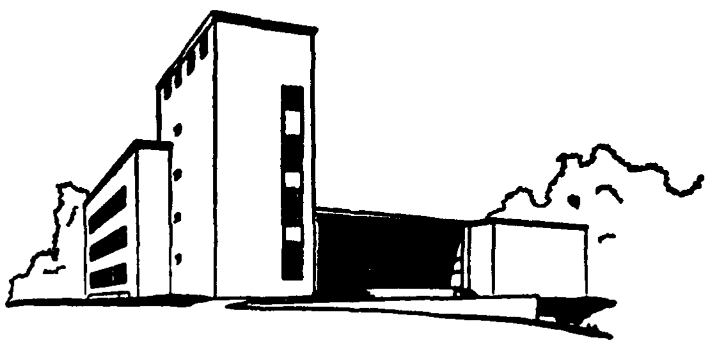
A LIFT-AND-PROJECT CUTTING PLANE ALGORITHM FOR MIXED 0-1 PROGRAMS

Egon Balas
Sebastián Ceria
and
Gérard Cornuéjols

Carnegie Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION
WILLIAM LARIMER MELLON, FOUNDER



1

Management Science Research Report No. #576

A LIFT-AND-PROJECT CUTTING PLANE ALGORITHM FOR MIXED 0-1 PROGRAMS

Egon Balas
Sebastián Ceria
and
Gérard Cornuéjols

October 1991

This research was supported in part by the National Science Foundation, Grant #DDM-8901495 and the Office of Naval Research through Contract N00014-85-K-0198.

Management Science Research Group
Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We propose a cutting plane algorithm for mixed 0-1 programs based on a family of polyhedra which strengthen the usual LP relaxation. We show how to generate a facet of a polyhedron in this family which is most violated by the current fractional point. This cut is found through the solution of a linear program that has about twice the size of the usual LP relaxation. A lifting step is used to reduce the size of the LP's needed to generate the cuts. An additional strengthening step suggested by Balas and Jeroslow is then applied. We report our computational experience with a preliminary version of the algorithm. This approach is related to the work of Balas on disjunctive programming, the matrix cut relaxations of Lovász and Schrijver and the hierarchy of relaxations of Sherali and Adams.

Key Words: Cutting planes, projection, mixed 0-1 programming, disjunctive programming.

1 Introduction

More than thirty years have elapsed since the emergence of cutting plane algorithms for mixed integer programming, but branch and bound is still the prevalent way to handle such problems. In the last 10-15 years there has been considerable progress in using combinatorial cutting planes for certain classes of pure integer programming problems, such as the symmetric traveling salesman problem, in combination with branch and bound (see, for instance [PR87]). The success of this approach, known as branch and cut, can be largely attributed to the fact that the combinatorial cutting planes used are often facets of the underlying integer polyhedron. For a mixed integer program, or for that matter a general pure integer program, facets for the integer polyhedron are not easy to obtain. For pure integer programs, one way to generate deep cuts is to use facets of the knapsack problems obtained by considering each constraint separately. This approach was applied successfully by Crowder, Johnson and Padberg [CJP83] to pure 0-1 programs without special structure. A similar idea was used by Van Roy and Wolsey [VW87] for mixed 0-1 programs.

Another way of *strengthening the* linear programming relaxation of an integer program is to lift the problem into a higher dimensional space, where a more convenient formulation may give a tighter relaxation. One then has a choice between working with this tighter relaxation in the higher dimensional space, or projecting it back onto the original space. In this latter case, the whole procedure can be viewed as a method for generating cutting planes in the original space.

One such procedure was recently proposed by Lovász and Schrijver [LS89] for 0-1 programs. The higher dimensional space they use is obtained by multiplying every inequality by every 0-1 variable and its complement in turn, then linearizing the resulting system of quadratic inequalities and finally projecting back the system onto the original space. The lifting phase of this procedure involves a squaring of the number of variables and an even steeper increase in the number of constraints, but iterating the lifting/projecting step a number of times equal to the number of original 0-1 variables yields the convex hull of feasible 0-1 points.

A similar lifting/projecting procedure, which obtains the integer hull in a non-iterative fashion through simultaneous multiplication of the original constraint set by all the 0-1 variables and their complements followed by projection, had been proposed by Sherali and Adams [SA88, SA89].

In this paper we propose a lifting/projecting procedure where the original constraint set is multiplied by a single 0-1 variable and its complement before projecting back onto the original space. The lifting phase of our procedure involves only a doubling rather than a squaring of the number of variables and constraints, nevertheless iterating the lifting/projecting step as many times as the original number of 0-1 variables yields the convex hull of feasible 0-1 points, as in the Lovász-Schrijver approach.

We then show that our iterated procedure is equivalent to the sequential convexification procedure for facial disjunctive programs (of which mixed 0-1 programs are a special case), introduced by Balas [B74b, B79] in the seventies. The new insight, which comes from rediscovering a previously known structure from an entirely different perspective, leads us to examine a class of finitely convergent cutting plane algorithms for mixed 0-1 programs based on the iterative lifting/projecting procedure outlined above. The cutting planes generated by the procedure are facets of the current projected polyhedron, and their derivation involves the solution of a linear program of roughly twice the size of the original problem. The objective function of this linear program is aimed at choosing among the members of the given family of cuts a deepest one, i.e, one that cuts off the optimal vertex of the current relaxation by more than any other member of the family.

The paper is organized as follows. Section 2 introduces the theory behind our approach. Section 2.1 states our lifting/projecting procedure and gives its main properties. Section 2.2 compares this procedure with the Lovász and Schrijver construction. Section 2.3 sketches the Sherali-Adams results and relates them to ours. Section 2.4 shows the equivalence of our lifting/projecting procedure to the sequential convexification procedure for facial disjunctive sets. Finally, Section 2.5 applies our procedure to the stable set polytope to recover some of the well-known facet inducing inequalities.

Section 3 discusses a class of cutting plane algorithms based on the material of Section 2. Section 3.1 outlines the approach and discusses some of the issues and options that arise. Section 3.2 gives a finiteness proof for a specialized version of the algorithm. Section 3.3

shows how some cutting planes can be generated from the simplex tableau, and Section 3.4 discusses a lifting step used to reduce the size of the LP's needed to generate the cuts. Section 3.5 applies to our inequalities the strengthening procedure introduced by Balas and Jeroslow for disjunctive cuts.

Finally, Section 4 describes our preliminary computational experience with some versions of the algorithm discussed under Section 3.1. The preliminary computational experiments that we carried out indicate that, for some classes of problems, a relatively low number of iterations is needed to find the optimum or get close to it.

2 Projection and Convexification

Define

$$
\begin{aligned}
K &:= \{x \in \Re^n : Ax \geq b, x \geq 0, x_j \leq 1, j = 1, \ldots, p\} \\
&:= \{x \in \Re^n : \tilde{A}x \geq \tilde{b}\}
\end{aligned}
$$

and

$$K^0 := \{x \in K : x_j \in \{0,1\}, j = 1, \ldots, p\}.$$

K^0 is a mixed integer set with n variables, p of which are 0, 1 constrained. K is the standard linear relaxation. In this section we consider procedures that yield conv K^0 starting from K.

2.1 A sequential convexification procedure

0. *Select an index $j \in \{1, \ldots, p\}$.*

1. *Multiply $\tilde{A}x \geq \tilde{b}$ with $1 - x_j$ and x_j to obtain the nonlinear system*

$$
\begin{aligned}
(1 - x_j)(\tilde{A}x - \tilde{b}) &\geq 0 \\
x_j(\tilde{A}x - \tilde{b}) &\geq 0 \qquad (1)
\end{aligned}
$$

2. *Linearize* (1) *by substituting y_i for $x_i x_j, i = 1, \ldots, n, i \neq j$, and x_j for x_j^2. Call the polyhedron defined by the resulting system $M_j(K)$.*

3. *Project $M_j(K)$ onto the x-space by eliminating $y_i, i = 1, \ldots, n, i \neq j$. Call the resulting polyhedron $P_j(K)$.*

Note that, if the system defining K has m constraints and n variables, the system defining $M_j(K)$ has $2m$ constraints and $2n-1$ variables. It is clear that $K^0 \subseteq P_j(K)$ and that $P_j(K) \subseteq K$. In fact, we have:

Theorem 2.1 $P_j(K) = conv(K \cap \{x \in \Re^n : x_j \in \{0,1\}\})$.

Since any x that satisfies $\tilde{A}x \geq \tilde{b}$ and $0 \leq x_j \leq 1$ clearly satisfies both $(1-x_j)(\tilde{A}x - \tilde{b}) \geq 0$ and $x_j(\tilde{A}x - \tilde{b}) \geq 0$, the multiplications performed in Step 1 above are not responsible for tightening the constraints of K. Further, replacing $x_i x_j$ with y_i for all $i \neq j$ in Step 2 above cannot tighten those constraints either. Yet, unless the 0-1 constraint is redundant for variable j, the projection $P_j(K)$ of the set $M_j(K)$ resulting from Step 2 is strictly contained in K. The only operation that is "accountable" for this tightening is the replacement of the terms x_j^2 by x_j. Indeed, while this substitution does not eliminate any points for which $x_j \in \{0,1\}$, it does cut off points x with $0 < x_j < 1$.

For $t \geq 2$, define $P_{i_1,\ldots,i_t}(K) = P_{i_t}(P_{i_{t-1}} \ldots (P_{i_1}(K)) \ldots)$.

Theorem 2.2 *For any $t = 1, \ldots, p$,*

$$P_{1,\ldots,t}(K) = conv(K \cap \{x \in \Re^n : x_j \in \{0,1\}, j = 1, \ldots, t\}).$$

Corollary 2.3 $P_{1,\ldots,p}(K) = conv\, K^0$.

Proof of Theorem 2.1.

$(i) P_j(K) \subseteq \text{conv}(K \cap \{x : x_j \in \{0,1\}\})$.

First assume $K \cap \{x : x_j = 0\} = \emptyset$. Then $x_j - \epsilon \geq 0$ is valid for K for some $\epsilon > 0$. This implies that $(1-x_j)(x_j - \epsilon) \geq 0$ is satisfied by any x that satisfies (1). Replacing x_j^2 by x_j, it follows that $x_j \geq 1$ is valid for $M_j(K)$ and for $P_j(K)$. This, together with $P_j(K) \subseteq K$, implies $P_j(K) \subseteq K \cap \{x : x_j = 1\}$ and hence (i).

Similarly, if $K \cap \{x : x_j = 1\} = \emptyset$, $x_j \leq 0$ is valid for $P_j(K)$, and again (i) follows.

Assume now that $K \cap \{x : x_j = 0\} \neq \emptyset$ and $K \cap \{x : x_j = 1\} \neq \emptyset$, and let $\alpha x \geq \beta$ be a valid inequality for $\text{conv}(K \cap \{x : x_j \in \{0,1\}\})$. Since $\alpha x \geq \beta$ is valid for $K \cap \{x : x_j = 0\}$, there exists $\lambda \geq 0$ such that $\alpha x + \lambda x_j \geq \beta$ is valid for K. Furthermore, since $\alpha x \geq \beta$ is valid for $K \cap \{x : x_j = 1\}$ there exists some $\mu \geq 0$ such that $\alpha x + \mu(1 - x_j) \geq \beta$ is valid for K. Now since $\alpha x + \lambda x_j - \beta \geq 0$ and $\alpha x + \mu(1 - x_j) \geq \beta$ are valid for K, the inequalities $(1 - x_j)(\alpha x + \lambda x_j - \beta) \geq 0$ and $x_j(\alpha x + \mu(1 - x_j) - \beta) \geq 0$ are satified by any x that satisfies (1). Adding these two inequalities yields $\alpha x + (\lambda + \mu)(x_j - x_j^2) - \beta \geq 0$ and, after setting $x_j^2 = x_j$, $\alpha x - \beta \geq 0$. Hence $\alpha x - \beta \geq 0$ is valid for $M_j(K)$ and for $P_j(K)$.

(ii) $\text{conv}(K \cap \{x : x_j \in \{0,1\}\}) \subseteq P_j(K)$.

To prove this, let $\alpha x \geq \beta$ be a valid linear inequality for $P_j(K)$. Then $\alpha x \geq \beta$ is dominated by some inequality of the form

$$u(\tilde{A}x - \tilde{b})(1 - x_j) + v(\tilde{A}x - \tilde{b})x_j \geq 0 \tag{2}$$

where $u, v \geq 0$ are row vectors such that all terms $x_i x_j$, $i \neq j$, cancel out, and where x_j is to be substituted everywhere for x_j^2. But any such inequality becomes $u(\tilde{A}x - \tilde{b}) \geq 0$ if $x_j = 0$, and $v(\tilde{A}x - \tilde{b}) \geq 0$ if $x_j = 1$. Thus (2) is valid for $K \cap \{x : x_j = 0\}$ and for $K \cap \{x : x_j = 1\}$; i.e., (2) and hence $\alpha x \geq \beta$ is valid for $\text{conv}(K \cap \{x : x_j \in \{0,1\}\})$. ∎

Proof of Theorem 2.2. For $t \in \{1, \ldots, p\}$, let $F_t := \{x : x_j \in \{0,1\} \text{ for } j = 1, \ldots, t\}$. We use induction on t. For $t = 1$, the result follows from Theorem 2.1. Suppose the result holds for $t = 1, \ldots, q - 1$ and let $t = q$, $2 \leq q \leq p$. Then

$$P_{1,\ldots,q}(K) = P_q(\text{conv}(K \cap F_{q-1})) = \text{conv}(\text{conv}(K \cap F_{q-1}) \cap \{x : x_q \in \{0,1\}\})$$

where the first equation is implied by the induction hypothesis, while the second follows from Theorem 2.1. The last expression can be rewritten as

$$\text{conv}\left((\text{conv}(K \cap F_{q-1}) \cap \{x : x_q = 0\}) \cup (\text{conv}(K \cap F_{q-1}) \cap \{x : x_q = 1\})\right) \tag{3}$$

Now let $S \subseteq \Re^n$ and let $H := \{x \in \Re^n : \alpha x = \beta\}$ be a hyperplane such that $\alpha x \geq \beta$ for all $x \in S$. We make the following

Claim . $H \cap convS = conv(S \cap H)$.

Proof. $x \in H \cap \operatorname{conv} S$ if and only if $\alpha x = \beta$ and $x = \sum_{i \in T} y^i \lambda_i$ for some finite T and $y_i \in S, \lambda_i \geq 0, i \in T$, such that $\sum_{i \in T} \lambda_i = 1$, and $\alpha y^i \geq \beta$, $i \in T$. But $\alpha x = \beta$ and $\alpha y^i \geq \beta, i \in T$, imply $\alpha y^i = \beta, i \in T$, hence $x \in H \cap \operatorname{conv} S \Leftrightarrow x \in \operatorname{conv}(S \cap H)$. ∎

Applying this result to (3) and using the fact that for any $S, T \in \Re^n$, $\operatorname{conv}(\operatorname{conv} S \cup \operatorname{conv} T) = \operatorname{conv}(S \cup T)$ we obtain

$$\begin{aligned} P_{1,\ldots,t}(K) &= \operatorname{conv}(K \cap F_{q-1} \cap \{x : x_q = 0\}) \cup (K \cap F_{q-1} \cap \{x : x_q = 1\})) \\ &= \operatorname{conv}(K \cap F_{q-1} \cap \{x : x_q \in \{0,1\}\}) \\ &= \operatorname{conv}(K \cap F_q). \end{aligned}$$ ∎

Corollary 2.3 follows immediately from Theorem 2.2 upon substituting p for t. Another consequence of Theorem 2.2 is the following.

Corollary 2.4 *$P_i(P_j(K)) = P_j(P_i(K))$, for $i, j \in \{1, \ldots, p\}, i \neq j$.*

2.2 The Lovász-Schrijver Construction

1. *Multiply $\tilde{A}x \geq \tilde{b}$ with x_j and $1 - x_j, j = 1, \ldots, p$, to obtain the nonlinear system*

$$\begin{aligned} (1 - x_1)(\tilde{A}x - \tilde{b}) &\geq 0 \\ x_1(\tilde{A}x - \tilde{b}) &\geq 0 \\ (1 - x_2)(\tilde{A}x - \tilde{b}) &\geq 0 \\ x_2(\tilde{A}x - \tilde{b}) &\geq 0 \\ \vdots \quad &\ \vdots \\ (1 - x_p)(\tilde{A}x - \tilde{b}) &\geq 0 \\ x_p(\tilde{A}x - \tilde{b}) &\geq 0 \end{aligned} \tag{4}$$

2. *Linearize (4) by substituting y_{ij} for $x_i x_j, i = 1, \ldots, n, j = 1, \ldots, p, i \neq j$ and x_j for x_j^2, $j = 1, \ldots, p$. Call the polyhedron defined from the resulting system $M(K)$.*

3. *Project $M(K)$ onto the x-space by eliminating $y_{ij}, i = 1, \ldots, n, j = 1, \ldots, p, i \neq j$. Call the resulting polyhedron $N(K)$.*

Note that, if the system defining K has m constraints and n variables, of which p are 0-1 constrained in K^0, the system defining $M(K)$ has $2pm$ constraints and $pn+n-p$ variables. Lovász and Schrijver have shown that $N(K)$ has the following properties:

Theorem 2.5 $N(K) \subseteq conv(K \cap \{x \in \Re^n : x_j \in \{0,1\}\})$, $j = 1,\ldots,p$.

Let $N^1(K) = N(K)$ and $N^t(K) = N(N^{t-1}(K))$, for $t \geq 2$.

Theorem 2.6 $N^p(K) = conv\, K^0$.

In other words, iterating the above procedure p times yields the integer hull.

Theorem 2.1 implies Theorem 2.5, since $N(K) \subseteq P_j(K)$.

Corollary 2.3 implies Theorem 2.6. Again, this follows from $N(K) \subseteq P_j(K), j = 1,\ldots,p$.

Note however, that the Lovász and Schrijver relaxation $N(K)$ is not only stronger than $P_j(K)$ for any j, but also stronger than $\cap_{j=1}^p P_j(K)$; the inclusion $N(K) \subseteq \cap_{j=1}^p P_j(K)$ can be strict.

2.3 The Sherali - Adams Construction

Somewhat earlier than Lovász and Schrijver, Sherali and Adams had proposed a similar convexification procedure [SA88].

Let K and K^0 be defined as above, and let $t \in \{1,\ldots,p\}$.

1. *Multiply* $\tilde{A}x \geq \tilde{b}$ *with every product of the form* $\left[\prod_{j\in J_1} x_j\right]\left[\prod_{j\in J_2}(1-x_j)\right]$, *where* J_1 *and* J_2 *are disjoint subsets of* $\{1,\ldots,p\}$ *such that* $|J_1 \cup J_2| = t$. *Call the resulting non-linear system* (NL_t).

2. *Linearize* (NL_t) *by (i) substituting* x_j *for* x_j^2; *and (ii) substituting a variable* w_J *for every product* $\prod_{j\in J} x_j$, *where* $J \subseteq \{1,\ldots,p\}$, *and* v_{Jk} *for every product* $x_k \prod_{j\in J} x_j$ *where* $J \subseteq \{1,\ldots,p\}$ *and* $k \in \{p+1,\ldots,n\}$. *Call the polyhedron defined by the resulting system* X_t.

3. *Project X_t onto the x-space by eliminating all w_J and v_{Jk}. Call the resulting polyhedron K_t.*

It is easy to see that $K^0 \subseteq K_p \subseteq \ldots \subset K_1 \subseteq K$. In addition, Sherali and Adams proved the following:

Theorem 2.7 [$SA88, SA89$] $K_p = convK^0$.

Next we prove a result which shows that Theorem 2.7 also follows from Theorem 2.2.

Theorem 2.8 *For $t = 1, \ldots, p$, $K_t \subseteq P_{1,\ldots,t}(K)$.*

Proof. Let $A^j x \geq b^j$ denote the linear system describing $P_{1,\ldots,j}(K)$, for $j = 1, \ldots, t$. Let $\alpha x \geq \beta$ be one of the inequalities defining $P_{1,\ldots,t}(K)$. Then $\alpha x \geq \beta$ can be obtained by taking a nonnegative linear combination of the inequalities $(1 - x_t)(A^{t-1}x - b^{t-1}) \geq 0$ and $x_t(A^{t-1}x - b^{t-1}) \geq 0$, with multipliers that eliminate the nonlinear products $x_i x_t, i \neq t$ and substituting x_t^2 by x_t. By the same argument every inequality of the system $A^{t-1}x - b^{t-1} \geq 0$ can be obtained by taking a nonnegative linear combination of the inequalities $(1 - x_{t-1})(A^{t-2}x - b^{t-2}) \geq 0$ and $x_{t-1}(A^{t-2}x - b^{t-2}) \geq 0$, with multipliers that eliminate all products $x_i x_{t-1}, i \neq t-1$ and setting $x_{t-1}^2 = x_{t-1}$. By inductively repeating this argument we can obtain $\alpha x \geq \beta$ in terms of the inequalities of (NL_t), by first substituting x_j by $x_j^2, j = 1, \ldots, t$, and then eliminating the remaining nonlinear terms using as multipliers the product of the multipliers used in each step of the induction. Therefore $\alpha x \geq \beta$ is valid for K_t and the result follows. ∎

Now Theorem 2.7 follows from Corollary 2.3 and Theorem 2.8. It also follows from Theorem 2.6 and a proposition in [LS89] that shows that $K_t \subseteq N^t(K)$.

2.4 The Connection with Disjunctive Programming

The results of Section 2.1 are closely related to results obtained earlier in the context of disjunctive programming, i.e. optimization over unions of polyhedra. The first of these is the following basic lifting theorem for unions of polyhedra:

Theorem 2.9 [*B74b, B85*] *Let $\Pi_i := \{x \in \Re^n : A^i x \geq b^i\}, i \in Q$ be a finite set of nonempty polyhedra. Then $conv(\cup_{i \in Q} \Pi_i)$ is the set of those $x \in \Re^n$ for which there exist vectors $(y^i, y_0^i), i \in Q$, such that*

$$\begin{array}{lllcl} x & -\sum_{i\in Q} y^i & & = & 0 \\ & A^i y^i & -b^i y_0^i & \geq & 0 \\ & & y_0^i & \geq & 0 \\ & & \sum_{i \in Q} y_0^i & = & 1 \end{array} \tag{5}$$

Theorem 2.9 assumes $\Pi_i \neq \emptyset, i \in Q$. If $\Pi_k = \emptyset$ for some $k \in Q$, the theorem is still valid [B85] if the following regularity condition holds:

$$A^k x \geq 0 \Rightarrow x = \sum y^i \text{ for some } i \in Q \backslash \{k\} \text{ such that } \Pi_i \neq \emptyset \text{ and } A^i y^i \geq 0.$$

Next we show that Theorem 2.9, specialized to the case when $|Q| = 2$ and

$$\Pi_1 := K \cap \{x : x_j = 0\}, \Pi_2 := K \cap \{x : x_j = 1\}$$

yields Theorem 2.1. Indeed in this case (5) becomes

$$\begin{array}{llllllcl} x & -z & & -y & & & = & 0 \\ & \tilde{A}z & -\tilde{b}z_0 & & & & \geq & 0 \\ & z_j & & & & & = & 0 \\ & & & \tilde{A}y & -\tilde{b}y_0 & & \geq & 0 \\ & & & y_j & -y_0 & & = & 0 \\ & & z_0 & & +y_0 & & = & 1 \\ & & & & z_0, y_0 & & \geq & 0 \end{array} \tag{6}$$

It is easy to see that the above regularity condition is satisfied for (6), since the matrices A^i associated with z and y are the same, namely $\tilde{A}$.

On the other hand, $P_j(K)$ is the projection onto the x-space of $M_j(K)$, the polytope whose defining system is obtained from (1) by setting $y_i := x_i x_j, i = 1, \ldots, n, y_j := x_j = x_j^2$. The result of these operations is:

$$\begin{array}{rcl} \tilde{A}x - \tilde{b} - \tilde{A}y + \tilde{b}x_j & \geq & 0 \\ \tilde{A}y - \tilde{b}x_j & \geq & 0 \end{array} \tag{7}$$

If we now define $y_0 := x_j, z := x - y, z_0 := 1 - x_j$, (7) can be rewritten as (6), where the equation $z_j = 0$ follows from $y_j = x_j, z_j = x_j - y_j$; and the equation $y_j = y_0$ follows from $y_j = x_j = y_0$.

Since Theorem 2.1 asserts that $\mathrm{conv}(\Pi_1 \cup \Pi_2)$ is the projection on the x-space of the polytope defined by (6), it is a specialization of Theorem 2.9 to this case.

An alternative characterization of the convex hull of a union of polyhedra, also obtained in the context of disjunctive programming, is contained in the following theorem, which will play an important role in the cutting plane algorithm of Section 3. We state the result as it applies to $P_j(K)$ $(= \mathrm{conv}(\Pi_1 \cup \Pi_2))$.

Theorem 2.10 [$B74b, B79$]

$$P_j(K) = \{x \in \Re^n : \alpha x \geq \beta \textit{ for all } (\alpha, \beta) \in P_j^*(K)\},$$

where $P_j^*(K)$ is the set of those $(\alpha, \beta) \in \Re^{n+1}$ for which there exist vectors $u, v \in \Re^{m+n+p}$ and $u_0, v_0 \in \Re$ satisfying:

$$\begin{array}{lllllcl}
\alpha & -u\tilde{A} & -u_0 e_j & & & = & 0 \\
\alpha & & & -v\tilde{A} & -v_0 e_j & = & 0 \\
& u\tilde{b} & & & & \geq & \beta \\
& & & v\tilde{b} & +v_0 & \geq & \beta \\
& & & & u, v & \geq & 0
\end{array} \tag{8}$$

where e_j is the j^{th} unit vector in $\Re^n$.

Further, if K is a full dimensional polyhedron and $\Pi_1 \neq \emptyset \neq \Pi_2$, there is a 1-1 correspondence between facets of $P_j(K)$ and extreme points of $P_j^*(K)_\beta$, the polyhedron obtained from the cone $P_j^*(K)$ by setting $\beta = 1$ or $\beta = -1$.

Next we turn to the sequential convexification theorem for facial disjunctive sets. If Π is a polyhedron containing the polyhedra $\Pi_i, i \in Q$, then the disjunctive set $S := \cup_{i \in Q} \Pi_i$ can be written in conjunctive normal form as

$$S = \{x \in \Pi : \bigvee_{k \in Q_h} d^k x \geq d_0^k, h = 1, \ldots, q\}, \tag{9}$$

where $|Q_h| = |Q|$ for all h, and each disjunction h contains exactly one inequality from the system defining each Π_i.

The disjunctive set S is called facial if each inequality $d^k x \geq d_0^k, k \in Q_h, h = 1, \ldots, q$, defines a face of Π.

Theorem 2.11 [B74b, B79] *Let S be defined by (9). Let $S_0 := \Pi$ and for $h = 1, \ldots, q$, let*

$$S_h := conv(S_{h-1} \cap \{\bigvee_{k \in Q_h} d^k x \geq d_0^k\}).$$

If S is facial, then $S_q = conv\, S$.

When S is of the form

$$K^0 := \{x \in K : x_h = 0 \vee x_h = 1, h = 1, \ldots, p\},$$

it is clearly facial, and thus Theorem 2.11 specializes to Theorem 2.2.

While faciality is a sufficient condition for the theorem to hold, it is not necessary. A necessary condition was given in [BTT89].

We will say that an inequality $\alpha x \geq \beta$, valid for K^0, has *disjunctive rank* r if r is the smallest integer such that there exists a subset $\{i_1, \ldots, i_r\}$ of $\{1, \ldots, p\}$, such that $\alpha x \geq \beta$ is valid for $P_{i_1, \ldots, i_r}(K)$.

In the case of a pure 0-1 programming problem it is interesting to compare the disjunctive rank of an inequality with its Chvátal rank.

Let $K = \{x \in \Re^2 : -2x_1 + x_2 \leq 0, 2x_1 + x_2 \leq 2, 0 \leq x_j \leq 1, j = 1, 2\}$, with $p = 2$.

It follows from Theorem 2.1 that the inequality $x_2 \leq 0$ has disjunctive rank 1, but it is easy to verify that it has Chvátal rank 2.

Since the disjunctive rank, according to Theorem 2.2, never exceeds the number of variables, whereas no such upper bound is known for the Chvátal rank, one might expect the disjunctive rank to always be less than or equal to the Chvátal rank. This, however, is not the case, as will be illustrated in the next section.

2.5 Application to the Stable Set Polytope

A stable set (independent set, vertex packing) in a graph $G := (V, E)$ is a subset $S \subseteq V$ such that no two vertices of S are adjacent. The stable set polytope is the convex hull of the incidence vectors of stable sets in G:

$$S(G) := \text{conv}\{x \in \{0,1\}^n : x_i + x_j \leq 1, \forall (i,j) \in E\}$$

The linear programming relaxation of $S(G)$

$$FS(G) := \{x \in \Re^n_+ : x_i + x_j \leq 1, \forall (i,j) \in E\},$$

sometimes called the fractional stable set polytope, strictly contains $S(G)$ whenever G is not bipartite. Facets of $S(G)$ include the odd hole and clique inequalities. For $W \subseteq V, G \backslash W$ denotes the subgraph of G induced by $V \backslash W$. For $v \in V, \Gamma(v)$ denotes the set of vertices adjacent to v. Given G, *deleting* v and *contracting* v are defined as replacing G with $G \backslash \{v\}$ and with $G \backslash (\{v\} \cup \Gamma(v))$, respectively. Clearly these operations correspond to setting $x_v = 0$ and $x_v = 1$ in $S(G)$.

If $ax \leq b$ is a valid inequality for $S(G)$ we say that the inequalities

$$\sum_{j \in V \backslash \{v\}} a_j x_j \leq b \quad \text{and} \quad \sum_{j \in V \backslash (\{v\} \cup \Gamma(v))} a_j x_j \leq b - a_v$$

are obtained from $ax \leq b$ by the deletion and contraction of v, respectively.

The following two properties are shown by Lovász and Schrijver [LS89] to hold for the set $N(FS(G))$. Here we show them to also hold for the larger sets $P_j(FS(G))$.

Lemma 2.12 *If $ax \leq b$ is a valid inequality for $S(G)$ and there exists $j \in V$ such that the inequalities obtained from $ax \leq b$ by deletion of j and contraction of j are valid for $FS(G \backslash \{j\})$ and $FS(G \backslash (\{j\} \cup \Gamma(j)))$, respectively, then $ax \leq b$ is valid for $P_j(FS(G))$.*

Proof. Follows from the fact that (Theorem 2.1) $P_j(FS(G)) = \text{conv}(FS(G) \cap \{x : x_j \in \{0,1\}\})$ ∎

For the purpose of this discussion, an odd hole in G is defined as a chordless cycle of odd length (i.e. triangles are included).

Theorem 2.13 *Let $C \subseteq V$ induce an odd hole in G. Then the odd hole inequality*

$$\sum_{i \in C} x_i \leq \frac{|C|-1}{2}$$

is valid for $P_j(FS(G))$ for any $j \in C$.

Proof. Let $j \in C$. The inequalities obtained from the odd hole inequality by deleting and contracting j are valid for $FS(G \backslash \{j\})$ and $FS(G \backslash (\{j\} \cup \Gamma(j)))$ respectively, since the subgraphs of G induced by $C \backslash \{j\}$ and $C \backslash (\{j\} \cup \Gamma(j)))$ are both bipartite. Hence from Lemma 2.12, the odd hole inequality is valid for $P_j(FS(G))$. ∎

Thus the odd hole inequalities, which have Chvátal rank 1, also have disjunctive rank 1.

Corollary 2.14 *All odd hole inequalities are valid for the polytope*

$$P(FS(G)) := \bigcap_{j \in V} P_j(FS(G)).$$

Proof. Follows from Theorem 2.13. ∎

As mentioned in Section 2.2, the Lovász and Schrijver relaxation $N(K)$ can be stronger than the intersection of the relaxations $P_j(K)$. In our case, this would imply that the inclusion $N(FS(G)) \subseteq P(FS(G))$ is strict. This, however, is not the case; i.e. for the stable set problem the two relaxations are the same. Lovász and Schrijver [LS89] have characterized $N(FS(G))$ as precisely the polytope defined by the inequalities defining $FS(G)$ and the odd hole inequalities.

Proposition 2.15 $P(FS(G)) = N(FS(G))$.

Proof. The inclusion $N(FS(G)) \subseteq P(FS(G))$ has already been discussed. The inclusion $P(FS(G)) \subseteq N(FS(G))$ follows from the fact that $P(FS(G))$ satisfies all the odd hole inequalities. ∎

Another well known class of valid inequalities for the stable set polytope is that associated with cliques, i.e., the sets of pairwise adjacent vertices.

The clique inequality

$$\sum_{j \in K} x_j \leq 1$$

where $K \subseteq V$ is a clique of G, is known to induce a facet of $S(G)$ if and only if the clique K is (inclusion-) maximal. Clique inequalities are known to have Chvátal rank $\lceil log_2 |K| \rceil$.

Theorem 2.16 *For any clique K, the clique inequality $\sum_{i \in K} x_i \leq 1$ has disjunctive rank $|K| - 2$.*

Proof. First we show by induction that the rank of $\sum_{i \in K} x_i \leq 1$ is at most $|K| - 2$. For $|K| = 3$ the result follows from Lemma 2.13. Now suppose the result holds for every clique K such that $|K| \leq k$ and let K' be a clique of size $k+1$. Let $j \in K'$ and $K = K' \backslash \{j\}$. By the inductive hypothesis, the inequality $\sum_{i \in K} x_i \leq 1$, obtained from $\sum_{i \in K'} x_i \leq 1$ by deletion of j, has rank at most k, i.e. there exists $\{i_1, \ldots, i_k\} \subseteq V$ such that the inequality is valid for $P_{i_1,\ldots,i_k}(FS(G))$. Also, the inequality obtained from $\sum_{i \in K'} x_i \leq 1$ by contraction of j is $0 \leq 0$ and also valid for $P_{i_1,\ldots,i_k}(FS(G))$. Hence, by Theorem 2.1, $\sum_{i \in K'} x_i \leq 1$ is valid for $P_{i_1,\ldots,i_k,j}(FS(G))$ and hence the disjunctive rank of $\sum_{i \in K'} x_i \leq 1$ is at most $k + 1 = |K'|$.

To prove that the disjunctive rank of $\sum_{i \in K} x_i \leq 1$ is exactly $|K| - 2$, suppose it is $s \leq |K| - 3$. Then, there exists $\{i_1, \ldots, i_s\} \subseteq V$ such that the inequality is valid for $P_{i_1,\ldots,i_s}(FS(G))$. It then follows from Theorem 2.1 that, whether $i_s \in K$ or $i_s \notin K$, the inequality

$$\sum_{i \in K \backslash \{i_s\}} x_i \leq 1$$

is valid for $P_{i_1,\ldots,i_{s-1}}(FS(G \backslash \{i_s\}))$. Applying this reasoning recursively to $K \backslash \{i_s\}$, $K \backslash \{i_s, i_{s-1}\}$, etc., the inequality

$$\sum_{i \in K \backslash \{i_1,\ldots,i_s\}} x_i \leq 1$$

with $|K \backslash \{i_1, \ldots, i_s\}| \geq 3$ is valid for $FS(G \backslash \{i_1, \ldots, i_s\})$, a contradiction. ∎

Thus the disjunctive rank of cliques inequalities for cliques of size ≥ 5, is larger than their Chvátal rank.

3 Some Cutting Plane Algorithms

3.1 The General Procedure

In this section we discuss cutting plane algorithms for mixed 0-1 programs based on the sequential convexification procedure of Section 2.1. In particular, we address the problem

$$min\{cx : x \in K, x_j \in \{0,1\}, j = 1,\ldots,p\}, \qquad (MIP)$$

where, as before, $K = \{x \in \Re^n : \tilde{A}x \geq \tilde{b}\}$.

We wish to use facets of $P_j(K)$ as cutting planes. For this purpose we will generate inequalities $\alpha x \geq \beta$ such that (α, β) is an extreme ray of the cone $P_j^*(K)$ of Theorem 2.10. This can be done by solving a linear program of the form

$$max\{a\alpha + b\beta : (\alpha, \beta) \in P_j^*(K) \cap S\}, \tag{10}$$

where $(a, b) \in \Re^{n+1}$ is a vector that determines the direction of the cut, $P_j^*(K)$ is the polyhedral cone defined by (8), while S is a "normalization" set defined by one or more constraints meant to truncate the cone $P_j^*(K)$.

The general outline of such a procedure is as follows:

0. $t := 1. K^1 := K = \{x \in \Re^n : \tilde{A}x \geq \tilde{b}\}$.

1. *Find* $cx^t := min\{cx : x \in K^t\}$.
If $x_j^t \in \{0,1\}$ *for* $j = 1,\ldots,p$, *stop.*

2. *For* $j \in \{1,\ldots,p\}$ *such that* $0 < x_j^t < 1$, *find*

$$a^t\alpha^j + b^t\beta^j := max\{a^t\alpha + b^t\beta : (\alpha,\beta) \in P_j^*(K^t) \cap S\}.$$

3. *Define* K^{t+1} *by adding to the constraints of* K^t *the cuts* $\alpha^j x \geq \beta^j$ *generated in Step 2 (and perhaps removing some cuts added earlier).*

4. *Set* $t := t + 1$ *and go to 1.*

There are several options for choosing the set S and the vector (a^t, b^t) in Step 2.

Normalization 1 : We say that $(\alpha, \beta) \in P_j^*(K^t)$ defines a deepest cut if it maximizes the (Euclidean) distance between x^t and the hyperplane $\alpha x = \beta$.

Maximizing the distance between x^t and $\alpha x = \beta$ is the same as maximizing the distance between x^t and its orthogonal projection on $\alpha x = \beta$, which is $\hat{x} = x^t - \lambda\alpha$ for some $\lambda > 0$. Since $\alpha x = \beta$ is the same as $\lambda \alpha x = \lambda \beta$ w.l.o.g. we can take $\lambda = 1$. Thus $\hat{x} - x^t = \alpha$. Furthermore, $\beta - \alpha x^t = \alpha\hat{x} - \alpha x^t = \alpha\alpha$. Thus a deepest cut is obtained for the vector $(a^t, b^t) = (-x^t, 1)$ and the set $S := \{(\alpha, \beta) : \beta - \alpha x^t = \alpha\alpha\}$. However, the resulting problem (10) is not a linear program, as the equation defining S is quadratic. Thus we are led to consider some alternatives to this "optimal" normalization.

We continue to use $(a^t, b^t) = (-x^t, 1)$ so that the objective in (10) remains to maximize the amount by which the point x^t violates the cut $\alpha x \geq \beta$. We consider the following options for S :

Normalization 2 : One may simply require that $\beta = 1$ or $\beta = -1$. In many problems it is easy to decide whether one should want a cut $\alpha x \geq \beta$ with $\beta > 0$ or $\beta < 0$. One advantage of this approach is that the linear program (10) to be solved in Step 2 can be reduced by eliminating the variables $\alpha_i, i = 1, \ldots, n$ from the system (8) defining $P_j^*(K)$. Thus for $\beta = 1$ or $\beta = -1$, (10) becomes

$$
\begin{array}{lllll}
\text{Min } (u\tilde{A} + u_0 e_j)x^t & & & & \\
\text{subject to} & & & & \\
u\tilde{A} \quad -v\tilde{A} \quad +(u_0 \quad -v_0)e_j & = & 0 & & \\
u\tilde{b} & \geq & \beta & & \\
\qquad\quad v\tilde{b} \qquad\qquad +v_0 & \geq & \beta & & \\
\qquad\qquad\qquad\qquad u, v & \geq & 0 & &
\end{array}
\tag{11}
$$

On the other hand, the drawback of this formulation is that the optimal solution sought may not exist. Indeed, it is known [B74b, B79] that the linear program (11) has a minimum if and only if $\lambda x^t \in P_j(K)$ for some $\lambda > 0$. For important classes of problems this condition is always satisfied. But for others it is not, and the task of generating a strong cut using (11) becomes cumbersome.

The next two normalizations are aimed at guaranteeing the existence of a finite optimum in (10).

Normalization 3 : We require that $\|\alpha\|_\infty \leq 1$, by defining $S := \{(\alpha, \beta) : -1 \leq \alpha_i \leq 1, i = 1, \ldots, n\}$.

Normalization 4 : We require that $\|\alpha\|_1 \leq 1$, by defining $S := \{(\alpha, \beta) : \sum_{i=1}^n |\alpha_i| \leq 1\}$. The absolute value constraint used here can be linearized by introducing $2n$ new variables α_i^+, α_i^-, $i = 1, \ldots, n$, writing

$$S := \{(\alpha, \beta) : \alpha = \alpha^+ - \alpha^-; \alpha^+, \alpha^- \geq 0; \sum_{i=1}^n (\alpha_i^+ + \alpha_i^-) \leq 1\},$$

and eliminating α.

3.2 Finite termination

While there are several finite algorithms for pure integer programming, finiteness is much harder to achieve in the mixed integer case. Gomory, who proved that his cutting plane algorithms for pure integer programming are finitely convergent, was able to prove finite convergence of his mixed integer programming algorithm [G60] only for the case when the objective function is itself integer constrained. However, this assumption cannot be made without loss of generality; and if the assumption is removed, Gomory's algorithm is not finite, as shown by White [W61] (see [S75]).

The first finite mixed integer programming cutting plane algorithm was developed by Jeroslow [J80] in the context of facial disjunctive programming, of which (MIP) is a special case. To guarantee finiteness, Jeroslow uses a game theoretic framework for choosing the cuts to generate. His convergence proof is based on the sequential convexification theorem of Section 2.4, and uses the fact that every cutting plane generated in the algorithm is a facet of some member of a finite family of polyhedra.

In this section we give a finiteness proof for a particular version of our procedure. Although our proof is simpler than that of Jeroslow, it uses the same basic idea.

The general procedure outlined in Section 3.1 need not be finitely convergent. To insure finite convergence, additional details have to be specified. We start with some notation.

In a general iteration of the procedure of Section 3.1, the current polyhedron K^t is defined by the inequalities of $\tilde{A}x \geq \tilde{b}$ together with a set of cuts. For $j \in \{1,\ldots,p\}$, a cut that appears in the definition of K^t is called a *j-cut* if it was generated as a cut for some $P_j(.)$, i.e., from the disjunction $x_j = 0 \vee x_j = 1$. Let K_j^t be the polyhedron defined by $\tilde{A}x \geq \tilde{b}$ and all i-cuts for $i = 1,\ldots,j$, with $K_0^t = K$. Note that with this notation $K_p^t = K^t$ and $K_j^t \subseteq K_i^t$ for all i,j such that $0 \leq i < j \leq p$.

Let $P_j^*(K)_S := P_j^*(K) \cap S$, where S is the set defined in Normalization 3 or 4.

Specialized Cutting Plane Algorithm

0. $t := 1, K^1 := K = \{x \in \Re^n : \tilde{A}x \geq \tilde{b}\}$.

1. *Find* $cx^t := min\{cx : x \in K^t\}$.
If $x_j^t \in \{0,1\}$ *for* $j = 1,\ldots,p$, *stop.*

2. *Let* $j \in \{1,\ldots,p\}$ *be the largest index such that* $0 < x_j^t < 1$. *Generate a j-cut* $\alpha^j x \geq \beta^j$ *by solving*

$$max\{\beta - \alpha x^t : (\alpha,\beta) \in P_j^*(K_{j-1}^t)_S\}.$$

3. *Define* K^{t+1} *by adding the j-cut* $\alpha^j x \geq \beta^j$ *to the constraints of* K^t.

4. *Set* $t := t+1$ *and go to 1.*

Theorem 3.1 *The Specialized Cutting Plane Algorithm finds an optimal solution to (MIP) in finitely many iterations.*

Proof. We need to prove two claims:

(i) The inequality $\alpha^j x \geq \beta^j$ generated in Step 2 cuts off x^t.

To prove this, we show that x^t, an extreme point of $K^t (= K_p^t)$, is also an extreme point of K_j^t. This is trivially true if $j = p$, so assume that $j < p$. Since $x_p^t \in \{0,1\}$, it follows that $x_p^t \in K_{p-1}^t \cap \{x \in \Re^n : x_p = x_p^t\} \subseteq P_p(K_{p-1}^t) \subseteq K_p^t$ and therefore x^t is an extreme point of $K_{p-1}^t \cap \{x \in \Re^n : x_p = x_p^t\}$, hence also of K_{p-1}^t (since $K_{p-1}^t \cap \{x \in \Re^n : x_p = x_p^t\}$ is a face of K^t_{p-1}). By induction, since $x^t_k \in \{0,1\}$ for $k = p, p-1, \dots, j+1$ it follows that x^t is an extreme point of $K^t_{p-1}, K^t_{p-2}, \dots, K^t_j$.

Next we show that $x^t \notin P_j(K^t_{j-1})$. Since $P_j(K^t_{j-1}) \subseteq K^t_j$, if $x^t \in P_j(K^t_{j-1})$ then x^t is an extreme point of $P_j(K^t_{j-1})$. But all extreme points of $P_j(K^t_{j-1})$ have a j^{th} component equal to 0 or 1, whereas $0 < x^t_j < 1$.

Since $x^t \notin P_j(K^t_{j-1})$, the inequality $\alpha^j x \geq \beta^j$ generated in Step 2 is violated by x^t.

(*ii*) For $j = 1, \dots, p$ the number of j-cuts generated by the algorithm is finite.

We prove this by induction. The statement is certainly true for $j = 1$ as every 1-cut generated corresponds to an extreme point of $P^*_1(K^t_0)_S = P^*_1(K)_S$, of which there are only finitely many, and as shown in Claim 1, every 1-cut generated cuts off some x^t that satisfies all 1-cuts generated earlier.

Suppose now that the statement is true for all $i = 1, \dots, j-1$ and let $i = j$. By the induction hypothesis, the set K^t_{j-1} is redefined in Step 3 of the algorithm (by addition of some i-cut for $i \in \{1, \dots, j-1\}$) only a finite number of times. Between any two such redefinitions, only a finite number of j-cuts can be generated, since each j-cut corresponds to an extreme point of $P^*_j(K^t_{j-1})_S$, of which there are only finitely many, and each j-cut cuts off some x^t which satisfies all j-cuts generated earlier. Hence only a finite number of j-cuts are generated during the entire algorithm, which completes the induction. ■

3.3 Cuts from the basis inverse

Let x^t be the current fractional solution in the t^{th} iteration of the cutting plane procedure, i.e. $cx^t = min\{cx : x \in K^t\}, K^t = \{x \in \Re^n : \tilde{A}^t x \geq \tilde{b}^t\}$. In general, for a cut $\alpha x \geq \beta$ in $P^*_j(K^t)$ defined by (8), the variables u_i and v_i can be strictly positive even if the corresponding constraint $\sum_{j=1}^n \tilde{a}^t_{ij} x_j \geq \tilde{b}^t_i$ is not satisfied as equality by x^t. In the case where we impose that the only u_i, v_i that are allowed to be strictly positive are those for which the slack corresponding to the constraint $\sum_{j=1}^n \tilde{a}^t_{ij} x_j \geq \tilde{b}^t_i$ is nonbasic, a cut can be obtained without having to solve a linear program, as shown below:

Let B^t be the matrix obtained from $\tilde{A}^t$ by keeping only the rows corresponding to constraints for which the associated slack variable is nonbasic. Let $C = \{x \in \Re^n : B^t x \geq d^t\}$ be the polyhedron defined by those inequlities of $\tilde{A}^t x \geq \tilde{b}^t$ corresponding to the rows of B^t. Then, by applying Theorem 2.10 to $P_j(C)$ and eliminating the variables $\alpha_i, i = 1, \ldots, n$, and β from the corresponding system (8) we get:

$$\begin{aligned} (u^B - v^B)B^t &= (v_0^B - u_0^B)e_j \\ (u^B - v^B)d^t - s_1 + s_2 &= v_0^B \\ u^B, v^B, s_1, s_2 &\geq 0 \end{aligned} \tag{12}$$

where s_1 and s_2 are the nonnegative slack variables corresponding to the inequalities $u\tilde{b} \geq \beta$ and $v\tilde{b} + v_0 \geq \beta$ in (8). Since the matrix B^t is invertible, given $v_0^B - u_0^B$, the vector $z^B = u^B - v^B$ is uniquely determined from the first n constraints of (12). Moreover, a basic solution to (12) satisfies $u_i^B.v_i^B = 0$ for all i, which implies that u^B and v^B are uniquely defined from z^B as its positive and negative parts respectively ($u^B = (z^B)^+, v^B = (z^B)^-$). This leads us to investigate the cuts that are obtained by taking as our normalization constraint the equality $v_0^B - u_0^B = 1$. These cuts are uniquely defined from (12) except for the values of s_1 and s_2. Whenever $s_1 = s_2 = 0$ we get the following simple formula for the cut $\alpha x \geq \beta$:

$$\begin{aligned} u^B &= (B_j^{t^{-1}})^+ \\ v^B &= (B_j^{t^{-1}})^- \\ \alpha_i &= (B_j^{t^{-1}})^+ B_i^t, \qquad i = 1, \ldots, n, i \neq j, \\ \alpha_j &= (B_j^{t^{-1}})^+ B_j^t + B_j^{t^{-1}} d^t - 1 \\ \beta &= (B_j^{t^{-1}})^+ d^t \end{aligned} \tag{13}$$

where $B_j^{t^{-1}}$ denotes the $j-th$ row of $B^{t^{-1}}$ and $B^{t^{-1}} = (B_j^{t^{-1}})^+ - (B_j^{t^{-1}})^-$. It is important to point out that the $j-th$ row of the matrix $B^{t^{-1}}$ is readily available from the simplex tableau defining the solution x^t.

Furthermore, the cut-hyperplane $\alpha x = \beta$ goes through the points where the boundary hyperplane $x_j = 0$ or $x_j = 1$ of the set $0 \leq x_j \leq 1$ is intersected by the rays of the cone with apex at x^t defined by the inequalities in C. Therefore the cut obtained this way is the intersection cut associated with the convex set $0 \leq x_j \leq 1$. Intersection cuts were introduced by Balas [B71], [B74a], see also Glover [Gl73].

We end this section with another normalization that has an interesting property. Let $K^t, \tilde{A}^t, \tilde{b}^t$, and x^t be defined as in the cutting plane procedure of Section 3.1.

Normalization 5 : We require that $v_0 - u_0 = 1$.

It can be shown that whenever the last two inequalities of (8) are required to hold with equality, the unique cut obtained by imposing this normalization is the intersection cut (13).

3.4 Cut Lifting

In this section we show that cutting planes with essentially the same properties as those derived by the procedure of Section 3.1 can be obtained from a smaller linear program than the one over $P_j^*(K^t)_S$, by working in the subspace defined by the fractional components of x^t, and then lifting the inequality into the original space. This is important not only because it is a computationally cheaper way of getting essentially the same cut, but also because in a branch and cut context it provides a way of generating cutting planes at one node of the search tree and lifting them into cutting planes valid at every node.

Let us consider the LP needed to generate a cut $\alpha x \geq \beta$ with Normalization 2, given by (11). Let $F = \{i \in \{1,\ldots,p\} : 0 < x_i^t < 1\} \cup \{i \in \{p+1,\ldots,n\} : x_i^t > 0\}$. W.l.o.g. we can assume that if $i \in \{1,\ldots,n\}$, $i \notin F$ then $x_i^t = 0$, since for those i such $x_i^t = 1$ the variable x_i can be complemented by changing the sign of $\tilde{A}_i$, the i^{th} column of $\tilde{A}$, and replacing $\tilde{b}$ by $\tilde{b} - \tilde{A}_i$.

Consider the problem derived from (11) by removing from $\tilde{A}$ all the columns corresponding to $\{1,\ldots,n\}\backslash F$. Let $\tilde{A}_k$ denote the k^{th} column of $\tilde{A}$, and let $\tilde{A}_k^F$ be the column obtained from $\tilde{A}_k$ by removing the components (all equal to 0) corresponding to the inequalities $x_h \geq 0, h \notin F$ and $-x_h \geq -1$ for $h \in \{1,\ldots,p\}\backslash F$. Then (11) can then be rewritten as:

$$
\begin{array}{lllllll}
\text{Min } \sum_{k\in F} u\tilde{A}_k^F x_k^t + u_0 x_j^t & & & & & & \\
\text{subject to} & & & & & & \\
u\tilde{A}_k^F & -v\tilde{A}_k^F & & & = & 0 & k \in F\backslash\{j\} \\
u\tilde{A}_j^F & -v\tilde{A}_j^F & +u_0 & -v_0 & = & 0 & \\
u\tilde{b} & & & & \geq & \beta & \\
& v\tilde{b} & & +v_0 & \geq & \beta & \\
& & & u, v & \geq & 0 &
\end{array} \tag{14}
$$

where $\beta = 1$ or $\beta = -1$.

Note that some of the variables and constraints in (11) are not present in (14). The constraints that are missing are:

$$u\tilde{A}_k - v\tilde{A}_k = 0, \quad k \notin F$$

while the missing variables are u_{m+i} for $i \notin F$ and u_{m+n+i} for $i \in \{1,\ldots,p\}, i \notin F$. Here $m+1,\ldots,m+n$ are the indices associated with the primal constraints $x_h \geq 0, h = 1,\ldots,n$ and $m+n+1,\ldots,m+n+p$ are those associated with the primal constraints $-x_h \geq -1$, $h = 1,\ldots,p$.

The following theorem shows how an optimal solution to (14) can be extended to an optimal solution to (11).

Theorem 3.2 *Let (u^F, v^F) be an optimal solution to (14). Extend (u^F, v^F) to a solution $(\tilde{u}, \tilde{v})$ of (11) by defining:*

$$
\begin{aligned}
\tilde{u}_i &= u_i^F \quad \textit{for } i = 1,\ldots,m, \\
\tilde{v}_i &= v_i^F \quad \textit{for } i = 1,\ldots,m, \\
\tilde{u}_{m+i} &= \begin{cases} (v^F - u^F)\tilde{A}_k^F & \textit{if } v^F\tilde{A}_k^F > u^F\tilde{A}_k^F \\ 0 & \textit{otherwise} \end{cases} \quad \textit{for } i \notin F, \\
\tilde{v}_{m+i} &= \begin{cases} (u^F - v^F)\tilde{A}_k^F & \textit{if } u^F\tilde{A}_k^F > v^F\tilde{A}_k^F \\ 0 & \textit{otherwise} \end{cases} \quad \textit{for } i \notin F, \\
\tilde{u}_{m+n+i} &= \tilde{v}_{m+n+i} = 0 \quad \textit{for } i \in \{1,\ldots,p\}, i \notin F.
\end{aligned}
$$

Then $(\tilde{u}, \tilde{v})$ is a basic feasible optimal solution to (11).

Proof. By construction $(\tilde{u}, \tilde{v})$ satisfies all the constraints of (11) missing from (14), while the remaining constraints are not affected. Thus $(\tilde{u}, \tilde{v})$ is feasible for (11).

To see that $(\tilde{u}, \tilde{v})$ is basic, note that (u^F, v^F) is a basic solution to (14), and $(\tilde{u}, \tilde{v})$ contains exactly one extra positive component for every constraint of (11) missing from (14). Furthermore note that the missing constraints are affinely independent from each other and from the constraints of (14), so their addition to (14) increases the rank of the latter exactly by their number. Thus $(\tilde{u}, \tilde{v})$ is basic for (11).

Now let z^F be the optimal solution to the dual of (14) associated with (u^F, v^F). Extend z^F to a feasible solution to the dual of (11), by setting to 0 all components associated with those constraints of (11) missing from (14). Then the reduced costs of (14) remain unchanged in (11). As to the reduced costs of the variables of (11) missing from (14), the situation is as follows:

For the variables $\tilde{u}_{m+n+i}$ for $i \in \{1, \ldots, p\}, i \notin F$, the reduced cost is

$$-x_k^t - (-z_{0u}^F) = z_{0u}^F \text{ (since } x_k^t = 0)$$

where z_{0u}^F is the dual variable associated with the next to last constraint of (11). Since that constraint is an inequality, $z_{0u}^F \geq 0$.

For the variables $\tilde{v}_{m+n+i}$ for $i \in \{1, \ldots, p\}, i \notin F$, the reduced cost is

$$0 - (-z_{0v}^F) = z_{0v}^F \geq 0,$$

where z_{0v}^F is the dual variable associated with the last inequality of (11).

For the variables $\tilde{u}_{m+i}$ for $i \in \{1, \ldots, n\}, i \notin F$, the reduced cost is

$$x_k^t - (0) = 0 \text{ (since } x_k^t = 0)$$

whereas that associated with the variables $\tilde{v}_{m+i}$ for $i \in \{1, \ldots, n\}, i \notin F$, the reduced cost is

$$0 - (0) = 0.$$

Thus all the reduced costs are nonnegative and hence $(\tilde{u}, \tilde{v})$ is optimal for (11). ■

Theorem 3.2 does not carry over directly to Normalizations 3 and 4. However, it can be proved for the following variants of these normalizations.

Normalization 3' : $S = \{(\alpha, \beta) : -1 \le \alpha_i \le 1, \text{ for } i \in F\}$.

Normalization 4' : $S = \{(\alpha, \beta) : \sum_{i \in F} |\alpha_i| \le 1\}$.

3.5 Cut Strengthening

The cutting planes $\alpha x \ge \beta$ in $P_j^*(K)$ for some $j \in \{1, \ldots, p\}$, can be strengthened by using the integrality condition on variables other than x_j, as shown by Balas and Jeroslow [BJ80].

Consider the system (8) of Section 2, defining $P_j^*(K)$. If we separate the non-negativity constraints $x \ge 0$ from the rest of the inequalities in $\tilde{A}x \ge \tilde{b}$, i.e. write the system without $x \ge 0$ as $\hat{A}x \ge \hat{b}, x \ge 0$, then (8) becomes

$$\begin{array}{cccccccc} \alpha & -\hat{u}\hat{A} & -u_0 e_j & & & \ge & 0 \\ \alpha & & & -\hat{v}\hat{A} & -v_0 e_j & \ge & 0 \\ & \hat{u}\hat{b} & & & & \ge & \beta \\ & & & \hat{v}\hat{b} & +v_0 & \ge & \beta \\ & & & & \hat{u}, \hat{v} & \ge & 0 \end{array} \tag{15}$$

and the coefficients of the cut $\alpha x \ge \beta$ can be written as

$$\begin{aligned} \alpha_k &= max\left\{\alpha_k^1, \alpha_k^2\right\}, k = 1, \ldots, n, \\ \beta &= min\{\beta_1, \beta_2\} \end{aligned}$$

where

$$\begin{array}{llll} \alpha_k^1 = \hat{u}\hat{A}_k, & \alpha_k^2 = \hat{v}\hat{A}_k, & \text{for } k = 1, \ldots, p, k \ne j; \\ \alpha_j^1 = \hat{u}\hat{A}_j - u_0, & \alpha_j^2 = \hat{v}\hat{A}_j + v_0 & \end{array}$$

(with $\hat{A}_k$ denoting the kth column of $\hat{A}$) and

$$\beta^1 = \hat{u}\hat{b}, \qquad \beta^2 = \hat{v}\hat{b} + v_0.$$

The cutting plane $\alpha x \ge \beta$, with $\beta \ne 0$, can be strengthened to $\gamma x \ge \frac{\beta}{|\beta|}$ where β is defined as above, while γ is given by:

$$\gamma_k = min\left\{\frac{1}{|\beta_1|}(\alpha_k^1 + u_0\lceil m_k \rceil), \frac{1}{|\beta_2|}(\alpha_k^2 - v_0\lfloor m_k \rfloor)\right\} \text{ for } k = 1, \ldots, p; \tag{16}$$

$$\gamma_k = max\left\{\frac{1}{|\beta|}\alpha_k^1, \frac{1}{|\beta|}\alpha_k^2\right\} \text{ for } k = p+1, \ldots, n;$$

with

$$m_k = \frac{\alpha_k^2 \, |\beta^1| - \alpha_k^1 \, |\beta^2|}{u_0 \, |\beta^2| + v_0 \, |\beta^1|} \tag{17}$$

For the validity of this strengthening see [BJ80, B79].

Note that if $m_k = 0$ the coefficient γ_k in the strengthened cut will be the same as in the unstrengthened cut. It can be shown that this is the case for all components k such that $x_k^i = 1$ when $\beta^1 = \beta^2$, a frequently occurring case in practice. By complementing the variable x_k and then applying the strengthening procedure, we may get $m_k \neq 0$. In our computational experiments we followed this practice.

We also note that, while a linear transformation of the system $\tilde{A}x \geq \tilde{b}$ leaves $K, P_j(K)$ and its facets unchanged, it can change the effect of the strengthening procedure.

Suppose, for instance, that instead of applying the disjunction $x_j = 0 \vee x_j = 1$ to the system $\tilde{A}x \geq \tilde{b}$, we first solve the linear program $\min\{cx : \tilde{A}x \geq \tilde{b}\}$ to obtain for $\tilde{A}x \geq \tilde{b}$ the expression:

$$\begin{aligned} x_i &= \bar{a}_{i0} + \sum_{k \in J} \bar{a}_{ik}(-x_k) \qquad \text{for } i \in I \\ x_k &\geq 0, \qquad k \in I \cup J, \end{aligned} \tag{18}$$

where I, J index the basic and nonbasic variables respectively, and then restate (18) as

$$\begin{aligned} -\sum_{k \in J} \bar{a}_{ik} x_k &\geq -\bar{a}_{i0}, \quad i \in I \\ x_k &\geq 0, \quad k \in J. \end{aligned} \tag{19}$$

Suppose also that $j \in I$; then the disjunction $x_j = 0 \vee x_j = 1$ becomes

$$\sum_{k \in J} \bar{a}_{jk} x_k \geq \bar{a}_{j0} \vee -\sum_{k \in J} \bar{a}_{jk} x_k \geq 1 - \bar{a}_{j0}. \tag{20}$$

Now the cuts defined from (19), (20) will be expressed in terms of the variables $x_k, k \in J$; and although these cuts are equivalent to the ones obtained from $\tilde{A}x \geq \tilde{b}, x_j = 0 \vee x_j = 1$, when it comes to the strengthening procedure, the outcome will in general be different in the two cases. First of all, in one case the strengthening procedure can be applied to all coefficients $\alpha_k, k = 1, \ldots, p$, whereas in the other case only to the coefficients α_k for $k \in \{1, \ldots, p\} \cap J$. Second, the different numerical values may yield different strengthening parameters m_k in the two cases.

One reason to look at the family of strengthened cuts derived from (19), (20) is that the Gomory cut for mixed integer programming [G60] is a member of this family. To see this, it suffices to look at the general form of the above cut and assign some special values to the multipliers used in its derivation. To do this, we have to restate the system (15) defining $P_j^*(K)$ in terms of the variables used in (19), (20):

$$\begin{array}{llllllll}
\alpha_k & +\sum_{i\in I} u_i^1 \bar{a}_{ik} & -u_0\bar{a}_{jk} & & & \geq & 0 & \text{for } k \in J \\
\alpha_k & & & +\sum_{i\in I} v_i^1 \bar{a}_{ik} & +v_0\bar{a}_{jk} & \geq & 0 & \text{for } k \in J \\
 & -\sum_{i\in I} u_i^1 \bar{a}_{i0} & +u_0\bar{a}_{j0} & & & \geq & \beta & \\
 & & & -\sum_{i\in I} v_i^1 \bar{a}_{i0} & +v_0(1-\bar{a}_{j0}) & \geq & \beta &
\end{array}$$

As before, we can write the cut as $\alpha x \geq \beta$, with $\alpha_k = max\,\{\alpha_k^1, \alpha_k^2\}$ and $\beta = min\,\{\beta_1, \beta_2\}$, where

$$\begin{array}{llllllr}
\alpha_k^1 & = & -\sum_{i\in I} u_i^1 \bar{a}_{ik} + u_0\bar{a}_{jk}, & \alpha_k^2 & = & -\sum_{i\in I} v_i^1 \bar{a}_{ik} - v_0\bar{a}_{jk}, & k \in J \\
\beta^1 & = & -\sum_{i\in I} u_i^1 \bar{a}_{i0} + u_0\bar{a}_{j0}, & \beta^2 & = & -\sum_{i\in I} v_i^1 \bar{a}_{i0} + v_0(1-\bar{a}_{j0}). &
\end{array}$$

The strengthened cut then becomes $\gamma x \geq \frac{\beta}{|\beta|}$ where γ_k is defined by (16), (17).

Theorem 3.3 *The mixed integer Gomory cut [G60] is $\gamma x \geq \frac{\beta}{|\beta|}$ with the choice of multipliers:*

$$\begin{array}{llllllr}
u_i^1 & = & 0, & v_i^1 & = & 0, & i \in I, \\
u_0 & = & 1/\bar{a}_{j0}, & v_0 & = & 1/(1-\bar{a}_{j0}). &
\end{array}$$

Proof. Using the multipliers defined in the theorem we obtain:

$$\alpha_k^1 = \bar{a}_{jk}/\bar{a}_{j0}, \qquad \alpha_k^2 = -\bar{a}_{jk}/(1-\bar{a}_{j0}), \qquad k \in J$$

and $\beta_1 = \beta_2 = 1$. Substituting these values into (16) and (17) yields $m_k = -\bar{a}_{jk}$ and

$$\gamma_k = \min\left\{\frac{\bar{a}_{jk} + \lceil -\bar{a}_{jk} \rceil}{\bar{a}_{j0}}, \frac{-\bar{a}_{jk} - \lfloor -\bar{a}_{jk} \rfloor}{(1-\bar{a}_{j0})}\right\} \text{ for } k = 1, \ldots, p$$

$$\gamma_k = \max\left\{\frac{\bar{a}_{jk}}{\bar{a}_{j0}}, \frac{-\bar{a}_{jk}}{(1-\bar{a}_{j0})}\right\} \text{ for } k = p+1, \ldots, n$$

which is the mixed integer cut of [G60]. ■

4 Computational Experience

Preliminary versions of the cutting plane procedure discussed in Section 3.1, with Normalizations 2, 3, and 4, were tested on several classes of problems. At every iteration, Step 2 was applied to every $j \in \{1, \ldots, p\}$ such that $0 < x_j^t < 1$; i.e., a cut was generated for every 0-1 variable that took a fractional value in the optimal solution to the linear program solved in Step 1. The cuts were generated in the subspace defined by the fractional variables and then lifted with the procedure of Section 3.4 into the full space. The strengthening procedure of Section 3.5 was then applied to every cut generated. The strengthened cuts were considered in the order of the amount by which they were violated by the current solution (most violated first), and a cut was added to the constraint set if the cosine of the angle between its normal vector and that of all previously added cuts differed by at most $\theta < 1$, where θ is a parameter chosen by the user (Here we took $\theta = 0.999$). Similarly, the remaining cuts were considered one at a time starting with the most violated one. Each cut was compared with all previously added cuts for this iteration, using the parameter θ to decide whether to add it to the formulation. The experiments were run for a maximum of 30 iterations, and if the objective function failed to improve significantly over several consecutive iterations, the run was stopped earlier.

The linear programs encountered during the procedure were solved using the CPLEX library. For the purpose of benchmarking and comparison, the test problems were also solved with a branch and bound code (LINDO for most of the problems, Carpaneto and Toth's [CT80] for the TSP's), as well as with a procedure using Gomory's cutting planes. For better comparability, the Gomory cuts were used in the framework of our procedure; i.e. in Step 2 of our procedure, instead of generating one of our cuts for each $j \in \{1, \ldots, p\}$ such that

$0 < x_j^t < 1$, a mixed integer Gomory cut was generated from each row of the simplex tableau corresponding to a (basic) variable x_j such that $0 < x_j^t < 1$. As with our procedure, we used the parameter θ to decide which cuts to add to the formulation. The tests were run on a SUN Sparcstation 330.

We considered four different classes of test problems. The first one is a set of unstructured 0-1 programs where Normalization 2 does not apply. The set is used to compare Normalizations 3 and 4 with Gomory cuts. The second class is a set of randomly generated vertex packing problems where we compare Normalization 2 with Gomory cuts. The third class are fixed-charge network problems formulated as mixed 0-1 programs. The last class consists of two real world asymmetric TSP's where our algorithm is compared with a problem specific branch and bound algorithm.

The first class of test problems consists of a set of pure 0-1 programs with fairly large integrality gaps (i.e., differences between the value of the LP and IP optima). The BM problems are tightly constrained general 0-1 programs with positive and negative coefficients, randomly generated by Bouvier and Messoumian [BM65]. The LSB and LSC problems have a real world origin and are taken from Lemke and Spielberg [LSp67]. The PE problems are capital budgeting (multiple knapsack) models, with all positive coefficients, originating with Peterson [PE67]. All of these problems are also described in [BMa80]. The CJP problems are from Crowder, Johnson and Padberg [CJP80] and have a real world origin. Table 1 describes the problems by giving the number of their variables and constraints, the value of the LP optimum and the integer optimum, and the number of search tree nodes it took LINDO's branch and bound code to solve them.

Problem name	Number of constraints	Number of variables	Value of LP optimum	Value of IP optimum	Branch & bound tree nodes
BM13	15	15	14.96	26	26
BM14	15	15	1.50	2	2
BM19	25	20	31.05	47	16
BM20	27	20	33.96	47	14
BM22	20	28	19.31	33	224
BM24	20	28	25.78	38	242
LSB	28	35	521.05	550	42
LSC	12	44	56.61	73	934
PE4	10	20	-6155.33	-6120	76
PE5	10	28	-12462.10	-12400	98
PE6	5	39	-10672.34	-10618	84
PE7	5	50	-16612.82	-16537	476
CJP33	15	33	2520.57	3089	7086
CJP40	23	40	61796.54	62027	104
CJP201	133	201	6875.00	7615	1730
CJP282	241	282	176867.50	258411	10252

Table 1

Table 2 shows the difference between running our algorithm with and without cut strengthening.

	Normalization 4					
	Strengthening			No Strengthening		
Problem Name	Cuts	Iterations	% Gap closed	Cuts	Iterations	% Gap closed
LSB	150	14	100	261	30	91
LSC	51	7	100	317	30	84

Table 2

Table 3 compares two versions of our algorithm with a version using Gomory cuts. Table 4 shows our results on the PE and CJP problems.

	Normalization 3			Normalization 4			Gomory cuts		
Problem Name	Cuts	Iterations	% Gap closed	Cuts	Iterations	% Gap closed	Cuts	Iterations	% Gap closed
BM13	370	30	100	28	4	100	110	30	42
BM14	2	1	100	2	1	100	2	1	100
BM19	121	13	100	42	4	100	112	30	79
BM20	132	15	100	23	4	100	115	30	79
BM22	405	30	51	446	30	55	93	30	32
BM24	514	30	53	536	30	53	103	30	32
LSB	461	30	97	150	14	100	65	30	65
LSC	370	30	100	51	7	100	35	7	100

Table 3

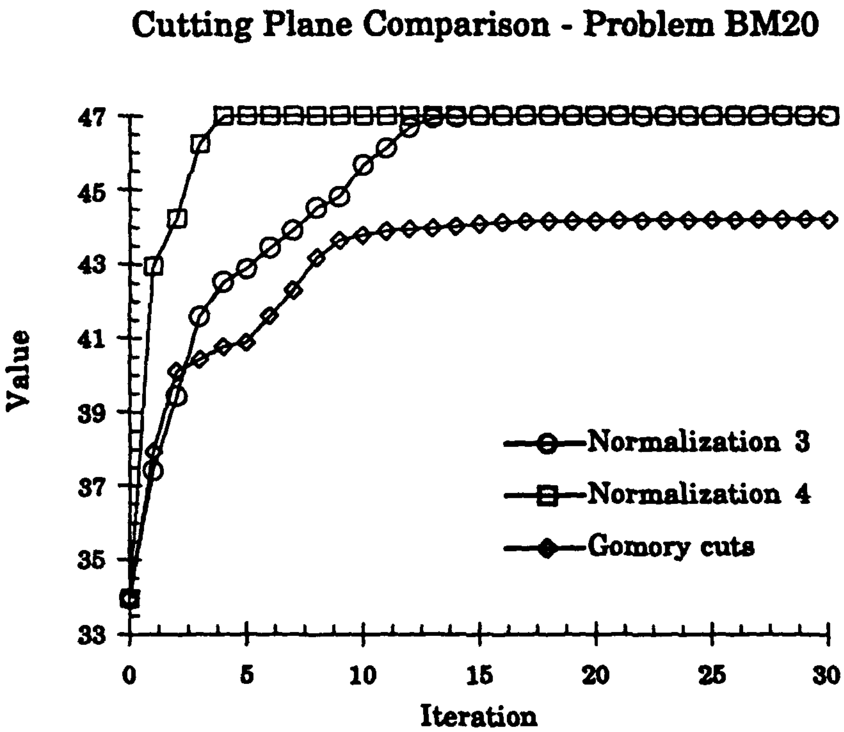
	Normalization 4			Gomory cuts		
Problem Name	Cuts	Iterations	% Gap closed	Cuts	Iterations	% Gap closed
PE4	128	24	100	46	30	94
PE5	148	30	92	40	30	67
PE6	276	30	79	40	30	35
PE7	302	30	86	46	30	24
CJP33	558	30	77	168	30	72
CJP40	19	5	100	12	4	100
CJP201	1024	15	73	365	30	66
CJP282	521	15	96	350	30	22

Table 4

As it can be seen from Tables 3 and 4, the procedure with Normalization 4 found optimal solutions to 8 of the 16 problems. Normalization 4 seems consistenly better than 3. As mentioned earlier, we ran the procedure for at most 30 iterations in each case. Although the tables do not reflect this, most of the gap reduction tends to happen during the first few iterations. In a few instances, like CJP201 and CJP282, after 15 iterations progress had slowed down to the extent that prompted the termination of the run.

Figure 1 contains detailed information for the solution of the problem BM20, iteration per iteration.

The next class of problems consists of randomly generated vertex packing (maximum stable set) problems. We present the solution of one problem on a graph with 30 vertices and 30% edge density and three problems on graphs with 100 vertices and densities 5, 10 and 15%. Since all the cuts $\alpha x \geq \beta$ for this problem have $\alpha \leq 0$ and $\beta < 0$, Normalization 2 was used (with $\beta = -1$). Table 5 contains the problem data; the last column gives the number of search tree nodes generated by LINDO's branch code. Table 6 contains the results of the runs. For the 30-vertex problem we started with the edge formulation $FS(G)$ as described in Section 2.5. The initial LP solution was all fractional ($x_j = 1/2$, for $j = 1, \ldots, 30$). The problem was solved in 10 iterations after adding 242 cuts. It is interesting to note that the cutting planes generated in iteration 1, 2 and 3 were almost exclusively clique inequalities. The striking point is that the cuts do not deteriorate as quickly as with other general cutting plane algorithms. For example, in iteration 4, after 90 cuts had already been added to the problem, the cuts that were generated included clique inequalities, odd hole inequalities and lifted odd-hole inequalities. Even after iteration 4, the cutting planes generated had small integer left-hand-side coefficients (after appropiate scaling of the right-hand-side coefficient). For the 100-vertex problems, we started from a formulation in which the rows of the constraint matrix represent a clique cover of all the edges. Each clique in the cover is obtained in a greedy fashion by starting with a vertex of largest degree and expanding the clique with a neighbor of largest degree until the clique is maximal. The cutting plane algorithm was able to solve problem VP5 in 6 iterations (this problem took 4000 nodes when solved by LINDO's branch and bound code), but it ran into difficulties with problem VP15 after closing 2/3 of the integrality gap.



Cutting Plane Comparison - Problem BM20
47
45
43
41
39
37
35
33
Value
0
5
10
15
20
25
30
Iteration
Normalization 3
Normalization 4
Gomory cuts

Figure 1

Problem name	Number of constraints	Number of variables	Value of LP optimum	Value of IP optimum	Branch & bound tree nodes
VP30	168	30	15.00	7	62
VP5	211	100	45.25	42	4000
VP10	356	100	38.00	31	1024
VP15	440	100	33.86	25	2670

Table 5

	Normalization 2			Gomory Cuts		
Problem Name	Cuts	Iterations	% Gap closed	Cuts	Iterations	% Gap closed
VP30	242	10	100	466	30	62
VP5	398	6	100	1015	30	66
VP10	1129	13	91	1239	25	17
VP15	619	7	67	773	30	6

Table 6

The next class of problems we considered are fixed-charge network flow problems of the form:

$$
\begin{array}{llll}
\text{Min } \sum c_{ij}x_{ij} + \sum h_{ij}y_{ij} & & & \\
\text{subject to} & & & \\
\sum_j y_{ij} \;\; - \sum_j y_{ji} & = & b_i & \text{for all } i \\
\qquad\qquad\quad y_{ij} & \le & u_{ij}x_{ij} & \text{for all arcs } (i,j) \\
\qquad\qquad\quad y & \ge & 0, \qquad x_{ij} \in \{0,1\} & \text{for all arcs } (i,j)
\end{array}
$$

We randomly generated the problems as follows. The first three problems are fixed-charge capacitated transportation problems, i.e. fixed-charge network flow problems on a bipartite graph. The next thı are fixed-charge problems on general networks. The arcs in the network are randomly generated to match a specified density of the graph. For the first class we generated three problems, CTR1, CTR2, and CTR3, with each of the two node sets in the bipartite graph of sizes 10, 15, and 20, and densities 80%, 50% and 35%, respectively.

For the second class we generated three problems, FXC1, FXC2, and FXC3, on networks with 10, 15, and 20 nodes, and the densities of 80%, 50% and 35 %, respectively. For both classes, the fixed cost c_{ij} of opening an arc (i,j) was randomly generated as an integer in the range [0,20], the variable cost h_{ij} of using an arc as a real number in the range [0,2], and the capacity u_{ij} of an arc as an integer in the range [1,20]. The demands and supplies b_i where randomly generated as integers in the range $[-20,20]$ and so that they satisfy $\sum b_i = 0$.

Table 7 contains the description of these problems, including the number of constraints, the number of 0-1 and continuous variables and the number of nodes that it took LINDO's branch and bound to solve them. Table 8 contains the results with our cutting plane procedure.

Problem name	Constraints	Integer variables	Continuous variables	Value of LP optimum	Value of IP optimum	B & B tree nodes
CTN1	103	83	83	128.58	183.34	180800
CTN2	150	120	120	169.79	239.21	1060254[1]
CTN3	182	142	142	313.80	432.28	465382[1]
FXC1	92	82	82	46.10	62.62	740
FXC2	123	108	108	116.19	148.91	1352
FXC3	161	141	141	152.01	197.98	773096

Table 7

[1]Number of nodes at which run was stopped without finding an optimal solution.

	Normalization 4		
Problem Name	Cuts	Iterations	% Gap closed
CTN1	434	15	94
CTN2	511	15	99
CTN3	582	15	95
FXC1	215	15	98
FXC2	388	15	95
FXC3	451	15	90

Table 8

We note that although the problems in this set are very difficult for a general branch and bound algorithm, our procedure manages to close most of the integrality gap, as illustrated by the data in Table 8. LINDO's branch and bound code was not able to solve problems CTN2 and CTN3 to optimality when using the standard linear programming relaxation for bounding. In order to solve both problems we applied branch and bound to the strengthened linear program resulting after 8 iterations of our algorithm. In Table 9 we show for problem CTN2, the benefits of strengthening the formulation before applying branch and bound. The results are given for a number of iterations of our strengthening procedure ranging from 0 to 11. We report the total number of cuts generated by our algorithm, the CPU time (in minutes) taken to generate these cuts, the number of nodes in the branch and bound tree needed by LINDO to solve the problem, and the total computing time taken (in minutes) including cut generation.

Iteration number	Cuts	CPU time	B & B tree nodes	Total CPU time
0	0	0.00	>1000000	>2000
1	27	0.38	>600000	>2000
2	47	0.63	360012	1640.17
3	76	1.78	42120	335.28
4	110	3.87	32758	391.78
5	150	10.32	11498	224.25
6	185	15.57	1486	86.57
7	222	21.52	3522	215.52
8	274	50.09	2548	288.32
9	322	76.64	694	167.10
10	348	81.93	622	214.78
11	382	94.58	>500	>2000

Table 9

The last class of test problems consists of two difficult real world asymmetric traveling salesman problems. The two problems that we ran have 17 and 43 nodes (cities), respectively. They are samples of scheduling problems that arise regularly at chemical plants of the Dupont Company. These problems proved to be hard to solve by other existing methods. In particular, the branch and bound code of Carpaneto and Toth [CP80], one of the most efficients codes for this problem, took more than 110,000 nodes in the branch and bound tree to solve the problem on 17 cities. For the problem on 43 cities, it could not find a tour after running for 25.2 CPU hours, exceeding the memory limitations and enumerating more than 580,000 nodes in the branch and bound tree.

We ran our algorithm on these problems with a change in Step 1. Instead of the standard linear programming relaxation, the algorithm starts by solving the assignment problem. If the LP solution is a tour, the algorithm stops. If the LP solution is integer (i.e. an assignment) but not a tour, the algorithm identifies the subtours and adds to the linear program the corresponding subtour elimination inequalities and repeats Step 1. If the solution to the current linear program is fractional, Step 2 of the algorithm is applied, i.e. a family of cutting planes is generated.

For the 17 city problem our algorithm needed 10 iterations to solve the problem to optimality, four of them corresponding to the generation of subtour elimination constraints. A total of 151 cuts were added during the procedure.

In the case of the 43 city problem, in view of the large number of edges, we had to use cost matrix sparsification techniques. To get an initial sparse cost matrix we generated several "good" tours heuristically and took the union of their arc sets as the only arcs of our graph. We then applied to this sparse problem our cutting plane procedure in its modified form described above. When an optimal tour was obtained for the sparse problem, the reduced costs of the missing arcs were checked, and a subset of the arcs with negative reduced costs was added to the problem. The procedure was then repeated.

The best earlier solution to this problem, of value 5621, was found by Repetto [R91] using a collection of heuristics. Our procedure was able to find, after the second set of edges was chosen, and 20% of the arcs were present, a solution of value 5620. After several additional iterations there were still missing arcs with negative reduced costs, and the procedure was stopped as computationally too expensive.

A second approach was then tried. The linear programming relaxation was solved with the fully dense cost matrix, with only a subset of the subtour elimination constraints. The lower bound LB obtained this way was then used in conjunction with the upper bound $UB = 5620$ to fix at 0 all variables whose reduced cost exceeded $UB - LB$. Cutting planes were then generated, which raised the value of LB to 5616. At this point, with an integrality gap of $5620 - 5616 = 4$, the problem containing only those constraints tight at the last linear programming optimum, was fed to LINDO's branch and bound code. This linear program had a total of 172 constraints, 86 of them corresponding to assignment constraints, 57 to subtour elimination constraints and 29 to cuts generated with the cutting plane algorithm. After enumerating 374 nodes in the branch and bound tree the code was able to prove optimality of the solution with value 5620 and the problem was solved.

These computational experiments, although preliminary, suggest several conclusions. First, like with other cutting plane algorithms, the effect of the cutting planes on the objective function value, and hence on the integrality gap, is more significant at the beginning and less significant as more cuts are generated. This is partly due to the fact that as the number of cuts increases, their direction tends to get closer to that of the objective function. This points to the need of embedding the procedure into an enumerative framework, so that whenever the cuts become "shallow", branching can be performed to move away from the current LP optimum. Nevertheless, it is a remarkable feature of our procedure, and the family of cutting planes it generates, that no *numerical problems* were encountered (with these cuts) throughout the experiment.

5 Bibliography

[B71] E. Balas, Intersection cuts - A new type of cutting planes for integer programming, *Operations Research*, 19, (1971), 19-39.

[B74a] E. Balas, Intersection cuts for disjunctive constraints, MSRR No. 330, Carnegie Mellon University, February 1974.

[B74b] E. Balas, Disjunctive Programming: Properties of the convex hull of feasible points, MSRR No. 348, Carnegie-Mellon University, (July 1974).

[B79] E. Balas, Disjunctive Programming, *Annals of Discrete Mathematics*, 5, (1979), 3-51.

[B85] E. Balas, Disjunctive Programming and a Hierarchy of relaxations for discrete optimization problems, *SIAM Journal on Algebraic and Discrete Methods*, Vol. 6, (1985), 466-486.

[BJ80] E. Balas and R. Jeroslow, Strengthening cuts for mixed integer programs, *Europen Journal of Operations Research* 4, No. 4, (1980).

[BMa80] E. Balas and C. Martin, Pivot and Complement-A heuristic for 0-1 programming, *Managment Science*, Vol. 26, No. 1, (1980), 86-96.

[BTT89] E. Balas, J. Tama and J. Tind, Sequential convexification in reverse convex and disjunctive programming, *Mathematical Programming*, No. 44, (1989), 337-350.

[BM65] B. Bouvier and G. Messoumian, Programmes Lineaires en variables bivalentes-Algorithme de Balas, Université de Grénoble, France, (1965).

[CT80] C. Carpanetto and P. Toth, Branching and bounding criteria for the asymmetric traveling salesman problem, *Managment Science*, Vol. 26, No. 7, (1980).

[Gl73] F. Glover, Convexity cuts and cut search, *Operations Research*, No. 21, (1973),123-134.

[G60] R. Gomory, An algorithm for the mixed integer problem, RM-2597, The Rand Corporation, (1960).

[J80] R. Jeroslow, A cutting plane game for facial disjunctive programs, *SIAM J. Control and Optimization*, Vol. 18, No. 3, (1980), 264-280.

[LS89] L. Lovász and A. Schrijver, Cones of matrices and set functions, and 0-1 optimization, Report BS-R8925, Centrum voor Wiskunde en Informatica, (1989).

[LSp67] C. Lemke and K. Spielberg, A capital budgeting heuristic algorithm using exchange operations, *AIEE Transactions*, Vol. 6, (1974), 143-150.

[PE67] C. Petersen, Computational experience with variants of the Balas algorithm applied to the selection of R&D projects, *Management Science*, Vol. 13, (1967), 736-750.

[PR87] M. Padberg and G. Rinaldi, Optimization of a 537-city TSP by Branch and Cut, *OR Letters*, 6, 1-8.

[R91] B. Repetto, personal communication.

[Sa75] H. Salkin, Integer Programming, Addison Wesley, (1975).

[SA88] H. Sherali and W. Adams, A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems, Virginia Tech, Technical Report, (1988).

[SA89] H. Sherali and W. Adams, A hierarchy of relaxations and convex hull representations for mixed-integer zero-one programming problems, Virginia Tech, Technical Report, (1989).

[VW87] T. Van Roy and L. Wolsey, Solving mixed integer programming problems using automatic reformulation, *Operations Research*, Vol. 35, No. 1, (1987), 45-47.

[W61] W. White, On Gomory's mixed integer algorithm, Senior Thesis, Department of Mathematics, Princeton University, (1961).

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MSRR #576	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A LIFT-AND-PROJECT CUTTING PLANE ALGORITHM FOR MIXED 0-1 PROGRAMS		5. TYPE OF REPORT & PERIOD COVERED Technical Report--Oct 1991
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Egon Balas Sebastian Ceria Gerard Cornuejols		8. CONTRACT OR GRANT NUMBER(s) N00014-85-K-0198
9. PERFORMING ORGANIZATION NAME AND ADDRESS Graduate School of Industrial Administration Carnegie Mellon University Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Personnel and Training Research Programs Office of Naval Research (Code 434) Arlington, VA 22217		12. REPORT DATE October 1991
		13. NUMBER OF PAGES 39
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report)
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report)		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Cutting planes Projection Mixed 0-1 programming Disjunctive programming		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We propose a cutting plane algorithm for mixed 0-1 programs based on a family of polyhedra which strengthen the usual LP relaxation. We show how to generate a facet of a polyhedron in this family which is most violated by the current fractional point. This cut is found through the solution of a linear program that has about twice the size of the usual LP relaxation. A lifting step is used to reduce the size of the LP's needed to generate the cuts. An additional strengthened step suggested by Balas and Jeroslow is then applied. We report our computational experience with a preliminary version of the algorithm. This approach is related to

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered) the work of Balas on disjunctive programming, the matrix cut relaxations of Lovasz and Schrijver and the hierarchy of relaxations of Sherali and Adams.