Reduce-and-split cuts: Improving the performance of mixed integer Gomory cuts [1]

Kent Andersen [2] Gérard Cornuéjols [2] Yanjun Li [3]

January 20, 2005

Abstract

Mixed integer Gomory cuts have become an integral part of state-of-the-art software for solving mixed integer linear programming problems. Therefore, improvements in the performance of these cutting planes can be of great practical value. In this paper we present a simple and fast heuristic for improving the coefficients on the continuous variables in the mixed integer Gomory cuts. This is motivated by the fact that, in a mixed integer Gomory cut, the coefficient of an integer variable lies between 0 and 1, whereas for a continuous variable, there is no upper bound. The heuristic tries to reduce the coefficients of the continuous variables. We call the resulting cuts reduce-and-split cuts. We found that, on several test problems, reduce-and-split cuts can substantially enhance the performance of a branch-and-bound algorithm.

Keywords: Integer programming, mixed integer programming, cutting plane, split cut, mixed integer Gomory cut, reduce-and-split cut

1 Introduction

Many management problems can be formulated as mixed integer linear programs (MILP). The diversity of applications is well represented in MIPLIB 3.0 [8], an electronically available library of MILP's: airline crew scheduling, telecommunication network design, electricity generation, and many others. Other recent application areas include combinatorial auctions (CombineNet), financial engineering (Axioma) and sports scheduling such as for the National Football League (Optimal Planning Solutions) and Major League Baseball (The Sports Scheduling Group), just to cite a few.

MILP formulations have become more practical recently due to great improvements in commercial software (Xpress and Cplex are currently two of the best). Indeed, MILP's that would have required years of computing time 15 years ago can be solved in seconds today: The speedup factor is roughly 1000 for linear programming algorithms, 100 for integer programming algorithms and 1000 for hardware (desktop computers) for an overall speedup of 100 million (see Bixby, Gu, Rothberg and Wunderling [9] for a detailed study). Despite these spectacular improvements, there is need for greater speedups in several application areas (such as combinatorial auctions and sports

[1] Supported by NSF grant DMI-0098427 and ONR grant N00014-97-1-0196.
[2] Tepper School of Business, Carnegie Mellon University, Pittsburgh, PA, emails: {kha,gc0v}@andrew.cmu.edu.
[3] Krannert School of Management, Purdue University, West Lafayette, IN, email: li14@mgmt.purdue.edu.

scheduling, for instance). There is potential for further gains in integer programming algorithms. The purpose of this paper is to propose such an improvement.

The best current software packages for solving MILP's use a combination of techniques, including enumeration (branch-and-bound) and several types of cutting planes. The idea of solving mixed integer linear programs using cutting planes dates back to the work of Gomory [18] in the late fifties and early sixties. Mixed Integer Gomory cuts (MIG cuts) fell out of favor in the late sixties, but were revived in the nineties (Balas, Ceria, Cornuéjols and Natraj [5]). Today, MIG cuts play a major role in state-of-the-art software for solving MILP's (Bixby, Gu, Rothberg and Wunderling [9]), accounting for much of the speedup in integer programming algorithms beyond the improvements in linear programming mentioned above. Therefore any improvement in the performance of these cuts can be of great practical value.

In this paper, we exploit the fact that MIG cuts are split cuts (Balas [2], Cook, Kannan and Schrijver [12] and Nemhauser and Wolsey [21]) to improve their performance. Any split cut can be derived from a basis of the linear programming relaxation and a disjunction (Andersen, Cornuéjols and Li [1]). To improve a MIG cut, we try to improve the disjunction while keeping the basis fixed. This can be viewed as a natural counterpart of the work of Balas and Perregaard [7], where they improve the cut by modifying the basis while keeping the disjunction fixed. Their procedure generates lift-and-project cuts [4] and it has been implemented recently in the software package Xpress. Other strengthening procedures were proposed by Ceria, Cornuéjols and Dawande [11] and by Eckstein and Nediak [17].

What do we mean by "improving" a MIG cut? This question is essentially the same as the question of how to measure cut quality. A possible measure is the *distance cut off*, i.e. the Euclidean distance between the cut and the point it is designed to cut off. In this paper, we present a closed form formula for the distance cut off by a split cut. Furthermore, we analyze the factors that determine this measure. We find that an important factor is the size of the coefficients of the continuous variables in the cut. This is used to design an algorithm for modifying the MIG cuts to generate another set of split cuts with reduced coefficients on the continuous variables. We call these cuts reduce-and-split cuts. We applied reduce-and-split cuts to a set of test problems from MIPLIB 3.0 [8], a standard library of test problems for mixed integer linear programming. On several test problems, these cuts drastically reduce the number of nodes in the search tree in a cut-and-branch framework.

Consider the Mixed Integer Linear Program (MILP):

$$\text{(MILP)} \qquad min\{c^T x : Ax = b, x \geq 0_n, x_j \text{ integer for } j \in N_I\},$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ and $N_I \subseteq N := \{1, 2, \ldots, n\}$. WLOG assume A is of full row rank. The linear programming problem obtained from MILP by dropping the integrality conditions on x_j for $j \in N_I$ is denoted LP. The sets P_I and P denote the feasible solutions to MILP and LP respectively. A basis matrix for LP is an $m \times m$ invertible submatrix of A. A basis for LP is an m-subset B of N such that the submatrix of A induced by B is a basis matrix.

An important concept for cut generation is that of a disjunction (see Balas [3]). In fact, any MILP can be formulated as a linear program in which the feasible solutions have to satisfy a set of disjunctions. An important special case is the *split disjunction* $D(\pi, \pi_0)$ of the form $\pi^T x \leq \pi_0 \vee \pi^T x \geq \pi_0 + 1$, where $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$ and $\pi_j = 0$ for $j \notin N_I$. Here "$\vee$" denotes the "or" operator, which requires that x satisfies either $\pi^T x \leq \pi_0$ or $\pi^T x \geq \pi_0 + 1$. Clearly any feasible solution to

MILP has to satisfy every split disjunction. Split disjunctions can therefore be used to generate cutting planes that cut off points of P violating $D(\pi, \pi_0)$. The set of all split disjunctions is denoted by $\Pi := \{D(\pi, \pi_0) : (\pi, \pi_0) \in \mathbb{Z}^{n+1} \text{ and } \ \pi_j = 0 \text{ for } j \notin N_I\}$.

Let $D(\pi, \pi_0) \in \Pi$ be an arbitrary split disjunction, and let $F_{D(\pi,\pi_0)}$ denote the set of points in $\mathbb{R}^n$ that satisfy this disjunction. Clearly any inequality that is valid for the points in $P \cap F_{D(\pi,\pi_0)}$ is also valid for $Conv(P \cap F_{D(\pi,\pi_0)})$, where $Conv(P \cap F_{D(\pi,\pi_0)})$ denotes the convex hull of $P \cap F_{D(\pi,\pi_0)}$. A *split inequality* is an inequality that is valid for $Conv(P \cap F_{D(\pi,\pi_0)})$ for some split disjunction $D(\pi, \pi_0)$. The set of points in P that satisfy all split inequalities is called the *split closure* of MILP, and is defined by:

$$\mathrm{SC} := \bigcap_{D(\pi,\pi_0)\in\Pi} Conv(P \cap F_{D(\pi,\pi_0)}).$$

Any split inequality is equal to or dominated by a split cut derived from a basis B and a split disjunction $D(\pi, \pi_0)$ (Andersen, Cornuéjols and Li [1]). Also, any MIG cut can be viewed as a split cut (Balas [2] and Nemhauser and Wolsey [21]). This means that, given any MIG cut, there exists $D(\pi^G, \pi_0^G) \in \Pi$ and a basis B such that the split cut derived from $D(\pi^G, \pi_0^G)$ and B is the given MIG cut. We use this relationship to improve the performance of MIG cuts.

The remainder of this paper is organized as follows. In Sect. 2, we present the theoretical foundation for our algorithm. In particular, we derive a closed form formula for the distance cut off by a split cut, i.e. a formula that, given any split disjunction $D(\pi, \pi_0) \in \Pi$ and a basis B, outputs the distance cut off $d(B, \pi, \pi_0)$ by the split cut derived from $D(\pi, \pi_0)$ and B. In Sect. 3, we use the theory presented in Sect. 2 to design an algorithm for generating cutting planes that improve the MIG cuts on the continuous variables. In Sect. 4, we present our experiments to test the performance of these cutting planes. We incorporate the cuts into the commercial code CPLEX [15]. The resulting algorithm is tested on MIPLIB 3.0 [8].

2 Theoretical foundation

This section contains the theoretical foundation for the cut generator we have developed. Throughout this section, B denotes an arbitrary basis of LP. The basis B is not required to be feasible. The outline of the section is as follows. In Sect. 2.1, we introduce the conic polyhedron associated with the basis B. In Sect. 2.2, we demonstrate how a split cut can be derived from a split disjunction and B. In Sect. 2.3, we derive a closed form formula for the distance cut off by a split cut. Split cuts derived from bases of LP are also known as *intersection cuts* (Balas [2]). In Sect. 2.4, we show that the strengthening procedure introduced by Balas and Jeroslow [6] can be re-interpreted as a modification procedure for the underlying split disjunction. In Sect. 2.5, we demonstrate that any MIG cut can be obtained as an intersection cut by choosing the disjunction appropriately (Gomory [18] and Balas [2]). In Sect. 2.6, we demonstrate the equivalence between split cuts and MIG cuts generated from integer combinations of the rows of the simplex tableau.

2.1 The conic polyhedron associated with a basis

A key concept for the theory presented here is the idea of relaxing the linear programming relaxation of MILP by dropping constraints. Specifically, the relaxations we use can be derived from bases of LP. Let $J := N \setminus B$ index the non-basic variables. The *conic polyhedron* associated with B is given by:

$$P(B) := \{x \in \mathbb{R}^n : Ax = b \text{ and } x_j \geq 0 \text{ for } j \in J\}. \tag{1}$$

The set $P(B)$ is the relaxation of P obtained by deleting the non-negativity constraints on the basic variables. Observe that $P(B)$ is a translate of a polyhedral cone. Specifically, we may write $P(B) = C + \bar{x}$, where C is the polyhedral cone $C := \{x \in \mathbb{R}^n : Ax = 0 \text{ and } x_j \geq 0 \text{ for } j \in J\}$, and $\bar{x}$ solves the system $Ax = b$ and $x_j = 0$ for $j \in J$. The vector $\bar{x} \in \mathbb{R}^n$ is known as the *basic solution* corresponding to the basis B.

The extreme rays of the polyhedral cone C can be obtained as follows. Observe that, since the columns of A indexed by B are linearly independent, we can solve the system $Ax = b$ in terms of the basic variables:

$$\bar{x}_i = x_i + \sum_{j \in J} \bar{a}_{ij} x_j, \qquad i \in B. \tag{2}$$

The above system is known as the *simplex tableau*. The extreme rays of C can be obtained from the coefficients of the simplex tableau as follows. Given $j \in J$, define the vector r^j:

$$r^j_k := \begin{cases} -\bar{a}_{kj} & \text{if } k \in B, \\ 1 & \text{if } k = j, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

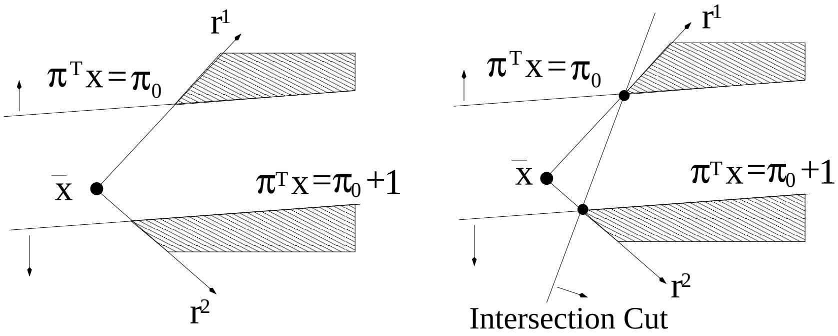
The conic polyhedron $P(B)$ can then be written as $P(B) = \bar{x} + \mathit{Cone}(\{r^j\}_{j \in J})$, where $\mathit{Cone}(\{r^j\}_{j \in J})$ denotes the polyhedral cone generated by the vectors $\{r^j\}_{j \in J}$. Observe that, since there are $|B| = m$ basic variables, there are $|J| = n - m$ non-basic variables. It follows that $P(B)$ has exactly $n - m$ extreme rays.

2.2 Intersection cuts and split cuts

We now derive the intersection cut introduced by Balas [2]. Let $D(\pi, \pi_0)$ be an arbitrary split disjunction. Assume $\bar{x}$ violates the disjunction $D(\pi, \pi_0)$, and define $\epsilon(\pi, \pi_0) := \pi^T \bar{x} - \pi_0$ to be the amount by which $\bar{x}$ violates the first term of the disjunction $D(\pi, \pi_0)$. Since $\pi_0 < \pi^T \bar{x} < \pi_0 + 1$, we have $0 < \epsilon(\pi, \pi_0) < 1$. Also, for $j \in J$, define scalars:

$$\alpha_j(\pi, \pi_0) := \begin{cases} -\frac{\epsilon(\pi,\pi_0)}{\pi^T r^j} & \text{if } \pi^T r^j < 0, \\ \frac{1-\epsilon(\pi,\pi_0)}{\pi^T r^j} & \text{if } \pi^T r^j > 0, \\ +\infty & \text{otherwise.} \end{cases} \tag{4}$$

The interpretation of the numbers $\alpha_j(\pi, \pi_0)$ for $j \in J$ is the following. Let $x^j(\alpha) := \bar{x} + \alpha r^j$, where $\alpha \in \mathbb{R}_+$, denote the half-line starting in $\bar{x}$ in the direction r^j. The value $\alpha_j(\pi, \pi_0)$ is the smallest value of $\alpha \in \mathbb{R}_+$ such that $x^j(\alpha)$ satisfies the disjunction $D(\pi, \pi_0)$. In other words, the point $x^j(\alpha_j(\pi, \pi_0))$ is the intersection of the half-line starting in $\bar{x}$ in direction r^j with the hyperplane



(a) Conic polyhedron and a split disjunction

(b) Intersection cut

Figure 1: Deriving the intersection cut from a conic polyhedron

$\pi^T x = \pi_0$ or the hyperplane $\pi^T x = \pi_0 + 1$. Note that $\alpha_j(\pi, \pi_0) = +\infty$ when the direction r^j is parallel to the hyperplane $\pi^T x = \pi_0$. Given the numbers $\alpha_j(\pi, \pi_0)$ for $j \in J$, the *intersection cut* associated with B and $D(\pi, \pi_0)$ is given by:

$$\sum_{j \in J} \frac{x_j}{\alpha_j(\pi, \pi_0)} \geq 1. \tag{5}$$

The validity of this inequality for the set of points in $P(B)$ that satisfy the disjunction $D(\pi, \pi_0)$ was proven by Balas [2]. In fact, the intersection cut gives a complete description of the set of points in $P(B)$ that satisfy the disjunction $D(\pi, \pi_0)$, i.e. we have:

$$Conv(P(B) \cap F_{D(\pi,\pi_0)}) = \{x \in P(B) : (5)\}.$$

Figure 1 gives an example of a conic polyhedron with two extreme rays ($|J| = 2$). The split disjunction $D(\pi, \pi_0)$ splits the conic polyhedron into the two disjoint shaded parts. Observe that the two points that determine the intersection cut are the first points that satisfy the disjunction $D(\pi, \pi_0)$ when traveling from $\bar{x}$ in the directions r^1 and r^2.

In a recent paper [1], we showed that intersection cuts are sufficient for describing the split closure of MILP. Let $\mathcal{B}^*$ denote the set of all bases of LP. We have:

$$\mathrm{SC} = \bigcap_{B \in \mathcal{B}^*} \bigcap_{D(\pi,\pi_0) \in \Pi} Conv(P(B) \cap F_{D(\pi,\pi_0)}).$$

However, it is NP-hard to optimize a linear function over SC, even when P is a conic polyhedron [10], [13].

2.3 The distance cut off

In this section, we give a closed form formula for the Euclidian distance cut off by an intersection cut derived from a split disjunction $D(\pi, \pi_0)$ and a basis B, i.e. the distance between $\bar{x}$ and the hyperplane defined by (5). Let $d(B, \pi, \pi_0)$ denote the distance cut off by this cut. We have:

Lemma 1 *Let B be a basis of LP, let $\bar{x}$ be the corresponding basic solution, and let $D(\pi, \pi_0)$ be a split disjunction violated by $\bar{x}$. The distance cut off by the split cut derived from B and $D(\pi, \pi_0)$ satisfies:*

$$(d(B, \pi, \pi_0))^2 = \frac{1}{\sum_{j \in J} \frac{1}{(\alpha_j(\pi, \pi_0))^2}} \tag{6}$$

Proof. Let $\gamma^T x \geq 1$, where $\gamma \in \mathbb{R}^n$, denote the intersection cut (5) derived from B and the disjunction $D(\pi, \pi_0)$. Then $\gamma_j = 0$ for $j \in N \setminus J$, $\gamma_j = \frac{1}{\alpha_j(\pi, \pi_0)}$ for $j \in J$ and $\gamma^T \bar{x} = 0$. Since γ is a normal vector to the intersection cut (5), it follows that $d(B, \pi, \pi_0)$ satisfies $\gamma^T(\bar{x} + d(B, \pi, \pi_0)\ \frac{\gamma}{\|\gamma\|_2}) = 1$. Isolating $d(B, \pi, \pi_0)$ in this expression gives the formula. □

The above formula gives a direct relationship between the coefficients in the intersection cut (5) and the distance cut off. This relationship is part of the motivation for the algorithm we develop in Sect. 3, where we fix the basis B of the linear programming relaxation, and attempt to improve the coefficients of the continuous variables in the MIG cuts.

2.4 Strengthening

The idea of cut strengthening is produce *another* cut that cuts off more of P. Such a strengthening procedure was introduced by Balas and Jeroslow [6] for split cuts.

Given a non-basic integer variable x_k, consider replacing $D(\pi, \pi_0)$ with $D(\pi^k(\delta), \pi_0^k(\delta))$ of the form $\pi^k(\delta) = \pi + \delta e_k$ and $\pi_0^k(\delta) = \lfloor\ (\pi^k(\delta))^T \bar{x} \rfloor$, where $\delta \in \mathbb{Z}$ and e_k is the k^{th} unit vector in $\mathbb{R}^n$. In other words, we are considering modifying π on the k^{th} component. Naturally, we assume that the disjunction $D(\pi, \pi_0)$ is violated, so that $\pi_0 = \lfloor \pi^T \bar{x} \rfloor$.

Observe that, since k is non-basic, we have $(\pi^k(\delta))^T \bar{x} = \pi^T \bar{x}$ and $\pi_0^k(\delta) = \pi_0$ for every integer δ. This means that $\epsilon(\pi, \pi_0) := \pi^T \bar{x} - \pi_0 = (\pi^k(\delta))^T \bar{x} - \pi_0^k(\delta) = \epsilon(\pi^k(\delta), \pi_0^k(\delta))$ for every integer δ. The following lemma shows that there is an optimal integer δ given by a closed form formula, i.e. a value of δ that gives the strongest split cut among the ones derived from the basis B and split disjunctions of the form $D(\pi^k(\delta), \pi_0)$, where $\delta \in \mathbb{Z}$:

Lemma 2 *Let B be a basis of LP, and let $\bar{x}$ be the corresponding basic solution. Also, let $D(\pi, \pi_0)$ be a split disjunction violated by $\bar{x}$, and let x_k be a non-basic integer constrained variable. Define disjunctions $D(\pi^k(\delta), \pi_0)$ by $\pi^k(\delta) = \pi + \delta e_k$ for $\delta \in \mathbb{Z}$, and let δ^* be defined as:*

$$\delta^* := \begin{cases} -\lfloor \pi^T r^k \rfloor, & \textit{if } \lceil \pi^T r^k \rceil - \pi^T r^k > \epsilon(\pi, \pi_0) \textit{ and} \\ -\lceil \pi^T r^k \rceil, & \textit{if } \lceil \pi^T r^k \rceil - \pi^T r^k \leq \epsilon(\pi, \pi_0). \end{cases} \tag{7}$$

Then the split cut derived from the disjunction $D(\pi^k(\delta^*), \pi_0)$ and B dominates the split cut derived from the disjunction $D(\pi(\delta), \pi_0)$ and B for every integer δ.

Proof. Observe that, since x_k is non-basic, we have $(\pi^k(\delta))^T r^j = \pi^T r^j$ for every $\delta \in \mathbb{Z}$ and every $j \in J \setminus \{k\}$. Using (4), it follows that, for any $\delta \in \mathbb{Z}$, the intersection cut derived from $D(\pi^k(\delta), \pi_0)$ and B only differs from the intersection cut derived from $D(\pi, \pi_0)$ and B in the coefficient on x_k. Hence, to finish the proof, we only need to prove that δ^* maximizes $\alpha_k(\pi^k(\delta), \pi_0)$. Observe that, since $r_k^k = 1$, we have $(\pi^k(\delta))^T r^k = (\pi + \delta e_k)^T r^k = \pi^T r^k + \delta$. Therefore, we may write:

$$\alpha_k(\pi^k(\delta), \pi_0) := \begin{cases} -\frac{\epsilon(\pi,\pi_0)}{\pi^T r^k + \delta} & \text{if } \pi^T r^k + \delta < 0, \\ \frac{1-\epsilon(\pi,\pi_0)}{\pi^T r^k + \delta} & \text{if } \pi^T r^k + \delta > 0, \\ +\infty & \text{otherwise.} \end{cases} \tag{8}$$

Clearly $\alpha_k(\pi^k(\delta), \pi_0)$ is maximized by either $\delta_f := -\lfloor \pi^T r^k \rfloor$ or $\delta_c := -\lceil \pi^T r^k \rceil$. Now we have $\alpha_k(\pi^k(\delta_c), \pi_0) < \alpha_k(\pi^k(\delta_f), \pi_0) \iff \frac{\lceil \pi^T r^k \rceil - \pi^T r^k}{\epsilon(\pi,\pi_0)} > \frac{\pi^T r^k - \lfloor \pi^T r^k \rfloor}{1-\epsilon(\pi,\pi_0)} \iff \lceil \pi^T r^k \rceil - \pi^T r^k > \epsilon(\pi, \pi_0)$.□

A split disjunction, for which the strengthening step (7) does *not* lead to a stronger split cut for *any* non-basic integer constrained variable x_k, is called a *strengthened* split disjunction. Lemma 2 implies that, for a split disjunction $D(\pi, \pi_0)$, it suffices to know the values of π on the components corresponding to basic variables. For each non-basic integer constrained variable x_k, the best value of π_k can be computed using (7).

Lemma 2 also implies that the coefficient $\frac{1}{\alpha_k(\pi,\pi_0)}$ on a non-basic integer constrained variable x_k is in the interval $[0, 1]$ in a strengthened split cut. This is because, for a strengthened split disjunction $D(\pi, \pi_0)$, and a non-basic integer constrained variable x_k, (7) gives $\pi^T r^k \in [-1, 1]$, which implies $\frac{1}{\alpha_k(\pi,\pi_0)} \in [0, 1]$. Since the basic integer constrained variables do not appear in a split cut, this demonstrates that the coefficient of *any* integer constrained variable in a strengthened split cut is in the interval $[0, 1]$.

2.5 Mixed integer Gomory cuts as split cuts

We now demonstrate that the mixed integer Gomory cuts that can be obtained from the simplex tableau (2) associated with B can also be obtained as intersection cuts by using (5) with an appropriate choice of the disjunction $D(\pi, \pi_0)$. This was first shown by Balas [2].

First we derive the MIG cut. Let $\sum_{j \in N} g_j x_j = d$ be an arbitrary equality satisfied by every feasible solution to MILP. Define $f_0 := d - \lfloor d \rfloor$ to be the fractionality of d, and let $f_j := g_j - \lfloor g_j \rfloor$ denote the fractionality of g_j for $j \in N_I$. Gomory [18] showed that the following inequality is also satisfied by every feasible solution of MILP:

$$\begin{array}{lclcl} \displaystyle\sum_{j \in N_I : f_j \leq f_0} \frac{f_j}{f_0} x_j & + & \displaystyle\sum_{j \in N_I : f_j > f_0} \frac{(1-f_j)}{(1-f_0)} x_j & + & \\ \displaystyle\sum_{j \in N \setminus N_I : g_j \geq 0} \frac{g_j}{f_0} x_j & - & \displaystyle\sum_{j \in N \setminus N_I : g_j < 0} \frac{g_j}{(1-f_0)} x_j & \geq & 1. \end{array} \tag{9}$$

Now consider the special case where equality $\sum_{j \in N} g_j x_j = d$ is a row of the simplex tableau (2). In this case, let x_i be a basic integer constrained variable such that $\bar{x}_i$ is fractional. We have $d = \bar{x}_i$ and $g_j = \bar{a}_{ij}$ for $j \in J$, $g_i = 1$, and $g_j = 0$ for $j \in B \setminus \{i\}$. The MIG cut is given by:

$$\begin{array}{lclcl} \displaystyle\sum_{j \in J \cap N_I : f_j \le f_0} \frac{f_j}{f_0} x_j & + & \displaystyle\sum_{j \in J \cap N_I : f_j > f_0} \frac{(1-f_j)}{(1-f_0)} x_j & + & \\ \displaystyle\sum_{j \in J \setminus N_I : g_j \ge 0} \frac{\bar{a}_{ij}}{f_0} x_j & - & \displaystyle\sum_{j \in J \setminus N_I : g_j < 0} \frac{\bar{a}_{ij}}{(1-f_0)} x_j & \ge & 1. \end{array}$$

The following lemma shows that this cut can be obtained from (5) by choosing an appropriate disjunction $D(\pi, \pi_0)$:

Lemma 3 *Let B be a basis of LP, and let $\bar{x}$ be the corresponding basic solution. Also, let x_i be a basic integer constrained variable, and suppose $\bar{x}_i$ is fractional. The MIG cut obtained from the row of the simplex tableau, in which x_i is basic, is given by the inequality $\sum_{j \in J} \frac{x_j}{\alpha_j(\pi^i, \pi_0^i)} \ge 1$, where $\pi_0^i := \lfloor \bar{x}_i \rfloor$, and for $j \in N_I$:*

$$\pi_j^i := \begin{cases} \lfloor \bar{a}_{ij} \rfloor & \text{if } j \in J \text{ and } f_j \le f_0, \\ \lceil \bar{a}_{ij} \rceil & \text{if } j \in J \text{ and } f_j > f_0, \\ 1 & \text{if } j = i \text{ and} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

Alternatively, the MIG cut can be obtained by first computing the intersection cut associated with the disjunction $D(e_i, \pi_0^i)$, and then applying the strengthening (7) *on the non-basic integer constrained variables.*

Proof. Let us compute $\alpha_j(\pi^i, \pi_0^i)$ for the above disjunction using formula (4), where $j \in J$. We have:

$$\epsilon(\pi, \pi_0) = (\pi^i)^T \bar{x} - \pi_0^i = \bar{x}_i - \lfloor \bar{x}_i \rfloor = f_0.$$

Using (3) and (10), we get

$$(\pi^i)^T r^j = \pi_i^i r_i^j - \pi_j^i r_j^j = \begin{cases} -f_j & \text{if } j \in N_I \text{ and } f_j \le f_0, \\ 1 - f_j & \text{if } j \in N_I \text{ and } f_j > f_0, \\ -\bar{a}_{ij} & \text{if } j \in J \setminus N_I. \end{cases} \tag{11}$$

Now $\alpha_j(\pi^i, \pi_0^i)$ follows from formula (4). This yields the MIG cut as claimed in the first part of the lemma. The second part of the lemma follows from Lemma 2, i.e. the disjunction given by (10) can be obtained from (7) by strengthening the disjunction $D(e_i, \pi_0^i)$. □

2.6 Split cuts, MIG cuts and the simplex tableau

We now give an alternative derivation of split cuts in terms of the simplex tableau. The starting point is the set of rows of the simplex tableau in which the basic variables are integer constrained:

$$\bar{x}_i = x_i + \sum_{j \in J} \bar{a}_{ij} x_j, \qquad i \in B \cap N_I, \tag{12}$$

Furthermore, for every non-basic integer constrained variable, we have the following trivial equation:

$$0 = x_j - x_j, \qquad j \in J \cap N_I. \tag{13}$$

Now, let $D(\pi, \pi_0)$ be an arbitrary split disjunction that is violated by $\bar{x}$. For every $k \in N_I$, put a weight of π_k on the corresponding equation, and sum all equations together. This gives:

$$\pi^T \bar{x} = \pi^T x - \sum_{j \in J} (\pi^T r^j) x_j.$$

The MIG cut derived from the above equation using (9) is exactly the same as the intersection cut (5) obtained from the disjunction $D(\pi, \pi_0)$. Note that the coefficients of the continuous variables x_j, $j \in J \backslash N_I$, in this MIG cut are proportional to $|\pi^T r^j|$. For example, when $\pi = ke_i$, computational experiments in [14] showed that the MIG cut tended to deteriorate with increasing values of the integer k when $J \setminus N_I \neq \emptyset$. This is mainly because of the increase in $|\pi^T r^j|$ for $j \in J \setminus N_I$. In the next section we present an algorithm for generating weights π that reduce $|\pi^T r^j|$ for $j \in J \setminus N_I$.

Note that $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$ and $\pi^T \bar{x} \notin \mathbb{Z}$ in the above discussion. Why do we not allow (π, π_0) to be nonintegral? The reason is that the resulting MIG inequality would not be a cut for the point $\bar{x}$ in general, since the MIG inequality might have nonzero coefficients on the basic variables.

3 Reducing the coefficients of the continuous variables in MIG cuts

In this section, we develop an algorithm that starts with the MIG cuts obtained from the optimal simplex tableau of LP, and then generates *another* set of split cuts that have better coefficients on the continuous variables.

For a basis B, let $J^I := J \cap N_I$ denote the set of non-basic variables that are integer constrained, and $J^C := J \setminus N_I$ the set of those that are continuous. For each integer constrained basic variable x_i, where $i \in B \cap N_I$, such that $\bar{x}_i$ is fractional, a split disjunction $D(\pi^i, \pi^i_0)$ generates the MIG cut associated with x_i (see Lemma 3):

$$\sum_{j \in J} \frac{x_j}{\alpha_j(\pi^i, \pi^i_0)} \geq 1.$$

Also, for every $i \in B \cap N_I$ for which $\bar{x}_i$ is integer, consider the disjunction $D(\pi^i, \pi^i_0)$, where $\pi^i = e_i$ and $\pi^i_0 = \bar{x}_i$, and e_i is the i^{th} unit vector in $\mathbb{R}^n$. Observe that this disjunction is not violated, and there is no corresponding split cut. We refer to these $|B \cap N_I|$ disjunctions as *MIG disjunctions* and we let $\Pi^G := \{D(\pi^i, \pi^i_0)\}_{i \in B \cap N_I}$.

3.1 Motivation

Is it possible to generate a new set of disjunctions, starting from the MIG disjunctions, that give stronger split cuts? To answer this question, we need to consider the delicate issue of how to measure the quality of a split cut. As mentioned in the introduction, an often used measure for cut quality is the distance cut off, i.e. the Euclidean distance between the hyperplane of the cut and the point that the cut is designed to cut off (see [4]). In Sect. 2.3, we showed that the distance cut off $d(B, \pi, \pi_0)$ satisfies the formula:

$$(d(B, \pi, \pi_0))^2 = \frac{1}{\sum_{j \in J} \frac{1}{(\alpha_j(\pi, \pi_0))^2}}.$$

Recall that $\frac{1}{\alpha_j(\pi,\pi_0)}$ for $j \in J$ is the coefficient of the non-basic variable x_j in the split cut derived from the basis B and the disjunction $D(\pi, \pi_0)$. Therefore it is natural to aim at split disjunctions $D(\pi, \pi_0)$ that increase the values of $\alpha_j(\pi, \pi_0)$ for $j \in J$. In the following, we analyze the factors that determine the size of $\alpha_j(\pi, \pi_0)$. Due to Lemma 2 in Sect. 2.4 on strengthening, we can assume that $D(\pi, \pi_0)$ is a strengthened split disjunction.

First suppose that x_j is integer constrained. Since $D(\pi, \pi_0)$ is a strengthened split disjunction, this implies $\frac{1}{\alpha_j(\pi,\pi_0)} \in [0, 1]$ no matter how the disjunction $D(\pi, \pi_0)$ is chosen. We do not attempt to increase these values of $\alpha_j(\pi, \pi_0)$.

On the other hand, if x_j is continuous, there is no upper bound on $\frac{1}{\alpha_j(\pi,\pi_0)}$. It therefore seems natural to try to modify the MIG disjunctions so as to increase the value of $\alpha_j(\pi, \pi_0)$ for $j \in J^C$. Recall that these values are given by:

$$\alpha_j(\pi, \pi_0) := \begin{cases} -\frac{\epsilon(\pi,\pi_0)}{\pi^T r^j} & \text{if } \pi^T r^j < 0, \\ \frac{1-\epsilon(\pi,\pi_0)}{\pi^T r^j} & \text{if } \pi^T r^j > 0, \\ +\infty & \text{otherwise.} \end{cases} \tag{14}$$

Thus $\alpha_j(\pi, \pi_0)$ depends on $\pi^T r^j$, i.e. how orthogonal π is to the extreme ray r^j, and on $\epsilon(\pi, \pi_0)$, i.e. how much $\bar{x}$ violates the split disjunction $D(\pi, \pi_0)$.

We try to increase the values $\alpha_j(\pi, \pi_0)$ for $j \in J^C$ by finding disjunctions $D(\pi, \pi_0)$ such that the values $|\pi^T r^j|$ are small for $j \in J^C$. Although the value of $\epsilon(\pi, \pi_0)$ is important in determining the size of $\alpha_j(\pi, \pi_0)$, we found this parameter hard to control. We have therefore treated it as a random noise.

The details of the reduction algorithm are discussed in the following sections. In Sect. 3.2, we describe the reduction algorithm as a modification routine for split disjunctions. In Sect. 3.3, we give another description of the reduction algorithm in terms of row operations of a certain matrix D. Our implementation of the reduction algorithm is discussed in Sect. 3.4. In Sect. 3.5 we prove a desirable property on the angles between the reduced rows of D.

3.2 The reduction algorithm: modifying the underlying split disjunctions

The algorithm has two parts:

(a) Modifying the MIG disjunctions in Π^G to obtain *another* set of split disjunctions that give split cuts with "better" coefficients on the continuous variables.

(b) Strengthening the split disjunctions obtained in (a) on the non-basic integer constrained variables.

Observe that for a given split disjunction $D(\pi, \pi_0)$, the coefficient on a continuous variable in the split cut derived from $D(\pi, \pi_0)$ and B does not depend on the components of π corresponding to non-basic variables, i.e. $\pi^T r^j = \sum_{k \in B} \pi_k r_k^j$. This is because $r_k^j = 0$ when x_j is continuous and x_k is integer constrained (see (3) of Sect. 2.1). Therefore, in part (a) of the algorithm, we can simplify the MIG disjunctions and work with the *simple* split disjunctions $\Pi^S := \{D(\bar{\pi}^i, \bar{\pi}_0^i)\}_{i \in B \cap N_I}$, where $\bar{\pi}^i = e_i$ and $\bar{\pi}_0^i = \lfloor \bar{x}_i \rfloor$. The set Π^S consists of the disjunctions $x_i \leq \lfloor \bar{x}_i \rfloor \vee x_i \geq \lceil \bar{x}_i \rceil$ for $i \in B \cap N_I$. The simple split disjunctions agree with the MIG disjunctions on the basic variables. Lemma 2 shows that the MIG disjunctions in Π^G can be obtained from the disjunctions in Π^S by strengthening.

We now describe part (a) of the reduction algorithm. Part (b) consists of applying Lemma 2 of Sect. 2.4 to the disjunctions obtained in (a). Let $\bar{\Pi}$ denote the current set of split disjunctions (where initially $\bar{\Pi} = \Pi^S$). In every iteration, a split disjunction $D(\pi, \pi_0)$ in the current set $\bar{\Pi}$ is considered and the algorithm tries to replace it with an improved split disjunction.

For our algorithm to be fast, we have adapted the following simple idea. To improve the disjunction $D(\pi, \pi_0)$, we use *another* disjunction $D(\pi', \pi'_0) \in \bar{\Pi}$, distinct from $D(\pi, \pi_0)$, and we consider disjunctions $D(\pi(\delta), \pi_0(\delta))$ of the form $\pi(\delta) = \pi + \delta\pi'$ and $\pi_0(\delta) = \lfloor (\pi(\delta))^T \bar{x} \rfloor$, where $\delta \in \mathbb{Z}$. If there exists $\delta \in \mathbb{Z}$ such that the disjunction $D(\pi(\delta), \pi_0(\delta))$ is "better" than the disjunction $D(\pi, \pi_0)$, the disjunction $D(\pi, \pi_0)$ is replaced by $D(\pi(\delta), \pi_0(\delta))$ in $\bar{\Pi}$.

The criterion for a "better" disjunction is the following. We find $\delta \in \mathbb{Z}$ that minimizes the following quantity:

$$f(\delta) := \sum_{j \in J^C} ((\pi(\delta))^T r^j)^2.$$

Since f is a quadratic convex function in δ, this minimization problem can be solved optimally by rounding. Specifically, we have $f(\delta) = g(\pi) + \delta^2 g(\pi') + 2\delta h(\pi, \pi')$, where $g(\pi) := \sum_{j \in J^C} (\pi^T r^j)^2$ and $h(\pi, \pi') := \sum_{j \in J_C} ((\pi')^T r^j)(\pi^T r^j)$. The optimal solution is either $\delta^* = \lfloor -\frac{h(\pi,\pi')}{g(\pi')} \rfloor$ or $\delta^* = \lceil -\frac{h(\pi,\pi')}{g(\pi')} \rceil$. If $f(\delta^*) < f(0) = \sum_{j \in J^C} (\pi^T r^j)^2$, the algorithm replaces the disjunction $D(\pi, \pi_0)$ by the disjunction $D(\pi(\delta^*), \pi_0(\delta^*))$ in $\bar{\Pi}$. This step is then iterated with new disjunctions $D(\pi, \pi_0)$ and $D(\pi', \pi'_0)$ until a stopping criterion is reached. We discuss the stopping criterion later.

3.3 The reduction algorithm in terms of row operations

We now give a description of the reduction algorithm in terms of row operations on a submatrix of the simplex tableau $\bar{A} := (\bar{a}_{ij})$. Let $B^I := B \cap N_I$ denote the basic variables that are integer constrained. Consider the rows of the simplex tableau corresponding to these integer constrained basic variables:

$$\bar{x}_i = x_i + \sum_{j \in J^I} \bar{a}_{ij} x_j + \sum_{j \in J^C} \bar{a}_{ij} x_j, \qquad i \in B^I, \tag{15}$$

where we have separated the non-basic integer constrained variables (J^I) from the non-basic continuous variables (J^C). Observe that the system (15) can be re-written in terms of the simple split disjunctions $\Pi^S = \{D(\bar{\pi}^i, \bar{\pi}_0^i)\}_{i\in B^I}$ introduced in the previous section:

$$(\bar{\pi}^i)^T\bar{x} =(\bar{\pi}^i)^T x - \sum_{j\in J^I}((\bar{\pi}^i)^T r^j)x_j - \sum_{j\in J^C}((\bar{\pi}^i)^T r^j)x_j, \qquad i \in B^I. \tag{16}$$

Now consider the operation of adding δ times one disjunction $D(\bar{\pi}^l, \bar{\pi}_0^l) \in \Pi^S$ to another disjunction $D(\bar{\pi}^k, \bar{\pi}_0^k) \in \Pi^S$, where $k, l \in B^I$ and $\delta \in \mathbb{Z}$. The new disjunction $D(\pi(\delta), \pi_0(\delta))$ is then given by $\pi(\delta) = \bar{\pi}^k + \delta\bar{\pi}^l$ and $\pi_0(\delta) = \lfloor(\pi(\delta))^T\bar{x}\rfloor$. Now, if we add δ times the l^{th} row of (16) to the k^{th} row of (16), we obtain:

$$(\bar{\pi}^i)^T\bar{x} =(\bar{\pi}^i)^T x - \sum_{j\in J^I}((\bar{\pi}^i)^T r^j)x_j - \sum_{j\in J^C}((\bar{\pi}^i)^T r^j)x_j, \qquad i \in B^I \setminus \{k\}, \tag{17}$$

$$(\pi(\delta))^T\bar{x} =(\pi(\delta))^T x - \sum_{j\in J^I}((\pi(\delta))^T r^j)x_j - \sum_{j\in J^C}((\pi(\delta))^T r^j)x_j. \tag{18}$$

Observe that the system (17)-(18) is of the same form as the system (16). Also, the part of the systems (16) and (17)-(18) corresponding to the continuous variables (J^C) contains all the information needed to evaluate the function $f(\delta) = \sum_{j\in J^C}((\pi(\delta))^T r^j)^2$ introduced in Sect. 3.2.

It follows that the reduction algorithm can be implemented as an algorithm that performs row operations on a matrix derived from the simplex tableau. Specifically, this matrix contains the coefficients of the continuous variables in the rows in which the basic variable is integer constrained. We now describe the reduction algorithm in this way.

WLOG we assume $B^I = \{1, 2, \ldots, |B^I|\}$ and $J^C = \{1, 2, \ldots, |J^C|\}$ (this is just a permutation of the rows and columns of the simplex tableau). Construct the matrix $D \in \mathbb{R}^{|B^I|\times|J^C|}$ by defining $d_{ij} := \bar{a}_{ij}$, i.e. d_{ij} is the coefficient on the continuous variable x_j in the row of the simplex tableau where the integer constrained variable x_i is basic. Let the vector $d_{i.}$ denote the i^{th} row of D.

In every iteration of the algorithm, we consider adding an integer multiple of one row of D to another row of D. Suppose we are considering adding a multiple of the l^{th} row of D to the k^{th} row of D. The problem we solve is:

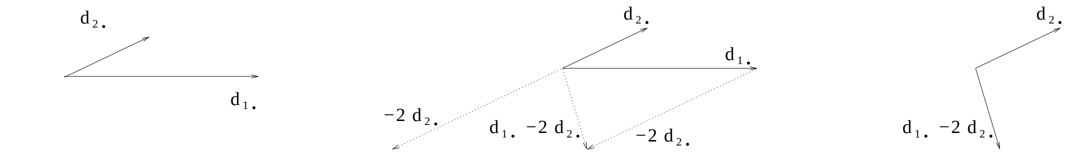
$$\min\{||d_{k.} + \delta d_{l.}||_2^2 : \delta \in \mathbb{Z}\}.$$

As in Sect. 3.2, the optimal solution δ^* to this problem is either $\lfloor -\frac{d_{k.}^T d_{l.}}{||d_{l.}||_2^2}\rfloor$ or $\lceil -\frac{d_{k.}^T d_{l.}}{||d_{l.}||_2^2}\rceil$. If δ^* satisfies $||d_{k.} + \delta^* d_{l.}||_2^2 < ||d_{k.}||_2^2$, the algorithm updates the k^{th} row of D to $d_{k.} := d_{k.} + \delta^* d_{l.}$.

The example of Figure 2 demonstrates how the algorithm works. In the example, the vector $d_{1.}$ is reduced by simply subtracting two times $d_{2.}$. Observe that the final vectors are more orthogonal than the initial vectors.

3.4 Implementation of the reduction algorithm

We now discuss the implementation details of the *reduction algorithm*.



(a) Two vectors $d_{1.}$ and $d_{2.}$ (b) Intermediate vectors (c) Final vectors

Figure 2: Using $d_{2.}$ to reduce $d_{1.}$

We first discuss the amount of work that has to be performed in each iteration. Given the matrix D, which we extract from the optimal simplex tableau, we first compute a $|B^I| \times |B^I|$ matrix of inner product between every pair $d_{k.}, d_{l.}$ of rows of the matrix D, where $k, l \in B^I$. Observe that this matrix also contains the norm of each of the rows of D on the diagonal. This gives all the information needed to compute δ^* for pairs $k, l \in B^I$. Recall that, for $k, l \in B^I$, δ^* is given by either $\lfloor -\frac{d_{k.}^T d_{l.}}{||d_{l.}||_2^2} \rfloor$ or $\lceil -\frac{d_{k.}^T d_{l.}}{||d_{l.}||_2^2} \rceil$. Once an improvement has been found, i.e. if $k, l \in B^I$ is such that δ^* satisfies $||d_{k.} + \delta^* d_{l.}||_2^2 < ||d_{k.}||_2^2$, we simply *update* the matrix of inner products. If $\tilde{d}_{k.} := d_{k.} + \delta^* d_{l.}$ improves $d_{k.}$, i.e. if $||\tilde{d}_{k.}||_2 < ||d_{k.}||_2$, the expression $\tilde{d}_{k.}^T d_{i.} = d_{k.}^T d_{i.} + d_{l.}^T d_{i.} \delta^*$ provides an update formula for the matrix of inner products.

We next discuss the sequence in which we consider pairs of rows of the matrix D for reduction. The algorithm first computes the reduction that can be obtained from each pair of rows $d_{k.}$, $d_{l.}$ of D. A sorted stack is maintained that contains those pairs of rows that give a reduction. In every iteration, the algorithm chooses the pair that gives the largest reduction. This process is then iterated.

The method, as described above, does not necessarily converge. To achieve convergence, we made the following adjustments. First, we introduced a tolerance for what is considered the zero vector. In our experiments, we treat any vector with norm less than $\epsilon := 10^{-5}$ as the zero vector, and we therefore do not use such a vector to reduce other vectors. The second tolerance we introduce is on what is considered a reduction. In our experiments, given any vector d, and an improved version of d, say $\tilde{d}$, i.e. we have $||\tilde{d}||_2 < ||d||_2$, we consider d reduced if $\frac{||\tilde{d}||_2}{||d||_2} < \rho := 0.95$. With these modifications, the algorithm converges because it keeps a fixed number of vectors and the norm of each vector can only be reduced a finite number of times by a factor ρ until it reaches ϵ.

The algorithm, as described above, can be implemented into a very fast reduction algorithm. The reduction algorithm did not use more than a few seconds on any problem in our computational experiment.

We summarize the algorithm:

Solve the LP relaxation and initialize D.

(a) Iterative Step: For each pair of rows $d_{k.}, d_{l.}$ of D with norm at least ϵ, compute δ^* and $\tilde{d}_{k.}$. Choose the pair that gives the largest reduction. If this reduction is better than ρ, replace $d_{k.}$ by $\tilde{d}_{k.}$ in D and repeat the Iterative Step. Otherwise, generate the set $\bar{\Pi}$ of disjunctions.

(b) Strengthen each disjunction $D(\pi, \pi_0) \in \bar{\Pi}$ by using Lemma 2 of Sect. 2.4 and compute the corresponding cuts using 5.

3.5 Angle between the reduced vectors

An important property of the above algorithm is that it leads to vectors $d_{i.}$ that are fairly orthogonal, like in the basis reduction algorithm of Lenstra, Lenstra and Lovász [20]. We now address this issue.

Lemma 4 *Let $i, k \in B^I$ be two nonzero rows of the matrix D. Then either:*

(i) $d_{i.}$ can be used to reduce the size of $d_{k.}$ or

(ii) $d_{k.}$ can be used to reduce the size of $d_{i.}$ or

(iii) The angle between $d_{i.}$ and $d_{k.}$ is between $60°$ and $120°$.

Proof. Assume that neither (i) nor (ii) hold. Since $d_{i.}$ can not be reduced by using $d_{k.}$, we have $||d_{i.} + \delta d_{k.}||_2^2 \geq ||d_{i.}||_2^2$ for all integers δ. In particular, for $\delta = -1$ and $\delta = 1$, we get $||d_{i.} - d_{k.}||_2^2 \geq ||d_{i.}||_2^2$ and $||d_{i.} + d_{k.}||_2^2 \geq ||d_{i.}||_2^2$. This implies that $-\frac{1}{2} \leq \frac{d_{i.}^T d_{k.}}{||d_{k.}||_2^2} \leq \frac{1}{2}$ and similarly $-\frac{1}{2} \leq \frac{d_{i.}^T d_{k.}}{||d_{i.}||_2^2} \leq \frac{1}{2}$. Thus $-\frac{1}{2} \leq \frac{d_{i.}^T d_{k.}}{||d_{i.}||_2 ||d_{k.}||_2} \leq \frac{1}{2}$, which implies (iii) since, given two non-zero vectors $d_{i.}$ and $d_{k.}$, the quantity:

$$\frac{d_{i.}^T d_{k.}}{||d_{i.}||_2 ||d_{k.}||_2}$$

is the cosine of the angle between the vectors $d_{i.}$ and $d_{k.}$. □

Lemma 4 demonstrates that the matrix D obtained at termination of the reduction algorithm has rows that are relatively orthogonal. Since the entries in matrix D are directly related to the coefficients in the split cuts that we generate, these cuts are likely to cut in directions that are also relatively orthogonal in the space of the continuous variables.

4 Computational results

We implemented the algorithm described in Sect. 3, and tested it on the problems in MIPLIB 3.0 [8]. These instances are available on the website www.caam.rice.edu/˜bixby/miplib/miplib.html. We used CPLEX 8.0 [15] to solve the LP and MILP problems.

We have included most problems available in MIPLIB 3.0. Some problems were excluded because they were too large for our experiments (dano3mip, fast0507, mod011, mkc, nw04 and rentacar). We also excluded three problems because they are too easy (mitre, mod010 and air03).

4.1 Performance of the reduction algorithm

We first discuss the performance of the reduction algorithm and investigate properties of the cutting planes it produces. The cutting planes obtained from the reduction algorithm are called *Reduce-and-Split cuts* (R & S cuts) in the remainder of the paper. It is possible that some (or all) of the R & S cuts are MIG cuts. In particular, this happens when the reduction algorithm does not change the coefficients on the continuous variables in a row of the simplex tableau from which a MIG cut is derived.

Table 1 contains statistics obtained when applying the reduction algorithm to the simplex tableau associated with the optimal basis to the LP relaxation. We have excluded problems air04, air05, bell5, enigma, harp2, khb05250 and modglob from this table. For these problems, the reduction algorithm did not produce any new cutting planes. This happens when either: (i) there are no continuous variables, (ii) the coefficients of the continuous variables are all zero, or (iii) the reduction algorithm cannot reduce the nonzero coefficients of the continuous variables. If a problem has inequality constraints, we treat the slack and surplus variables as continuous variables whose coefficients can potentially be reduced even if the problem is a pure integer program in the original variables.

As described earlier, our reduction algorithm first constructs a matrix D after which each row of D is iteratively replaced by a new row that has smaller size. We computed the size of the original matrix D, measured by the sum of the squared norms of the rows of D, and also the size of the matrix D' obtained at the termination of the reduction algorithm. The ratio between these two measures is recorded in the column of Table 1 headed "Reduction ratio". For 70% of the problems, our algorithm reduces the measure by a factor greater than two, and for 40% the reduction factor is greater than ten.

We now examine whether the reduction algorithm actually achieves reductions in the coefficients of the continuous variables. Increases in the coefficients on the continuous variables can happen, because the reduction algorithm does not consider the value $\epsilon(\pi, \pi_0)$. For every MIG cut obtained from the optimal basis of the LP relaxation, we computed the average coefficient on a continuous variable:

$$\frac{1}{|J^C|} \sum_{j \in J^C} \frac{1}{\alpha(\pi, \pi_0)}.$$

These values were then averaged over all MIG cuts. This gives the average coefficient on a continuous variable in the MIG cuts. We also computed the average coefficient on a continuous variable in the new cutting planes, i.e. those R & S cuts that are not MIG cuts. The column of Table 1 headed "Average coeff ratio (C)" contains the ratio between these two numbers. On most problems, our algorithm gives a significant decrease in the coefficients of the continuous variables. There are problems, however, where this does not happen. On problem gt2, for instance, the average coefficient of the continuous variables in the new R & S cuts is 63.88 times *higher* than in the MIG cuts even though the size of the matrix D was reduced to 0.88 times the original size.

We also investigated the properties of the R & S cuts on the integer constrained variables. We computed the average coefficient of an integer constrained variable for every MIG cut:

$$\frac{1}{|J^I|}\sum_{j\in J^I}\frac{1}{\alpha(\pi,\pi_0)}.$$

We then averaged these numbers over all MIG cuts to get the average coefficient of an integer constrained variable in the MIG cuts. We compared this number to the average coefficient of an integer constrained variable in those R & S cuts that are not MIG cuts. The ratio between these two numbers is recorded in the column of Table 1 headed "AVG coeff ratio (I)". For seven problems (egout, fixnet6, pp08a, qiu, stein27, stein45 and vpm1), all of the integer constrained variables are basic, which means that there are no integer constrained variables in the cuts. As can be seen from this column, the reduction algorithm often leads to an increase in the average coefficient of an integer constrained variable.

We now attempt to explain this increase. Our hypothesis is that the simplex tableau can contain many integer values. If the coefficient on an integer constrained variable in the simplex tableau is integer, the variable will not appear in the MIG cut that can be derived from this row of the simplex tableau. By taking integer combinations of the rows of the simplex tableau, it is likely that more fractional values are created. Most of the problems in MIPLIB 3.0 are described with matrices A that are sparse, i.e. many entries are equal to zero. Also, the non-zero coefficients of the matrix A are often integers. It seems likely that these properties can give a simplex tableau that has many integer entries.

To test our hypothesis, we computed the average *non-zero* coefficient on an integer constrained variable in the MIG cuts. This value was then compared with the average non-zero coefficient of an integer constrained variable in those R & S cuts that are not MIG cuts. The ratio between these two numbers is recorded in the column of Table 1 headed "Average nz coeff ratio (I)". Most of the numbers in this column are close to 1.00, and the average of *all* the numbers in the column is 1.015. We conclude that the non-zero coefficients of integer constrained variables are very similar in MIG and R & S cuts.

We observed that the number of integer values in the simplex tableau tends to decrease as cuts are added. After a few rounds of cutting planes have been added to the formulation, we found that the R & S and MIG cuts have a similar number of nonzero coefficients on the integer constrained variables.

In conclusion, our reduction algorithm often produces cuts that have much better coefficients on the continuous variables than the MIG cuts. However the coefficients do get worse on 20% of the problems. As for the coefficients of the integer constrained variables, they are mostly unaffected by the reduction algorithm but they can also get worse when the simplex tableau happens to contain a large number of integer values. We deal with these issues in the next section, where we design a cutting plane algorithm that tries to use the best R & S cuts.

4.2 Performance of the reduce-and-split cuts

We now test the performance of the R & S cuts in solving MILP's. We test whether replacing the MIG cuts with the R & S cuts gives a better cutting plane algorithm. We also design a hybrid cutting plane algorithm that attempts to use the best cuts of both classes.

Problem	*Reduction ratio*	*Average coeff ratio (C)*	*Average coeff* ratio (I)	*Average nz coeff* ratio (I)		*Problem*	*Reduction ratio*	*Average coeff ratio (C)*	*Average coeff ratio (I)*	*Average nz coeff* ratio (I)
10teams	0.07	0.04	0.91	0.98		misc06	0.35	0.20	1.07	0.70
arki001	0.00	0.02	6.54	2.36		misc07	0.09	0.15	1.04	1.02
bell3a	0.00	0.01	5.12	1.02		mod008	0.48	0.22	0.93	0.92
blend2	0.44	0.25	1.60	1.18		noswot	0.47	0.59	4.96	1.18
cap6000	0.94	0.02	1.07	1.07		p0033	0.01	0.84	1.81	1.14
danoint	0.26	2.79	0.86	0.86		p0201	0.13	0.09	1.23	1.00
dcmulti	0.45	0.92	1.32	1.41		p0282	0.22	1.35	2.33	1.95
dsbmip	0.01	0.09	2.15	1.17		p0548	0.00	0.02	1.49	0.63
egout	0.77	6.16	-	-		p2756	0.83	10.28	8.81	1.08
fiber	0.18	0.21	2.27	0.76		pk1	0.26	0.57	1.03	1.01
fixnet6	0.55	1.06	-	-		pp08a	0.74	0.46	-	-
flugpl	0.41	0.44	0.71	0.71		pp08acuts	0.56	0.22	1.07	0.76
gen	0.00	0.01	1.03	0.70		qiu	0.04	1.00	-	-
gesa2	0.05	0.49	1.59	0.96		qnet1	0.02	0.00	0.97	0.87
gesa2o	0.04	0.16	1.47	1.05		qnet1o	0.82	0.00	3.81	1.20
gesa3	0.03	2.26	1.65	1.07		rgn	0.05	0.13	0.54	0.92
gesa3o	0.03	0.13	1.07	0.74		rout	0.16	0.03	1.48	1.15
gt2	0.88	63.88	3.57	0.84		set1ch	0.82	0.07	10.92	1.21
lseu	0.13	0.53	1.18	0.88		seymour	0.06	0.01	0.96	0.79
markshare1	0.61	1.33	0.98	0.98		stein27	0.19	0.37	-	-
markshare2	0.94	1.06	0.88	0.88		stein45	0.01	0.08	-	-
mas74	0.00	0.02	0.76	0.76		swath	0.09	0.00	1.36	0.98
mas76	0.00	0.00	0.76	0.76		vpm1	0.84	0.20	-	-
misc03	0.08	0.24	1.00	1.00		vpm2	0.66	1.53	0.62	0.97

Table 1: Performance of the reduction algorithm

We compare three different cutting plane algorithms to strengthen the initial MILP formulation. In the first algorithm, we add 20 rounds of R & S cuts to the formulation. Each round consists of all R & S cuts that can be generated by applying the reduction algorithm to the current optimal basis. The LP obtained by adding these cuts is solved again to get a new optimal basis from which the next round of cuts is then generated. We call this cutting plane algorithm the *R & S algorithm*.

In the second cutting plane algorithm, we add the MIG cuts generated from the current optimal basis in each round of the cutting-plane algorithm. The algorithm is stopped when exactly the same number of cutting planes has been generated as were generated by the R & S algorithm. We call this cutting plane algorithm the *MIG algorithm*. In the last round of the MIG algorithm, we choose the MIG cuts that have the largest distance cut off. In all other rounds, we add all the MIG cuts that can be generated.

The third cutting plane algorithm uses both the MIG cuts and the R & S cuts. The algorithm tries to use the best of the MIG and R & S cuts while still adding roughly the same number of cuts in every round as the other two algorithms. We call this algorithm the *hybrid algorithm*. A typical iteration of this cutting plane algorithm can be described as follows. Let n_f denote the number of integer constrained variables that are fractional in the current optimal solution $\bar{x}$. The MIG algorithm would generate n_f cutting planes, and add them to the formulation. In the hybrid algorithm, we choose n_f cuts among the MIG and R & S cuts that have the largest distance cut off relative to $\bar{x}$. The cuts chosen are then added to the formulation, and the LP is re-optimized. As with the MIG algorithm, we stop the hybrid algorithm when exactly the same number of cutting planes has been added to the formulation as was generated by the R & S algorithm. In the last round of the hybrid algorithm, we choose the cuts (MIG or R & S cuts) that have the largest distance cut off.

We maintain a "cut pool", i.e. after each round of the cutting plane algorithm, we move inactive cuts into the cut pool for later use. Therefore, in every round, in addition to the new cuts that are generated from the optimal basis, we also add cutting planes from the cut pool that are violated again. Duplicate cuts arise when the reduction algorithm does not change a MIG cut. We deal with this issue by simply removing duplicate cutting planes, and duplicate cuts are not counted.

The cutting planes are generated from the values in an optimal simplex tableau, which are subject to numerical inaccuracies. Therefore, any split cut $d^T x \geq 1$ generated is modified to $d^T x \geq 1 - \epsilon$. We use $\epsilon = 10^{-10}$ in our implementation. We found that, with this choice of ϵ, the optimum solution was never cut off by the split cuts that we generated on all the test problems. An interesting research question is to develop a theory of the precision needed to guarantee that cuts are valid, but this is beyond the scope of this paper.

We tested the cutting planes in a cut-and-branch framework. After the cutting plane algorithm terminates, we add *all* cutting planes generated to the formulation and then solve with a branch-and-bound algorithm. The reason for adding the entire cut pool is to achieve a measure of the strength of *all* the cuts generated rather than just those that happen to be active at the end of the cutting plane algorithm. The branch-and-bound method is simulated with CPLEX's branch-and-cut algorithm with the options "no cuts", "no heuristic", "strong branching", and the optimal value of the MILP provided. Strong branching and providing the optimal value tend to keep the size of the branch-and-bound trees to a minimum. These choices are meant to reduce the variability in the number of nodes in the branch-and-bound tree due to uncontrollable factors, thus providing a more meaningful comparison between MIG, R & S and hybrid cuts.

Some problems in the MIPLIB 3.0 problem set are very difficult, requiring a prohibitive number of nodes in the branch-and-bound tree. We excluded those problems from this part of our experiments (arki001, danoint, harp2, markshare1, markshare2, mkc, noswot, set1ch, seymour and swath).

Table 2 contains the "best cases", where we observed a significant improvement in solution time or enumeration by using either the R & S or hybrid cuts over the MIG algorithm. The abbreviations "MIG", "R & S" and "Hybr" refer to the MIG, R & S and hybrid cutting plane algorithms respectively. The columns headed "Nodes MIG", "Nodes R & S" and "Nodes hybrid" contain the number of nodes in the branch-and-bound search tree in each case. The columns headed "Time MIG", "Time R & S" and "Time Hybr" contains the total time used to solve the problems (in seconds), including the time needed to generate the cuts. The column headed "Num Cuts" contains the number of cuts generated. As mentioned, we designed the three cutting plane algorithms so that the same number of cuts is used. Therefore, any difference in the performance of the three cutting plane algorithms is due to the difference in the quality of the cuts, but not in their number.

One possible measure of the quality of cuts is the number of nodes in the resulting branch-and-bound tree. Another is the *gap closed* by the cuts. The *integrality gap* for an MILP is the difference between the objective values of the LP relaxation and the MILP. By improving the LP relaxation using cutting planes, one can improve the integrality gap. The gap closed is the fraction of the original integrality gap that is closed by strengthening the formulation. The columns headed "Gap MIG", "Gap R & S" and "Gap Hybr" of Table 2 give the gap closed (in percentage) by the three cutting plane algorithms.

The problems dsbmip, flugpl and rgn were completely solved with the R & S algorithm while some enumeration was needed by the MIG algorithm. On problems bell5, gesa2o, pp08a, pp08acuts, rgn, and vpm1, the R & S cuts lead to a decrease in the solution time by more than a factor of ten. On the remaining problems, the improvement achieved was still very significant. Observe that, on all of the problems in Table 2, either the R & S algorithm or the hybrid algorithm closed the largest amount of integrality gap. The problems dsbmip and enigma do not have any integrality gap.

Table 3 contains the remainder of the problems listed in MIPLIB 3.0 [8]. On those problems, we did not see a substantial difference in the three approaches. On some problems the R & S cuts lead to smaller computing times, while on other problems they lead to larger computing times. It is interesting to observe that the MIG algorithm was clearly better only on the problem fixnet6. This indicates that the R & S cuts are better than the MIG cuts on average.

In many cases where the R & S cuts lead to more nodes than the MIG cutting plane algorithm, the amount of enumeration is very small. For such problems, the branching strategy, i.e. the way CPLEX decides to create subproblems, becomes important for the *actual* number of nodes, whereas the strength of the cutting planes is less significant. One would expect this to happen in those cases where the MIG cuts are already doing quite well. In such cases, the time spent on creating the R & S cuts is wasted.

Problem	*Nodes MIG*	*Nodes R & S*	*Nodes Hybr*	*Time MIG*	*Time R & S*	*Time Hybr*	*Gap MIG*	*Gap R & S*	*Gap Hybr*	*Num* Cuts
bell5	786808	868	1135	1826	1.72	2.06	91.47	90.05	94.60	337
dsbmip	20	0	15	3.77	2.60	2.70	-	-	-	60
enigma	795	494	414	2.52	1.79	1.65	-	-	-	56
flugpl	184	0	60	0.06	0.01	0.01	14.07	100.00	90.73	35
gesa2	743	116	97	86.10	56.75	44.96	46.14	96.78	98.13	1086
gesa2o	9145	75	37	1365	45.62	32.91	91.68	98.12	97.56	1060
mod008	1409	82	216	11.21	0.83	1.54	46.61	88.99	85.13	201
modglob	838528	306647	330135	559241	193649	224478	74.17	71.63	77.12	1124
p0033	159	297	35	0.22	0.46	0.10	72.44	79.27	88.48	199
p0201	163	111	65	24.10	25.88	8.10	52.25	74.93	50.92	312
pp08a	7467	745	1890	2486	290	737	83.12	91.99	91.62	754
pp08acuts	12347	1943	1010	5890	989	462	61.12	75.88	81.36	831
rgn	874	0	109	1.19	0.04	0.13	15.45	100.00	87.49	42
vpm1	7132	1	17	47.79	6.12	4.67	43.67	98.47	74.89	254
vpm2	38946	8073	4254	5479	1518	779	41.45	61.40	66.84	753

Table 2: The best cases

In conclusion, the Reduce-and-Split cuts improve the MIG cuts on average. On several problems, the Reduce-and-Split cuts lead to a significant decrease in solution time and amount of enumeration. A good cutting plane strategy is to use the Reduce-and-Split cuts in conjunction with the MIG cuts.

With regard to future work, the following two directions are worth investigating.

(1) Our reduction method may be adapted to improving the general 2-slope and 3-slope cuts developed by Gomory and Johnson [19]. These cuts, like the MIG cuts, have coefficients of the continuous variables that are linear functions of the tableau entries. Tableau row reduction may improve these cuts too.

(2) Following Balas and Perregaard [7] one may explore infeasible bases of the LP relaxation to improve split cuts. In this paper we improved the disjunction. It is still an open question to optimize simultaneously the basis and the disjunction to construct stronger Reduce-and-Split cuts.

References

[1] K. Andersen, G. Cornuéjols, and Y. Li. Split closure and intersection cuts, in W. Cook and A. Schulz, eds., 'Integer Programming and Combinatorial Optimization'. *Lecture Notes in Computer Science*, 2337:127–144, 2002.

[2] E. Balas. Intersection cuts - a new type of cutting planes for integer programming. *Operations Research*, 19:19–39, 1971.

[3] E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.

Problem	*Nodes MIG*	*Nodes R & S*	*Nodes Hybr*	*Time MIG*	*Time R & S*	*Time Hybr*	*Gap MIG*	*Gap R & S*	*Gap Hybr*	*Num Cuts*
10teams	181	170	866	1385	1358	5486	100.00	100.00	100.00	1749
air04	189	312	142	624	1397	494	10.53	12.73	10.75	468
air05	227	240	294	988	1242	1511	8.29	8.52	7.78	893
bell3a	35064	30857	26806	21.48	22.23	15.76	70.53	72.76	73.08	151
blend2	800	727	814	20.79	46.50	36.17	29.17	27.41	33.40	486
cap6000	3765	3480	3622	111	129	112	62.92	63.82	63.91	175
dcmulti	33	36	37	34.03	60.86	58.13	70.00	69.02	79.34	1064
egout	0	8	0	0.14	0.82	0.14	100.00	93.36	100.00	409
fiber	98	89	71	14.19	23.95	9.72	92.25	92.02	91.66	337
fixnet6	651	1337	1708	422	742	981	81.65	80.02	83.07	1028
gen	40	40	50	7.33	10.26	14.83	68.52	82.76	82.46	432
gesa3	19	16	13	11.68	31.19	13.73	67.55	91.99	92.77	628
gesa3o	31	23	28	12.96	38.69	19.22	68.12	90.31	89.14	663
gt2	24	29	6	0.37	0.77	0.30	100.00	99.59	100.00	328
khb05250	5	8	9	3.37	6.27	7.73	97.15	98.12	97.15	268
l152lav	79	86	93	19.66	26.13	17.57	18.53	29.40	21.84	148
lseu	273	659	287	3.02	7.25	3.08	72.41	65.07	73.27	291
mas74	1853449	2022892	2364995	59140	64297	78183	8.41	9.92	8.93	372
mas76	251767	242453	221915	3933	3532	3406	7.99	10.57	8.81	277
misc03	161	198	226	11.41	15.73	14.76	20.00	23.10	20.00	226
misc06	4	5	6	8.25	19.33	21.25	92.91	86.37	75.90	296
misc07	9398	10242	9386	670	880	788	0.72	2.51	0.72	155
p0282	33	29	33	3.05	3.28	2.78	27.44	39.13	27.35	549
p0548	15	29	16	5.01	12.52	10.78	95.28	75.02	93.89	1291
p2756	21	16	10	11.52	24.40	34.01	97.72	98.16	98.67	786
pk1	118906	108264	109450	5133	4707	4664	0.00	0.00	0.00	320
qiu	13375	21992	13133	15320	28015	15912	9.27	10.44	8.86	266
qnet1	31	49	37	102	204	125	30.10	50.90	44.70	941
qnet1o	28	42	38	67.11	140	95.33	69.21	69.70	71.63	718
rout	83800	139909	98576	24052	40780	28062	10.35	6.35	10.25	475
stein27	1208	1235	1224	14.50	14.55	14.38	0.00	0.00	0.00	350
stein45	14823	15303	14093	517	523	412	0.00	0.00	0.00	523

Table 3: The remaining cases

[4] E. Balas, S. Ceria, and G. Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42:1229–1246, 1996.

[5] E. Balas, S. Ceria, G. Cornuéjols, and N.R. Natraj. Gomory cuts revisited. *Operations Research Letters*, 19:1–9, 1996.

[6] E. Balas and R.G. Jeroslow. Strengthening cuts for mixed integer programs. *European Journal of Operations Research*, 4:224–234, 1980.

[7] E. Balas and M. Perregaard. A Precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming. *Mathematical Programming B*, 94:221–245, 2003.

[8] R. E. Bixby, S. Ceria, C.M. McZeal, and M.W.P. Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 1998.

[9] R. E. Bixby, Z. Gu, E. Rothberg, and R. Wunderling. Mixed integer programming: a progress report. In M. Grötschel, ed., "The sharpest cut: the impact of Manfred Padberg and his work". *MPS/SIAM Series on Optimization*, pages 309–326, 2004.

[10] A. Caprara and A. Letchford. On the separation of split cuts and related inequalities. *Mathematical Programming*, 94:279–294, 2003.

[11] S. Ceria, G. Cornuéjols, and M. Dawande. Combining and strengthening Gomory cuts, in E. Balas and J. Clausen, eds., 'Integer Programming and Combinatorial Optimization'. *Lecture Notes in Computer Science*, 920:438–451, 1995.

[12] W. Cook, R. Kannan, and A. Schrijver. Chvátal closures for mixed integer programs. *Mathematical Programming*, 47:155–174, 1990.

[13] G. Cornuéjols and Y. Li. A connection between cutting plane theory and the geometry of numbers. *Mathematical Programming*, 93:123–127, 2002.

[14] G. Cornuéjols, Y. Li, and D. Vandenbussche. K-cuts: a variation of Gomory mixed integer cuts from the LP tableau. *INFORMS Journal on Computing*, 15:385–396, 2003.

[15] CPLEX optimizer version 8.0, ILOG.

[16] H. Crowder, E. Johnson, and M. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31:803–834, 1983.

[17] J.E. Eckstein and M. Nediak. Depth-optimized convexity cuts. Technical report, RUTCOR, Rutgers, February 2003.

[18] R.E. Gomory. An algorithm for the mixed integer problem. Technical Report RM-2597, The Rand Corporation, Santa Monica, CA, 1960.

[19] R.E. Gomory and E.L. Johnson. T-space and cutting planes. *Mathematical Programming B*, 96:341–375, 2003.

[20] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[21] G.L. Nemhauser and L.A. Wolsey. A recursive procedure for generating all cuts for 0-1 mixed integer programs. *Mathematical Programming*, 46:379–390, 1990.