}$.

Reducing to a node set X^F, we obtain:

$$x_4 + x_5 \le 13 + s_5$$

$$x_4 \le 10y_4, x_5 \le 4y_5, x_4, x_5, s_5 \ge 0, y_4, y_5 \in \{0, 1\}.$$

The flow cover separation routine then generates the inequality

$$x_4 + x_5 \le 1 + 9y_4 + 3y_5 + s_5$$

violated by 0.6.

Reducing to a knapsack set with continuous variable X^{KC}, we obtain the set

$$s_3 + 10y_4 + 10y_5 \geq 13, s_3 \geq 0, y_4, y_5 \in \{0, 1\},$$

and the separation routine then generates the inequality

$$\frac{1}{3}s_3 + y_4 + y_5 \geq 2$$

violated by 0.2. ■

The two interface routines c) and d) enable us to generate tighter inequalities when capacities are present. For capacitated lot-sizing problems, see [21], these are in many cases facet-defining.

Finally the description of X^P and Figure 2 can be generalised to include the possibility of arcs between non-adjacent nodes in the path. Such arcs are allowed within the path construction routine, and either cancel out during the aggregation procedure, or appear as additional outflow arcs in the inequalities.

5 The bc-opt System

$bc - opt$ has been developed with the tools provided by the XPRESS subroutine library (XOSL). XPRESS-MP[25] is one of the major commercial mixed integer programming systems, and the subroutine library allows easy access to a series of subroutines so as to:

- load matrices, names, priority files
- carry out optimisation tasks such as solving LPs and IPs and get bases
- view models and solutions
- modify models
- read and change control variables
- handle output
- compile, link, etc.

A unique feature of XOSL is a *cut manager* which handles the management of a cut pool, thereby providing a branch-and-cut algorithm in which the user only needs to provide cut separation routines. Essentially the branch-and-cut algorithm is implemented as shown in figure 3.

Initialisation The first task of $bc - opt$, once the model has been read and preprocessed, is to recognize and store the vub, vlb and gub structures in the model, and classify the rows in preparation for the cut separation routines as described in Section 2.

Cutting Phase After optimising the current node, a routine is called which in turn calls the user provided cut separation routines. Each separation routine runs through the formulation row by row using the initial row classification to decide if an appropriate canonical set can be constructed, and if so the separation routine is called. A complete cycle through the rows is called a *pass*. Generated cuts can be added to the matrix and cuts which are no longer binding can be removed. Having done this the node is re-optimised and it is possible to call the cut manager callback routine again to generate more cuts.

Branching Once the cut manager finishes processing a node, the branch-and-bound algorithm continues in the usual way: an integer variable whose linear programming value is fractional is selected for branching, and the node is split into two by applying upper and lower bounds on the branching variable. Once a subproblem is split into two, the new subproblems are added to the node list with the appropriate pointers to the cut pool.

Node Selection Previously the XPRESS default strategy was closer to depth-first. It has been found useful to increase the options available in XPRESS for node selection. Two important options are

i) the choice of a best bound strategy for a certain number of initial nodes. This turns out to be effective when the cuts added by the system succeed in reducing the duality gap.

ii) the use of a temporary cutoff. The idea is that if one has a good a priori estimate of the optimal value, the temporary cutoff can be used to prevent the algorithm wasting time searching in parts of the tree that would not need to be explored if an optimal solution had already been found.

INITIALISATION

The problem is: $z = \max \quad cx$

$$x \in \mathcal{S} = \mathcal{P} \cap \mathcal{Z}^n$$

where $\mathcal{P} = \{x \in \mathcal{R}^n_+ | Ax \leq b, l_j \leq x_j \leq u_j \forall j\}$

Perform **Row Classification**, store **vubs**, **vlbs**, **gubs**.

Set $\mathcal{P}^0 = \mathcal{P}; u^0_j = u_j \ \forall j; l^0_j = l_j \ \forall j; \underline{z} = -\infty; NodeList = \emptyset; i = 0$

RESTORE

Restore formulation $\mathcal{P}^i$ for problem i: $z^i = \max \quad cx$

$$x \in \mathcal{S}^i$$

$$\text{where} \quad \begin{array}{ll} \mathcal{S}^i & = \mathcal{P}^i \cap \mathcal{Z}^n \\ \mathcal{P}^i & = \{x \in \mathcal{R}^n_+ | \quad Ax \leq b \\ & \qquad\qquad\qquad \Pi^i x \leq \Pi^i_0, l^i_j \leq x_j \leq u^i_j \forall j\} \end{array}$$

Remove i from $NodeList$

LP RELAXATION

Solve $\overline{z}^i = \max \quad cx = \quad c\overline{x}^i$

$$x \in \mathcal{P}^i$$

CUTTING - Iteration k

Look for $\left(\Pi^{i,k} \Pi^{i,k}_0\right)$

s.t. $\quad \Pi^{i,k}\overline{x}^i > \Pi^{i,k}_0 \quad$ and $\quad \Pi^{i,k} x \leq \Pi^{i,k}_0 \forall x \in \mathcal{S}^i$

If NO CUTS found goto **PRUNING**

else set $\Pi^i = \begin{pmatrix} \Pi^i \\ \Pi^{i,k} \end{pmatrix} \quad \Pi^i_0 = \begin{pmatrix} \Pi^i_0 \\ \Pi^{i,k}_0 \end{pmatrix}$ goto **LP RELAX**

PRUNING

If $\overline{z}^i \leq \underline{z}$ goto **NODE**

else if $\overline{x}^i \in \mathcal{S}^i$ then set $\underline{z} = \overline{z}^i$ goto **NODE**

BRANCHING

Choose k s.t. $\overline{x}^i_k$ fractional

Create 2 new problems ($\mathcal{S}^i = \mathcal{S}^{i+} \cup \mathcal{S}^{i-}$)

$l^{i+}_k = \lceil \underline{x}^i_k \rceil, u^{i-}_k = \lfloor \underline{x}^i_k \rfloor$

Update $NodeList = NodeList \cup \{i+, i-\}$

NODE

If $NodeList = \emptyset$ goto **EXIT**

Choose $i \in NodeList$ goto **RESTORE**

EXIT

If $\underline{z} > -\infty$ then $z = \underline{z}$

Figure 3: Different steps of a Branch-and-Cut Algorithm

As the latter option depends on some knowledge of the problem instance, it may be particularly useful when several similar instances of the same problem are solved.

$bc - opt$ also contains several other features and options. During the processing of a subproblem,
i) Apart from the six canonical separation routines described in Section 3.2 combined with the different interface routines of Section 4, it is also possible to use *model cuts*. Here a set of constraints is introduced as part of the initial matrix, but these constraints are removed from the formulation and stored in the cut pool from which they can be loaded whenever violated. This is useful when a class of valid inequalities can be described explicitly, but the addition of a very large number of rows would significantly slow down the solution of the linear programs.
ii) reduced cost fixing is used, and
iii) a primal heuristic based on successive rounding of fractional variables is available.

In developing the branch-and-cut tree, the user has the option
i) to define a cutting plane strategy by selecting which cuts to look for,
ii) to generate cuts every x nodes, or every y levels in the tree,
iii) to generate either locally valid cuts or globally valid cuts by using local/global bounds on variables. Violated globally valid cuts can be added at any node of the search tree, and
iv) to delete inactive cuts.

6 Computational Results

The development of $bc - opt$ has been part of an ESPRIT financed project PAMIPS [20]. The project provided a series of practical production planning, network design and electricity generation problems as benchmarks. In addition various other difficult problems encountered by XPRESS, and problems encountered in studying mixed integer programming formulations have been used as tests.

Below we report results on the MIPLIB3.0 library [5] which is a set of integer and mixed integer programs assembled as a testbed for researchers and software developers. Some of the PAMIPS instances already form part of this library.

6.1 The MIPLIB3.0 Test Set

MIPLIB3.0 contains 59 instances. Rather than treat matrices in abstract, we believe that improved formulations must in many cases be based on structure, so below we also classify the instances by type and/or difficulty: SC=set-covering, BP=pure 0-1, FN=fixed charge network, PP=production planning, EG=electricity generation, GT=generalised transportation, FL= facility location, D=diverse or unclear.

37 of the 59 problems can be classified as easy for $bc - opt$ in that they are solved within 5 minutes with the default strategy on a Pentium PRO 200 with 64 M of RAM. The default strategy is:

- At the top node: 5 rounds of cuts (one of pure knapsack inequalities with and without gubs, one of flowcover and knapsack with continuous variable inequalities, one of path inequalities, one of Gomory cuts, and the last round with all the cut types). Non binding cuts are deleted.
- In the tree: knapsack and flowcover cuts are generated every 8 levels. Non binding cuts are deleted.
- Tree search: use a best bound for $2^7 - 1$ nodes, and then the XPRESS default strategy.

These easy problems are presented in Tables 1 and 2. Table 1 contains 14 pure 0-1 instances and Table 2, 23 mixed integer problems. Column 1 contains the MIPLIB3.0 name, column 2 the problem type (our classification), columns 3-6 the number of rows, binary, integer and continuous variables respectively. Columns 7-9 headed LP, XLP and IP present the value of the initial LP after automatic preprocessing, the LP value after adding cuts at the top node before branching, and IP the optimal value. Columns 10 and 11 contain the time required in seconds to prove optimality, and the number of nodes in the branch-and-cut tree respectively when using Cuts every 8 levels in the tree, whereas columns 12 and 13 correspond to the cut-and-branch case when cuts are added at the top node only. Column 14 indicates which cuts are generated by $bc - opt$ (BK=0-1 knapsack, GK=gub knapsack, IK=integer knapsack, FC=flow cover, KC=knapsack with continuous variable, PI=Path inequalities, GM=Gomory mixed integer)

In Table 3 we list 8 other instances that can be solved to optimality but require a greater computational effort. These results are for the same default strategy but allowing a maximum time of 4 hours.

instance	Class	m	B	I	C	LP	XLP	IP	Secs	Nodes	Secs	Nodes	Type
									Branch&Cut		Cut&Branch		
p0033	BP	16	33	0	0	2819	3089	3089	0	1	0	1	BK,GK,GM
p0201	BP	133	201	0	0	7125	7125	7615	13	956	10	1022	GK
p0282	BP	241	282	0	0	180000	256512	258411	3	69	3	91	BK,GK
p0548	BP	176	548	0	0	426	8691	8691	2	1	2	1	BK,GK
p2756	BP	236	2756	0	0	2701	3117	3124	59	668	624	15399	BK,GK,GM
enigma	BP	21	100	0	0	0	0	0	1	315	2	598	BK,GK
fiber	BP	348	1195	0	0	156082	385255	405935	9	152	10	203	BK,GK
lseu	BP	28	89	0	0	944	1030	1120	3	878	3	1012	BK,GK
misc03	BP	96	159	0	1	1910	1910	3360	5	644	5	806	BK
mod008	BP	6	319	0	0	290	294	307	31	1664	50	1262	BK
mod010	BP	146	2655	0	0	6532	6535	6548	6	19	6	19	GK
air03	SC	124	10757	0	0	338864	340159	340159	51	1	51	1	GM
stein27	SC	28	27	0	0	13	13	18	18	9903	18	9903	GM
mitre	BP	1663	10724	0	0	114782	115155	115155	65	48	72	132	GK

Table 1: 0-1 MIPLIB3 problems easy with bc-opt default

instance	Class	m	B	I	C	LP	XLP	IP	Secs	Nodes	Secs	Nodes	Type
									Branch&Cut		Cut&Branch		
bell3a	FN	98	27	29	54	866171	873883	878430	189.9	49365	145	49159	GM
egout	FN	41	28	0	24	511	568	568	0	6	0	6	FC,KC,PI,GM
fixnet6	FN	478	378	0	499	3192	3634	3983	14	133	16	261	FC,KC,PI
modglob	FN	287	98	0	286	20430947	20720532	20740508	11	468	10	552	FC,KC,PI,GM
qnet1	FN	371	1288	129	0	14274	15664	16029	14	30	14	30	IK
qnet1_O	FN	333	1288	129	0	12095	15663	16029	6	13	6	13	IK
pp08a	PP	134	64	0	170	2748	7192	7350	21	1456	20	1772	FC,KC,PI,GM
pp08aCuts	PP	244	64	0	173	5480	7166	7350	42	1763	31	1324	FC,KC,PI,GM
rgn	PP	25	100	0	80	48.8	66	82.2	12	2276	11	2474	FC,PI,GM
set1ch	PP	424	235	0	408	35118	54517	54537	7	120	7	120	FC,KC,PI,GM
vpm1	PP	155	104	0	127	16.43	19.5	20	3	319	2	324	FC,KC,PI,GM
gen	EG	479	108	5	534	112233	112313	112313	1	1	1	1	BK,FC
gesa2	EG	1345	240	168	768	25492512	25726923	25779856	194	3968	151	4485	GM,KC
gesa3	EG	1297	216	168	696	27846437	27919556	27911042	26	383	19	440	IK,KC,GM
gesa3_O	EG	1153	336	312	432	27833632	27916560	27911042	15	438	12	445	KC,GM
gt2	GT	29	22	151	0	13460	20670	21160	1	13	1	13	IK,GM
khb	FL	101	24	0	1275	95919464	106735640	106940225	2	27	2	27	FC,KC,PI
dcmulti	FL	258	75	0	458	184034	184573	188182	12	987	10	935	FC,KC,GM
blend2	D	179	227	20	81	6.91	7.01	7.59	107	10013	88	14678	KC
dsbmip	D	1028	160	0	1479	-305	-305	-305	91	803	39	392	FC,GM
flugpl	D	17	0	10	6	1167185	1172560	1201500	1	357	23	32235	IK,GM
rentacar	D	967	28	0	2717	28928379	29363158	30356761	30	24	29	24	FC,KC,GM
misc06	D	552	112	0	1295	12841	12844	12850	3	98	3	98	GM

Table 2: Mixed Integer MIPLIB3 problems easy with bc-opt default

instance	Class	m	B	I	C	LP	XLP	IP	Secs	Nodes	Secs	Nodes	Type
									Branch&Cut		Cut&Branch		
air04	SC	783	8904	0	0	55535	55535	56137	13782	3218	13608	3218	
air05	SC	409	7195	0	0	25877	25877	26374	13879	10763	13701	10763	
stein45	SC	332	45	0	0	22	22	30	739	141994	739	141994	GM
misc07	BP	212	253	0	0	1415	1415	2810	528	31784	902	62696	BK
l152lav	BP	98	1989	0	0	4656	4656	4722	773	15892	787	17886	GK
cap6000	BP	2172	5995	0	0	-2451537	-2451524	-2451403	1289	7989	1201	8838	BK,GK
qiu	D	1193	49	0	792	-931	-703	-132	5320	40942	4475	40942	KC
vpm2	PP	155	104	0	127	10.26	12.16	13.75	3436	295946		***	FC,KC,PI,GM

Table 3: Problems solved using more ressources (*** Time limit exceeded)

Table 4 contains results for 10 hard instances after running for 4 hours. For 6 of these problems the limiting factor was the memory rather than the time limit (they exceeded 32000 actives nodes). Column IP contains the best IP solution known for this problem. For each of these problems the Best Lower Bound is shown in the column headed BLB, the value of the best feasible solution found in column BIP and the gap between these bounds as a percentage of BIP.

Four instances appear out of reach with the present code, three are 0-1 set covering problems with very large $(0, +1, -1)$ matrices $(fast0504, nw04$ and *seymour*) for which $bc-opt$ has no specialized routines. The remaining mixed-integer problem, $dano3mip$, is a multicommodity fixed charge network flow model for which solving a single linear programming relaxation is already very time-consuming [4].

These results show that even though Branch-and-Cut is in most cases 10 to 20 % slower than Cut-and-Branch, it seems interesting for hard problems ($p2756$, $misc07$) and, what is more important, it guarantees a significant reduction of the number of nodes, therefore allowing one to solve problems for which otherwise memory would be a limiting factor ($vpm2$).

Finally we should note that specialized strategies permit $bc-opt$ to solve most instances in Table 3 more rapidly, and to find the optimal solutions (but without a proof of optimality) of most of the instances in Table 4.

7 Conclusions

The results in this paper show that for many mixed integer problems, a combination of new separation routines and a branch-and-cut system permit us to now solve a large number of instances within a reasonable time. For a few of the more difficult instances adding cuts just at the top node (cut-and-branch) is insufficient. This contrasts with the observation that on the easier problems branch-and-cut is often slower than cut-and-branch. The extension of knapsack routines to include continuous variables seriously enlarges the range of problems for which cuts are generated. This idea has been taken further recently in [16].

One observation from this work is that progress in solving problems by cutting planes can be of at least three different types. Whereas most research to date has concentrated on either

i) Finding new valid inequalities and a separation routine for an existing canonical structure, or

instance	Class	m	B	I	C	LP	XLP	BLB	BIP	$\frac{BIP-BLB}{BIP}$	IP	Type	Limit
10teams	SC	211	1600	0	0	917	917	917.99	924	0.6%	924		TIME
harp2	D	102	1374	0	0	-74325169	-74166793	-74028979	-73766390	0.35%	-73899798*	BK	NODE
arki001		767	387	96	477	7579599	7579959	7580069	7582827	0.036%	7580813	IK,GM	TIME
bell5	FN	86	29	28	44	8608417	8937853	8942489	8997480	0.61%	8966406	IK,GM	NODE
danoint	FN	601	56	0	401	62.63	62.66	63.10	65.66	3.89%	65.66	FC,KC	TIME
gesa2_o	EG	1201	384	336	456	25476489	25680201	25769865	25782082	0.047%	25779856	KC,GM	NODE
pk1	D	46	55	0	31	0	0	4.97	11	54.81%	11	FC,KC,GM	NODE
mod011	D	1469	96	0	6704	-62081950	-62053347	-55121821	-54558535	1.02%	-54558535	FC	TIME
noswot	D	182	80	20	25	-43	-43	-43	-40	6.97%	-43	IK,FC,KC,GM	NODE
rout	D	291	300	15	240	981	982	1012	1083	6.55%	1077	IK,KC	NODE

Table 4: Hard problems

ii) Finding valid inequalities and a separation routine for a new canonical structure which must then be detected via a model interface,
the progress reported here has been largely due to
iii) Applying existing separation routines for a canonical structure via a new interface.

Many options remain to be tested. The default used in all the results presented here has been to treat cuts in the tree as local. As far as we know, all systems developed to date have used globally valid cuts. In comparing local and global cuts on a small number of problems, we have not observed significant differences, but further experimentation is needed.

Another question concerns branching strategies. As in [23], limited experimentation suggests that, when the cuts added are effective, a best bound solution strategy should be adopted for at least 100 or 200 nodes, before possibly returning to a default branching strategy.

Acknowledgement. We are grateful to G. Belvaux to his programming of the extension to the path routines, to Y. Pochet for many helpful comments and to the partners in the PAMIPS project for their collaboration and help.

References

[1] E. Balas, S. Ceria and G. Cornuéjols, Mixed 0–1 Programming by Lift-and-Project in a Branch-and-cut Framework, *Management Science* **42**, (1996) 1229-1246.

[2] E. Balas, S. Ceria, G. Cornuéjols and N. Natraj, Gomory Cuts Revisited, *Operations Research Letters* **19**, (1996) 1-9. .

[3] F. Barahona, M. Groetschel, M. Juenger and G. Reinelt, An application of combinatorial optimization to statistical physics and circuit layout design, *Operations Research* **36** (1988) 493-513.

[4] D. Bienstock, and O. Günlük, Computational Experience with a Difficult Mixed Integer Multicommodity Flow Problem, *Mathematical Programming* **68**, (1995) 213-237.

[5] R.E. Bixby, S. Ceria, C.M. McZeal and M.W.P. Savelsbergh, An updated Mixed Integer Programming Library: MIPLIB

3.0, text and problems available at { *http://www.caam.rice.edu/~bixby/miplib/miplib.html* }

[6] S. Ceria, C. Cordier, H. Marchand and L.A. Wolsey, Cutting Planes for Integer Programs with General Integer Variables, CORE DP9575, Université Catholique de Louvain, Louvain-la-Neuve (1995).

[7] W. Cook, L. Lovasz and P. Seymour, eds., Combinatorial Optimization, DIMACS Series in Discrete Mathematics and Computer Science, AMS 1995

[8] H. Crowder, E.L. Johnson and M.W. Padberg, Solving Large Scale Zero-One Linear Programming Problems, *Operations Research* **31**, (1983)803-834.

[9] R. E. Gomory, An algorithm for the mixed integer problem, RM-2597, The Rand Corporation (1960).

[10] M. Groetschel, On the symmetric travelling salesman problem: solution of a 120 city problem, *Mathematical Programming Study* **12** (1980) 61-77.

[11] M. Groetschel, C.L. Monma and M. Stoer, Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints, *Operations Research* **40**, (1992) 309-330.

[12] Z.Gu, G.L. Nemhauser and M.W.P. Savelsbergh, Lifted Cover Inequalities for 0-1 Integer Programs: Computation, School of Industrial and Systems Engineering, Georgia Institute of Technology, revised August 1995.

[13] Z.Gu, G.L. Nemhauser and M.W.P. Savelsbergh, Lifted Flow Cover Inequalities for Mixed 0-1 Integer Programs, LEC-96-05, School of Industrial and Systems Engineering, Georgia Institute of Technology (1996).

[14] E.L. Johnson and M.W. Padberg, A Note on the Knapsack Problem with Special Ordered Sets, *Operations Research Letters* **1**, (1981) 18-22.

[15] M. Juenger, G. Reinelt and S. Thienel, Practical problem solving with cutting plane algorithms in combinatorial optimization, 11-152 in [7].

[16] H. Marchand and L.A. Wolsey, The 0-1 knapsack problem with a single continuous variable, CORE Discussion Paper 9720, Louvain-la-Neuve, March 1997

[17] G.L. Nemhauser and L.A. Wolsey, Integer and Combinatorial Optimization, Wiley (1988).

[18] M.W. Padberg and S. Hong, On the symmetric traveling salesman problem: a computational Study, *Mathematical Programming Study* **12** (1980) 78-107.

[19] M.W. Padberg, T.J. Van Roy and L.A. Wolsey, Valid Linear Inequalities for Fixed Charge Problems, *Operations Research* **33**, (1985) 842-861.

[20] Pamips, Esprit Project 8755, Public Report Ref. DR4.3.5/I, 31/1/95.

[21] Y. Pochet, Valid inequalities and separation for capacitated economic lot-sizing, *Operations Research Letteres* **7**, (1988) 109-116.

[22] M.W.P Savelsbergh and G.L. Nemhauser, Functional description of MINTO, a Mixed INTeger Optimizer, Report COC-91-03A, Georgia Institute of Technology, Atlanta, Georgia (1993).

[23] T.J. Van Roy and L.A. Wolsey, Solving Mixed 0-1 Problems by Automatic Reformulation, *Operations Research* **35**, 45-57 (1987).

[24] L.A. Wolsey, Valid inequalities for mixed integer programs with generalised and variable upper bound constraints, *Discrete Applied Mathematics* **25**, (1990) 251-261.

[25] XPRESS-MP Optimisation Subroutine Library, Reference Manual, Release 9, Dash Associates, Blisworth House, Blisworth, Northants NN7 3BX, UK.