Local cuts for mixed-integer programming

Vašek Chvátal
Department of Computer Science and Software Engineering
Concordia University, Canada

William Cook*
School of Industrial and Systems Engineering
Georgia Institute of Technology

Daniel Espinoza†
Department of Industrial Engineering
Universidad de Chile

August 10, 2009

Abstract

A general framework for cutting-plane generation was proposed by Applegate et al. in the context of the traveling salesman problem. The process considers the image of a problem space under a linear mapping, chosen so that a relaxation of the mapped problem can be solved efficiently. Optimization in the mapped space can be used to find a separating hyperplane if one exists, and via substitution this gives a cutting plane in the original space. We apply this procedure to general mixed-integer programming problems, obtaining a range of possibilities for new sources of cutting planes.

1 Introduction

Consider a mixed-integer set

$$P_{IP} = \{x \in \mathbb{R}^n : \; Ax = d; \; l \leq x \leq u; \; x_j \in \mathbb{Z}, \; \forall j \in I\}, \tag{1}$$

where $A \in \mathbb{Q}^{m \times n}$, $d \in \mathbb{Q}^m$, $l \in (\mathbb{Q} \cup \{-\infty\})^n$, $u \in (\mathbb{Q} \cup \{+\infty\})^n$, and $I \subseteq \{1, \ldots, n\}$. As usual, $\mathbb{Z}$, $\mathbb{Q}$, and $\mathbb{R}$ denote the integer, rational, and real numbers, respectively. Given an objective vector $c \in \mathbb{Q}^n$, the problem of maximizing $c^T x$ subject to $x \in P_{IP}$ is a *mixed-integer programming* (MIP) problem. A linear relaxation

$$P_{LP} = \{x \in \mathbb{R}^n : \; Ax = d; \; A'x \leq d'; \; l \leq x \leq u\} \tag{2}$$

*Supported by NSF Grant CMMI-0726370 and ONR Grant N00014-03-1-0040.
†Supported by FONDECYT Grant 1070749 and ICM Grant P05-004F.

of P_{IP} is obtained by dropping the integer restrictions on the variables x_j for $j \in I$, and possibly adding a system of inequalities $A'x \leq d'$ that is satisfied by all vectors in P_{IP}. The additional inequalities $A'x \leq d'$ are called *cutting planes*, or *cuts*.

In the *cutting-plane method*, cuts are added in an iterative fashion, selecting inequalities that are valid for P_{IP}, but violated by an optimal solution x^* to the linear programming (LP) problem $\max(c^T x : x \in P_{LP})$. The cutting-plane method is combined with a branch-and-bound search in a hybrid algorithm, known as branch-and-cut, that is the most successful MIP solution approach to date.

Given a basic optimal solution x^* to $\max(c^T x : x \in P_{LP})$, an elegant technique of Gomory [20] efficiently produces a cutting plane whenever $x^* \notin P_{IP}$. Indeed, the Gomory mixed-integer (GMI) cuts found by this method are one of the most important practical sources of MIP cutting planes, as reported in computational tests by Bixby et al. [11]. In their study, effectiveness is measured by the improvement in the overall running time of the branch-and-cut algorithm for solving a specific large collection of MIP instances. In other studies, effectiveness is measured as the increase in the lower bound produced by the optimal objective value of the LP relaxation problem. With either measure, the additional value of GMI cuts tends to decrease as more and more cuts are added. The current practical reply to this decreasing performance is to use a variety of techniques for producing cuts in MIP solvers. Classes of MIP cutting planes currently adopted in leading codes include GMI cuts [9], mixed-integer-rounding (MIR) inequalities [27], knapsack covers [18, 23], flow covers [22], lift-and-project cuts [8] and $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts [16].

An alternative approach to cutting-plane generation was proposed by Applegate et al. [5] in the context of the traveling salesman problem (TSP). This work introduced a *local-cuts procedure* that relies on the equivalence of optimization and separation to get cuts from small (graphical) TSP instances that are the result of linear mappings of the original problem. The general process was subsequently adopted by Buchheim et al. [14, 15] in the solution of constrained quadratic 0-1 optimization problems and by Althaus et al. [4] in the solution of Steiner-tree problems.

The goal of the current paper is to develop the local-cuts paradigm for general mixed-integer programming. We begin in Section 2 by presenting a standard LP-based method for generating MIP cuts using an oracle description of a relaxation of the original problem. In Section 3 we show how to obtain facets, or high-dimensional faces, from valid inequalities produced by the cut-generation process, generalizing the tilting procedure of Applegate et al. This is followed, in Section 4, by a discussion of linear mappings to simplify the mixed-integer set P_{IP}. In Section 5 we present the overall framework for local cuts for general MIP problems, and discuss results of a computational study using a simple choice of a mapping function.

2 Separation via optimization

Let $P \subseteq \mathbb{R}^n$ be a rational polyhedron and let $x^* \in \mathbb{Q}^n$. The *separation problem* for (P, x^*) is to find an inequality $a^T x \leq b$ that is valid for P but violated by x^*, if such an inequality exists. A fundamental result in LP theory is the polynomial-time equivalence of separation and optimization for rational polyhedra, via the ellipsoid method. This theorem and its many combinatorial applications are discussed in Grötschel et al. [21]. In this context, polyhedra are represented implicitly. For example, the convex hull of a mixed-integer set P_{IP} can be represented by $Ax = d,\ l \leq x \leq u$ and $x_i \in \mathbb{Z}, \forall i \in I \subseteq \{1, \ldots, n\}$, rather than by an explicit linear description of the polyhedron.

We say that *OPT* is an *optimization oracle* for P if, for any $c \in \mathbb{Q}^n$, it asserts that P is the empty set, or provides $x^* \in P \cap \mathbb{Q}^n$ such that $c^T x^* \geq c^T x$ for all $x \in P$, or provides a ray $r^* \in \mathbb{Q}^n$ of P such that $c^T r^* > 0$. (A vector r is called a ray of the polyhedron P if there exists $x \in P$ such that $x + \lambda r \in P$ for all positive scalars λ.) The output of the oracle is of the form $(status, \beta, y)$, where *status* is one of `empty`, `unbounded` or `optimal`; β contains the optimal value of $\max(c^T x : x \in P)$ if the problem has an optimal solution; and y contains the optimal solution or an unbounded ray if the status is `optimal` or `unbounded`, respectively.

An optimization oracle can be used as a practical vehicle for solving the separation problem without resorting to the ellipsoid method (which can be overly time-consuming in this context). This idea was championed by Boyd [12, 13], who introduced a cutting-plane technique for MIP instances where the variables are restricted to take on 0 or 1 values. Boyd considers single-row relaxations, where his access to the corresponding polytope is via an optimization oracle, implemented as a dynamic-programming algorithm for 0-1 knapsack problems. His separation method is a variant of the simplex algorithm, and he calls the separating inequalities Fenchel cutting planes.

In our work on general MIP instances, we adopt a straightforward approach, based on an LP formulation of the separation problem. Let us write the rational polyhedron P as $P_g + P_r$, where P_g is a bounded polyhedron having vertices $\{g^i : i \in I_g\} \subset \mathbb{Q}^n$ and P_r is a convex cone generated by rays $\{r^i : i \in I_r\} \subset \mathbb{Q}^n$; here I_g and I_r are finite sets, allowing us to index the vertices and rays. A given $x^* \in \mathbb{Q}^n$ is an element of P if and only if there is a solution (λ^g, λ^r) to the system

$$\sum_{i \in I_g} \lambda_i^g g^i + \sum_{i \in I_r} \lambda_i^r r^i = x^*,\ e^T \lambda^g = 1,\ \lambda^g, \lambda^r \geq 0, \tag{3}$$

where $e \in \mathbb{Q}^n$ denotes the vector of all ones. By duality, system (3) has no solution if and only if there exist $a \in \mathbb{Q}^n$ and $b \in \mathbb{Q}$ such that

$$\begin{array}{rll} a^T g^i - b & \leq 0, & \forall i \in I_g, \\ a^T r^i & \leq 0, & \forall i \in I_r, \\ a^T x^* - b & > 0. & \end{array} \tag{4}$$

Any cut that separates x^* from P corresponds to a solution of (4). To choose among these possible cuts, we formulate the problem of maximizing the violation subject to the normalization $\|a\|_1 = 1$, that is, we set the L_1-norm of a to 1. This is a technique described in a number of studies, for example, Balas et al. [8]. The resulting LP model is

$$\begin{array}{rll} & \max \quad a^T x^* - b & \\ s.t. & a^T g^i - b \leq 0, & \forall i \in I_g, \\ & a^T r^i \leq 0, & \forall i \in I_r, \\ & a - u + v = 0,\ e^T(u+v) = 1,\ u, v \geq 0. & \end{array} \tag{5}$$

Its LP dual can be written as

$$\begin{array}{rl} & \min \quad s \\ s.t. & \sum_{i \in I_g} \lambda_i^g g^i + \sum_{i \in I_r} \lambda_i^r r^i + w = x^*, \\ & e^T \lambda^g = 1, \\ & -w + se^T \geq 0,\ w + se^T \geq 0,\ \lambda^r, \lambda^g \geq 0. \end{array} \tag{6}$$

The LP problem (6) has $3n+1$ constraints other than bounds on individual variables. If we choose to minimize $\|a\|_1$ subject to the constraint $a^T x^* = b+1$, then we obtain the following formulation for the problem

$$\begin{array}{lcll} & \min & e^T(u+v) & \\ s.t. & a^T g^i - b \leq 0 & & \forall i \in I_g, \\ & a^T r^i \leq 0 & & \forall i \in I_r, \\ & a - v + u = 0,\ a^T x^* - b = 1,\ u, v \geq 0, & & \end{array} \tag{7}$$

whose dual is

$$\begin{array}{c} \max \quad s \\ s.t. \quad s x^* - \sum\limits_{i \in I_g} \lambda_i^g g^i - \sum\limits_{i \in I_r} \lambda_i^r r^i + w = 0, \\ -s + e^T \lambda^g = 0, \\ \lambda^g, \lambda^r \geq 0,\ -e \leq w \leq e. \end{array} \tag{8}$$

Note that (8) has only $n+1$ constraints other than bounds on individual variables. Problem (8) is trivially feasible (the all-zero vector is always a solution); if (8) has an optimal solution, its dual gives us a separating inequality for P and x^*; if (8) is unbounded, then the unbounded ray provides us with a decomposition of x^* into elements of P_g and P_r, thus yielding a proof that $x^* \in P$. Finally, note that problems (5) and (7) are essentially the same, in the sense that any optimal solution of (5) is an optimal solution of (7) (after scaling) and any optimal solution of (7) is an optimal solution for (5) (after scaling).

The large number of variables in problem (8) can be handled by employing a column-generation technique. To begin, we select any (possibly empty) subsets of points and rays in P, with index sets I_g and I_r, respectively. We then solve (8) with this new I_g and I_r. If the LP problem is unbounded, then we have a proof that $x^* \in P$. Otherwise, we obtain a tentative inequality $a^T x \leq b$ that is violated by x^*. The optimization oracle can be called to check whether $P \subseteq \{x : a^T x \leq b\}$ by maximizing $a^T x$ over P. If $P \subseteq \{x : a^T x \leq b\}$, we return with the inequality $a^T x \leq b$. Otherwise, we either find a point $g^j \in P$ such that $a^T g^j > b$ and add j to I_g, or we find a ray r^j of P such that $a^T r^j > 0$ and add j to I_r, and repeat the process. A pseudo-code for the method is given in Algorithm 1. We apply Algorithm 1 in our local-cuts procedure, after mapping to a space where an efficient optimization oracle is available.

3 Obtaining high-dimensional faces

Consider a rational polyhedron $P \neq \emptyset$, a rational vector $x^* \notin P$, and a separating inequality

$$a^T x \leq b \tag{9}$$

found by Algorithm 1. We describe an algorithm that transforms (9) into an inequality defining a facet of P. The method extends the tilting algorithm for bounded polyhedra given by Applegate et al. [5] in the context of the TSP.

Let $P_o \subseteq P$ be the set of points found in Algorithm 1 satisfying (9) at equality; the assumptions imply $P_o \neq \emptyset$. For arbitrary $y_o \in P$, define $P^\perp$, the orthogonal complement of P, as

$$P^\perp := \left\{ x \in \mathbb{R}^n : x^T (y - y_o) = 0, \forall y \in P \right\}.$$

Algorithm 1 Separation through optimization oracle $\texttt{SEP}(OPT, x^*, I_g, I_r)$

Require: OPT is an optimization oracle for P, x^* is a point to be separated from P,
I_g indexes an initial set of points in P, I_r indexes an initial set of rays of P.

loop
  Solve (8) over I_g and I_r.
  if (8) is unbounded **then**
    return $x^* \in P$.
  let a, b be an optimal dual solution to (8)
  $(status, \beta, y) \leftarrow OPT(a)$.
  if $status = \texttt{unbounded}$ **then**
    add y to the set of rays indexed by I_r.
  else if $status = \texttt{optimal}$ and $\beta > b$ **then**
    add y to the set of points indexed by I_g.
  else
    return $x^* \notin P$, (a, b)

Let $P_o^\perp := \{p_1, \ldots, p_r\}$ be a generating set for a subspace of the linear space $P^\perp$, and let $\bar{x}$ be a point in P such that $a^T\bar{x} < b$. If $P_o^\perp$ generates all of $P^\perp$, then (9) defines a facet of P if and only if the system

$$w(p_i^T y_o) + v^T p_i = 0, \qquad \forall p_i \in P_o^\perp, \tag{10a}$$

$$w - v^T x_i = 0, \qquad \forall x_i \in P_o, \tag{10b}$$

$$w - v^T \bar{x} = 0, \tag{10c}$$

$$(v, w) \in [-1, 1]^{n+1} \tag{10d}$$

has as unique solution the all-zero vector. Problem (10) can be interpreted as: Is there a hyperplane that passes through $P_o \cup \{\bar{x}\}$ and is orthogonal to the set of equations defining P? If the points in P_o define a facet of P, then the answer is no, that is, the only solution to (10) is the all-zero vector. Note that condition (10d) is not really needed, but it is included to make the feasible region of (10) a compact set; on the other hand, condition (10c) ensures $P \not\subseteq \{x : a^T x = b\}$, that is, (9) defines a proper face of P.

The idea is to iteratively build $P_o, P_o^\perp$ and candidate inequalities using the conditions in (10). For that, we start with P_o as the active points on our current candidate inequality, and with $P_o^\perp = \emptyset$ (or any previously known subset of $P_o^\perp$). The facet-finding algorithm ensures that at every iteration either the dimension of the candidate sets P_o or $P_o^\perp$ is increased, while possibly modifying the current inequality, or it proves that (9) defines a facet of P, or it proves that (9) is valid for P as an equation. We do this in such a way that all access to P is through an optimization oracle OPT.

To determine if our current inequality $a^T x \leq b$ defines a proper face of P, we maximize $-a^T x$ over P. If the LP problem is unbounded, then we easily find a point $\bar{x} \in P$ such that $a^T\bar{x} < b$; if, on the other hand, the problem has an optimal solution with value different from $-b$, then such an optimal solution provides us with the sought $\bar{x} \in P$; otherwise, the optimal value is $-b$, thus proving that $a^T x = b$ is a valid equation for P that is violated by x^*.

If $a^T x \leq b$ defines a proper face of P, then, using $\bar{x}$, we check if the only solution to (10) is the all-zero vector. If this is the case, then we have a certificate that $a^T x \leq b$ defines a facet of P.

Otherwise, $(v, w, \bar{x})$, with $v \neq 0$, certifies that the current inequality is not facet-defining.

Suppose we have a non-facet certificate $(v, w, \bar{x})$. In this case we would like to increase the dimension of P_o or of $P_o^{\perp}$. The idea we use is to *tilt* our current inequality $a^T x \leq b$, using as pivot the set P_o, and using as rotating direction the vector (v, w), until we touch the border of P, identifying a new affinely independent point. To illustrate this method, consider $P' := \{(y,z) \in \mathbb{R}^2 : \exists x \in P, y = a^T x, z = v^T x\}$, as given in Figure 1. In this example, rotating $a^T x \leq b$ produces the

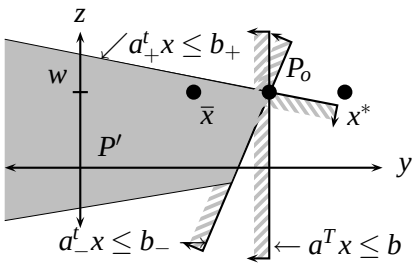

Figure 1: The gray area represents P'; the points $\bar{x}, P_o$ and x^* refer to their projection into P'.

inequalities $a_+^T x \leq b_+$ and $a_-^T x \leq b_-$, indicated in the figure. If we restrict ourselves to moving in this two-dimensional space, then these two inequalities are the only ones we can obtain by rotating $a^T x \leq b$.

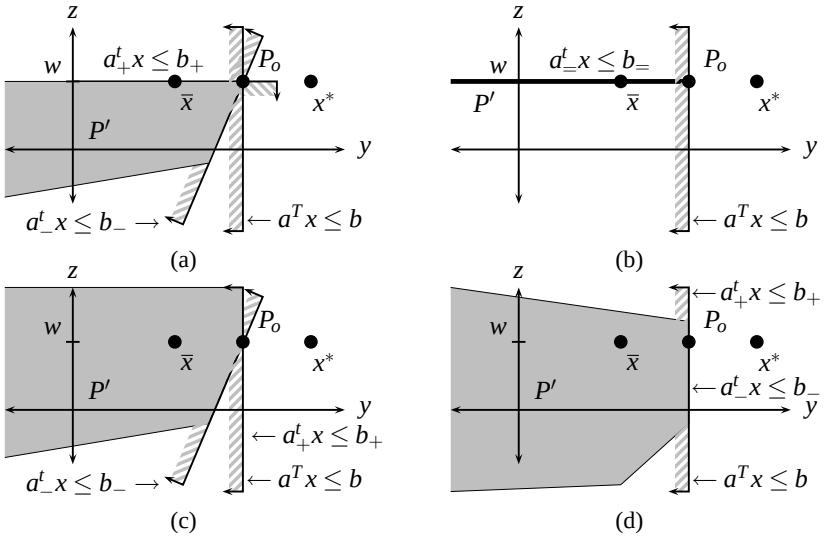

Figure 2: Possible ill-behaving outcomes for the mapping of P.

We must bear in mind that Figure 1 is just one possible outcome for P'. Indeed, Figure 2 shows four ill-behaving cases. Figure 2(a) is an example where one of the resulting inequalities coincides with $v^T x \leq w$; Figure 2(b) shows an example where $v^T x = w$ is a valid equation for P, giving us a new vector p to add to $P_o^{\perp}$; Figure 2(c) shows an example where one side of the tilting is in fact our original inequality $a^T x \leq b$; and Figure 2(d) shows an example where both sides of the tilting are our original inequality.

We provide a tilting algorithm in Section 3.1, but in the meantime, let us assume that we have a tilting routine with the following characteristics:

Condition 1 (Abstract Tilting Procedure: $\texttt{TILT}(a, b, v, w, \bar{x}, P_o, OPT)$)**.**

Input *The input of the algorithm should satisfy all of the following:*

- $a^T x \leq b$ *defines a face of* P *and* $P_o \subset P$ *is a set of points satisfying* $a^T x = b$*.*
- $v^T x \leq w$ *is an additional inequality such that* $v^T x = w$ *for all* $x \in P_o$*.*
- $\bar{x} \in P$ *is such that* $a^T \bar{x} < b$ *and* $v^T \bar{x} = w$*.*
- OPT *is an optimization oracle for* P*.*

Output *The output should be:*

- $(v', w', \bar{x}')$ *where* (v', w') *is a non-negative combination of* (v, w) *and of* (a, b)*,* $\max(v'^T x : x \in P) = w'$*, and* $\bar{x}'$ *belongs to* P *and satisfy both* $v'^T \bar{x}' = w'$ *and is affinely independent from* P_o*.*

If we have a non-facet certificate $(v, w, \bar{x})$ for the inequality $a^T x \leq b$ with point set P_o and equation set $P_o^{\perp}$, then we can obtain both a_+, b_+ and a_-, b_- with the calls

$$(a_+, b_+, \bar{x}_+) = \texttt{TILT}(a, b, v, w, \bar{x}, P_o, OPT), \quad \text{and}$$

$$(a_-, b_-, \bar{x}_-) = \texttt{TILT}(a, b, -v, -w, \bar{x}, P_o, OPT).$$

With this information, we can finish our facet procedure. If $(v_+, w_+) = (v, w)$, and $(v_-, w_-) = (-v, -w)$, then we are in the situation depicted in Figure 2(b), and we can add v to $P_o^{\perp}$, increasing its dimension by one. In any other situation we have two (possibly identical) inequalities, and we pick the one which is most violated by x^*[1]; assuming that a_+, b_+ is the most violated one, we replace a, b with a_+, b_+ and add $\bar{x}_+$ to P_o. Note that the newly added point increases the dimension of P_o by one.

Since at every step we either increase the dimension of $P_o^{\perp}$ or increase the dimension of P_o, the algorithm performs at most n iterations, and in every iteration we produce (or keep) a separating inequality. An outline of the complete method is given in Algorithm 2.

3.1 Solving the tilting problem

We now describe an algorithm that performs the tilting procedure satisfying Condition 1. We assume that we have a valid inequality $ax \leq b$ for P, a non-empty, finite set $P_o \subset P$ for which every element $x \in P_o$ satisfies $a^T x = b$. We also have another inequality (although it might not be valid for P) $v^T x \leq w$ and a point $\bar{x} \in P$, such that all $x \in P_o$ and $\bar{x}$ satisfy $v^T x = w$ and such that $a^T \bar{x} < b$. We show how to obtain a_+, b_+; the procedure for a_-, b_- is completely analogous.

Our objective is to find a valid inequality $v'^T x \leq w'$ for P and a point $\bar{x}'$ that is affinely independent from P_o and is such that $v'^T \bar{x}' = w'$. The idea is to use $v^T x \leq w$ as our candidate output constraint, and $\bar{x}$ as our candidate for an affinely independent point, thus, we only need to check if $\max(v^T x : x \in P) = w$.

If we maximize $v^T x$ over P and there is an optimal solution with value w, then we are in the situation depicted by Figure 2(a) or Figure 2(b). In this case, we return $v^T x \leq w$ as our tilted inequality and report $\bar{x}$ as our new affinely independent point.

[1] It is easy to see that at least one of them must be violated by x^* if the original $ax \leq b$ was violated by x^*

Algorithm 2 FACET$(a,b,x^*,P_o,P_o^\perp,OPT)$

Require: $a^Tx \le b$ is valid for P and $a^Tx^* > b$; $\emptyset \neq P_o \subset P$ is such that $a^Tx = b$ for all $x \in P_o$; $P_o^\perp \subset P^\perp$; OPT is an optimization oracle for P.

1: $x_o \leftarrow x \in P_o$ `/* select some point of` P_o`. */`
2: **loop**
3: `/* find proper face certificate */`
4: $(status, \beta, y) \leftarrow OPT(-a)$.
5: **if** $status =$ `optimal` and $\beta = -b$ **then**
6: **return** (`equation`, a,b) `/*` $P \subseteq \{x : a^Tx = b\}$ `*/`
7: **else if** $status =$ `unbounded` **then**
8: $\bar{x} \leftarrow y + x_o$.
9: **else**
10: $\bar{x} \leftarrow y$.
11: `/* get (non-)facet certificate */`
12: **if** $(0,0)$ is the unique solution for Problem (10) **then**
13: **return** (`facet`, a,b)
14: $(v,w) \leftarrow$ a non-trivial solution for Problem (10).
15: $(a_+,b_+,\bar{x}_+) \leftarrow$ TILT$(a,b,v,w,\bar{x},P_o,OPT)$.
16: $(a_-,b_-,\bar{x}_-) \leftarrow$ TILT$(a,b,-v,-w,\bar{x},P_o,OPT)$.
17: `/* update` $a,b,P_o,P_o^\perp$ `*/`
18: **if** $(a_+,b_+) = (-a_-,-b_-) = (v,w)$ **then**
19: $P_o^\perp \leftarrow P_o^\perp \cup \{v\}$. `/* grow dimension of` $P_o^\perp$ `*/`
20: **else**
21: $\lambda_+ \leftarrow \left(a_+^Tx^* - b_+\right)/\|a_+\|$.
22: $\lambda_- \leftarrow \left(a_-^Tx^* - b_-\right)/\|a_-\|$.
23: **if** $\lambda_+ > \lambda_-$ **then**
24: $(a,b) \leftarrow (a_+,b_+)$.
25: $P_o \leftarrow P_o \cup \{\bar{x}_+\}$. `/* grow dimension of` P_o `*/`
26: **else**
27: $(a,b) \leftarrow (a_-,b_-)$,
28: $P_o \leftarrow P_o \cup \{\bar{x}_-\}$. `/* grow dimension of` P_o `*/`

If $\max(v^T x : x \in P)$ is unbounded, then we are in the situation depicted by Figure 2(c) or by Figure 1. In this case, we have a ray r of P returned by the optimization oracle. Since $a^T x \leq b$ is a valid inequality for P, we have $a^T r \leq 0$. Let $x' = x + r$, where x is any point in P_o, note that x' is affinely independent from P_o, since $v^T x = w$ for all points in P_o and $v^T r > 0$. If, on the other hand, $\max(v^T x : x \in P)$ exists, then define x' as the optimal solution to $\max(v^T x : x \in P)$ that is returned by the oracle; in this case, we have again that x' is affinely independent from P_o.

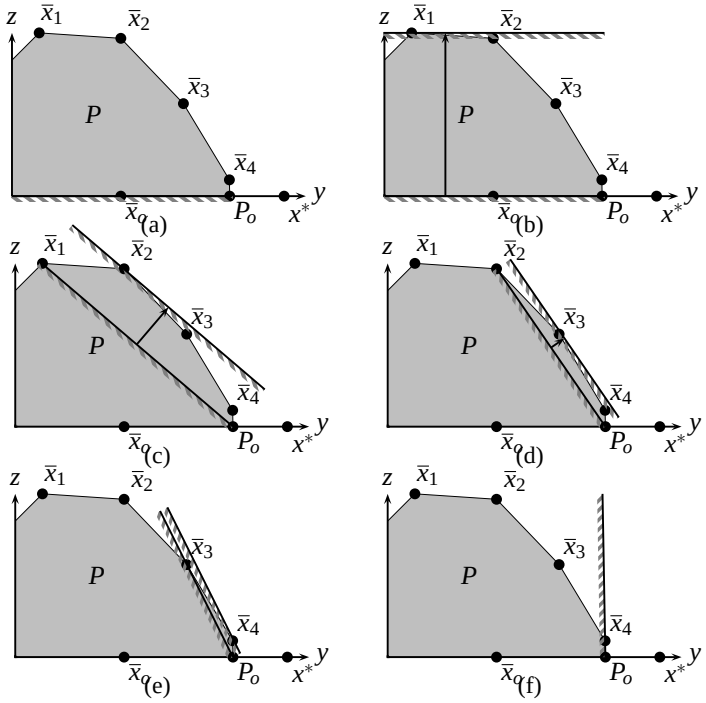

Figure 3: Sequence of push steps for P.

To continue our algorithm we will find a positive combination of $v^T x \leq w$ and of $a^T x \leq b$ such that every point in P_o still satisfies it at equality, but such that x' also satisfies it at equality. For this let $\lambda := v^T x' - w > 0$, $\mu := b - a^T x' \geq 0$, and define the inequality

$$(v', w') = \lambda(a, b) + \mu(v, w). \tag{11}$$

Since (11) is a positive combination of (a,b) and (v,w), every point x in P_o satisfies $v'^T x = w'$. Moreover, x' also satisfies (11) at equality. To see this, note that $v'^T x' - w' = \lambda(a^T x' - b) + \mu(v^T x' - w) = -\lambda\mu + \mu\lambda = 0$. Now we replace (v,w) by (v',w') and $\bar{x}$ by x', and we repeat the previous process.

Every step where we redefine our tentative (v,w) inequality is called a *push* step. Figure 3 shows a sequence of push steps that end with the desired inequality.

Algorithm 3 gives an outline of the tilting algorithm, and also defines the output for it. We now show that this algorithm satisfies Condition 1.

Lemma 1. *In every push step, there exists $\lambda_k \geq 0$ and $\mu_k \geq 0$ such that $(v_k, w_k) = \mu_k(v_o, w_o) + \lambda_k(a,b)$ and where $\mu_k + \lambda_k > 0$*

Algorithm 3 TILT$(a,b,v_o,w_o,\bar{x}_o,x_o,OPT)$

Require: $\bar{x}_o \in P$, $a^T\bar{x}_o < b$, $v_o^T\bar{x}_o = w_o, x_o \in P_o$, $a^T x_o = b$, $v_o^T x_o = w_o$.
$a^T x \leq b$ is a valid inequality for P, (a,b) and (v_o,w_o) are linearly independent.

$k \leftarrow 0$.
loop
  $(status,\beta,y) \leftarrow OPT(v_k)$
  if $status =$ `optimal` and $\beta = w$ **then**
    return $(v_k,w_k,\bar{x}_k)$
  if $status =$ `unbounded` **then**
    $\bar{x}_{k+1} \leftarrow y + x_o$
  else
    $\bar{x}_{k+1} \leftarrow y$
  $\lambda \leftarrow v_k^T\bar{x}_{k+1} - w_k,\ \mu \leftarrow b - a^T\bar{x}_{k+1}$
  $v_{k+1} \leftarrow \lambda a + \mu v_k,\ w_{k+1} \leftarrow \lambda b + \mu w_k$.
  $k \leftarrow k+1$.

Proof. We proceed by induction, the case $k = 0$ being trivially true with $\lambda_o = 0, \mu_o = 1$. We assume now that the result is true for k, and that the algorithm does not stop during this iteration (otherwise we have finished the proof). We have that $\bar{x}_{k+1}$ is such that $\lambda = v_k^T\bar{x}_{k+1} - w_k > 0$ and that $\mu = b - a^T\bar{x}_{k+1} \geq 0$. By definition, $(v_{k+1},w_{k+1}) = \lambda(a,b) + \mu(v_k,w_k)$, and by assumption $(v_k,w_k) = \lambda_k(a,b) + \mu_k(v_o,w_o)$. Now, by defining $\lambda_{k+1} = \lambda + \lambda_k\mu$ and $\mu_{k+1} = \mu\mu_k$, we obtain our result. □

Lemma 2. *At each step, $\bar{x}_k$ and all $\bar{x} \in P_o$ satisfy $v_k^T x = w_k$, and $\bar{x}_k$ is affinely independent from P_o.*

Proof. By construction. □

It remains to prove that the algorithm terminates after a finite number of steps. To this end, we need to make the assumption that for a given polyhedron P the oracle will return only one of a finite number of rays and points. This assumption is mild in our context, since any polyhedron with rational data can be represented as a finitely generated cone plus the convex hull of a finite set of feasible points. To verify that the algorithm terminates under this assumption we show that there is a function that strictly increases in every iteration.

Define $f(\lambda,\mu) : \mathbb{R}_+ \times \mathbb{R}_+ \to \mathbb{R}_+ \cup \{\infty\}$ as $f(\lambda,\mu) = \frac{\lambda}{\mu}$, where we set $\frac{t}{0} = \infty, \forall t \geq 0$.

Lemma 3. *For all integers $k \geq 0$, $f(\lambda_k,\mu_k) < f(\lambda_{k+1},\mu_{k+1})$.*

Proof. Note that $f(\lambda_o,\mu_o) = \frac{0}{1} = 0$, moreover, by the proof of Lemma 1 we have that $f(\lambda_{k+1},\mu_{k+1}) = \frac{\lambda+\lambda_k\mu}{\mu\mu_k} = \frac{\lambda}{\mu\mu_k} + \frac{\lambda_k}{\mu_k}$. Note that if $\mu_k = 0$ then $(v_k,w_k) = \lambda_k(a,b)$, thus ensuring (by hypothesis of the algorithm) that the algorithm will stop in iteration $k+1$. Then, we may assume that $\mu_k \neq 0$. By the previous equations, we have that $f(\lambda_{k+1},\mu_{k+1}) = \frac{\lambda}{\mu\mu_k} + f(\lambda_k,\mu_k)$. Since $\lambda > 0$ and $\mu \geq 0$, we have that $\frac{\lambda}{\mu\mu_k} > 0$. This proves our claim. □

Lemma 4. *At every iteration, $f(\lambda_k,\mu_k) = \frac{vx_k - w}{b - ax_k}$.*

Proof. Note that the statement is true for $k = 0$. For any other k we have $f(\lambda_{k+1},\mu_{k+1}) = \frac{v_k x_{k+1} - w_k + \mu\lambda_k}{\mu\mu_k}$ and $v_k x_{k+1} - w_k + \mu\lambda_k = \lambda_k a x_{k+1} + \mu_k v x_{k+1} - \lambda_k b - \mu_k w + (b - ax_{k+1})\lambda_k = \mu_k(vx_{k+1} - w)$. Since $\mu = b - ax_{k+1}$, we obtain the result. □

Theorem 5. *If the optimization oracle returns only a finite number of distinct feasible points and/or rays, then Algorithm 3 terminates in a finite number of steps.*

Proof. First, note that if in step 3 of Algorithm 3 the oracle returns $(\texttt{unbounded}, r_{k+1})$, then $x_{k+1} = r_{k+1} + x_o$, and since $ax_o = b, vx_o = w$ we obtain that $f(\lambda_{k+1}, \mu_{k+1}) = \frac{vr_{k+1}}{-ar_{k+1}}$. On the other hand, if the oracle returns $(\texttt{optimal}, x_{k+1})$ and $f(\lambda_{k+1}, \mu_{k+1}) = \frac{vx_{k+1}-w}{b-ax_{k+1}}$. This proves that f provides a total order for all possible outputs of the oracle. Since at every iteration we either stop or increase f, and the possible answers of the oracle are finite, the algorithm must terminate. □

3.2 More on the facet procedure

The inequality produced by our facet procedure can have arbitrarily small violation, as illustrated in Figure 4. However, the distance from x^* to the set of points satisfying both $a_+^T x \leq b_+$ and $a_-^T x \leq b_-$

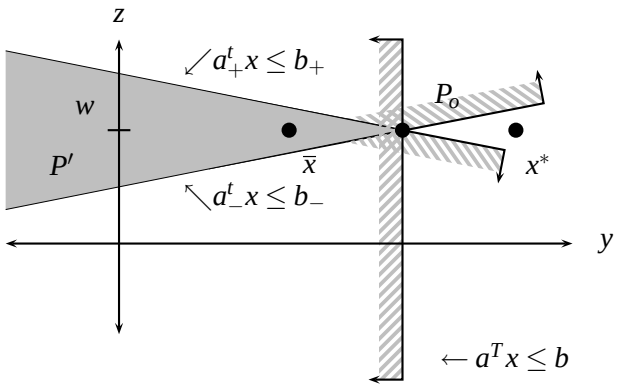

Figure 4: Rotated inequalities with small violation.

is at least the distance from x^* to the set of points satisfying our original constraint $a^T x \leq b$. We therefore choose to record both $a_+^T x \leq b_+$ and $a_-^T x \leq b_-$ in our implementation, but we proceed with the facet procedure only on the most violated of the two inequalities. Note that we might take this approach even further and iterate both inequalities through the facet procedure. The advantage of such a modification is that we find a set of facets of P that ensure the same violation as the original inequality, but the drawback is that it may require many more calls to the oracle.

High-dimensional faces. While the appeal of facets is clear, it may be that the computational cost of the overall procedure is too large. This suggests that we may want to stop the process after a certain number of steps. In this setting, an interesting question is how should we choose our non-trivial solution v, w to perform each tilting round?

Let us take a closer look into the facet certificate LP. Suppose we have a point $\bar{x}^* \in P$ in the affine space defined by the points in $P_o \cup \{\bar{x}\} = \{x_i : i = 1, \ldots, K\}$. Then, by definition, there exists $\lambda_i \in \mathbb{R}, i = 1, \ldots, K$ such that $\sum(\lambda_i : i = 1, \ldots, K) = 1$ and such that $\sum(\lambda_i x_i : i = 1, \ldots, K) = \bar{x}^*$. This

allows us to re-write the facet certificate LP as

$$w(y_o^T p_i) + v^T p_i = 0, \qquad \forall p_i \in P_o^\perp, \tag{12a}$$
$$w - v^T x_i = 0 \qquad \forall x_i \in P_o \cup \{\bar{x}, \} \tag{12b}$$
$$w - v^T \bar{x}^* = 0. \tag{12c}$$

Note that (12) is equivalent to our original formulation (10) except for the normalizing constraints in v, w. Moreover, by replacing w in equation (12b) we obtain

$$w(y_o^T p_i) + v^T p_i = 0, \qquad \forall p_i \in P_o^\perp, \tag{13a}$$
$$v^T (x_i - \bar{x}^*) = 0, \qquad \forall x_i \in P_o \cup \{\bar{x}\}, \tag{13b}$$
$$w - v^T \bar{x}^* = 0. \tag{13c}$$

Now by choosing $\bar{x}^*$ as the L_2-projection of x^* in the affine space generated by $P_o \cup \{\bar{x}\}$, we have that $(x^* - \bar{x}^*)^T x \leq (x^* - \bar{x}^*)^T \bar{x}^*$ is the best inequality separating x^* and the affine space generated by $P_o \cup \{\bar{x}\}$, in the sense of violation of the inequality and assuming that its norm is $\|x^* - \bar{x}^*\|_2$.

This suggests that we consider solving the problem

$$\max \quad v^T (x^* - \bar{x}^*) \tag{14a}$$
$$w(y_o^T p_i) + v^T p_i = 0, \qquad \forall p_i \in P_o^\perp, \tag{14b}$$
$$v^T (x_i - \bar{x}^*) = 0, \qquad \forall x_i \in P_o \cup \{\bar{x}\}, \tag{14c}$$
$$w - v^T \bar{x}^* = 0, \tag{14d}$$
$$\|v\|_2 \leq 1. \tag{14e}$$

Note that a solution to (14) gives us a closest approximation (under the L_2-norm) to the ideal solution $x^* - \bar{x}^*$ that also satisfies equation (14b).

However, if we are to take this approach, there remains the problem of computing $\bar{x}^*$. Fortunately, that is not needed at all. Indeed, by back-substituting (14d) and then eliminating the linearly dependent equation (14d), we end with the equivalent problem

$$\max \quad v^T x^* - w \tag{15a}$$
$$w(y_o^T p_i) + v^T p_i = 0, \qquad \forall p_i \in P_o^\perp, \tag{15b}$$
$$w - v^T x_i = 0, \qquad \forall x_i \in P_o \cup \{\bar{x}\}, \tag{15c}$$
$$\|v\|_2 \leq 1. \tag{15d}$$

A difficulty with (15) is that constraint (15d) is not linear, but note that if we replace it with an L_∞-normalization we should also obtain an approximation to $x^* - \bar{x}^*$. This is the approach that we take in our implementation. At every step of the facet procedure described in Algorithm 2, we choose a non-trivial solution (if one exists) of the problem

$$\max \quad v^T x^* - w \tag{16a}$$
$$w(y_o^T p_i) + v^T p_i = 0, \qquad \forall p_i \in P_o^\perp, \tag{16b}$$
$$w - v^T x_i = 0, \qquad \forall x_i \in P_o \cup \{\bar{x}\}, \tag{16c}$$
$$-1 \leq v_i \leq 1. \tag{16d}$$

This technique can be viewed as a greedy approach for choosing the new tentative $v^T x \leq w$ inequality. It would be interesting to find procedures to choose both $\bar{x}$ and (v,w) such that the final facet satisfies certain properties, for example, that its area is large.

4 The mapping problem

Rather than directly computing inequalities for P_{IP}, the local-cuts procedure works on related sets designed to be easier to handle via optimization methods. Consider a function $\pi : \mathbb{R}^n \to \mathbb{R}^{n'}$ and a set $P'_{IP} \subseteq \mathbb{R}^{n'}$ such that for all points $x \in P_{IP}$ we have $\pi(x) \in P'_{IP}$. We call π a *mapping function* and we call P'_{IP} a *valid mapping* for P_{IP}. We denote by $\overline{P}_{IP}$ and $\overline{P}'_{IP}$ the convex hull of P_{IP} and P'_{IP}, respectively.

If π is a linear (affine) function, then any valid inequality $a^T y \leq b$ for $\overline{P}'_{IP}$ yields a potential cutting plane for the original problem, namely $a^T \pi(x) \leq b$. Indeed, if we write $\pi(x) = Mx - m_o$, then the cut in the original space can be written as $a^T Mx \leq b + a^T m_o$. Note also that any linear mapping function satisfies $\pi(\overline{P}_{IP}) \subset \overline{P}'_{IP}$. We call π a *separating* mapping function if $\pi(x^*) \notin P'_{IP}$.

Many of the well-known techniques for producing general MIP cutting planes can be framed as methods for obtaining valid inequalities in suitably mapped spaces, including GMI cuts, MIR inequalities, and Fenchel cuts. The local-cuts paradigm provides a method for extending and combining these procedures, giving the possibility of strengthening the LP relaxations, at the expense of increased computation in the construction of the cuts.

A general discussion of separating mappings is given in the Ph.D. thesis of Espinoza [19]. Here we illustrate the procedure by considering a simple class called *implicit mappings*.

Implicit mappings. A natural way to obtain a mapping is to define P'_{IP} as the set of points y that are a linear combination of points x satisfying a set of linear combinations of the original constraints. More precisely, we say that a mapping is *implicit* if

$$P'_{IP} = \left\{ y \in \mathbb{R}^{n'} : \exists x \in \mathbb{R}^n,\ l \leq x \leq u,\ RAx = Rd, y = Mx, y_i \in \mathbb{Z}, \forall i \in I' \right\}$$

where R and M are matrices and $I' \subseteq \{1, \ldots, n'\}$. In this case the mapping function is just $\pi(x) = Mx$. By considering the system $RAx = Rd$ obtained by taking linear combinations of $Ax = d$, the class of implicit mappings includes those relaxations created by selecting rows of a simplex tableau for $\max(c^T x : x \in P_{LP})$, such as the GMI cuts. In the case of GMI cuts, the mapping P'_{IP} has a single integer variable, obtained by aggregating the integer variables appearing in the tableau row.

A sufficient condition for an implicit mapping to be valid is to have $M_{ij} = 0, \forall i \in I', j \notin I$, and $M_{ij} \in \mathbb{Z}, \forall i \in I', j \in I$. That is, all integer variables in P'_{IP} must be integer combinations of integer variables in P_{IP}. A necessary condition for an implicit mapping to be separating is that there exists some objective function $\mu \neq 0$ such that $\max(\mu^T y : y \in P'_{LP}) = \mu^T Mx^*$, where x^* is the point that we are separating. If we also have that $\pi(x^*)$ is the unique optimal solution to $\max(\mu^T y : y \in P'_{LP})$, and some integer-constrained variable $\pi(x^*)_i = y^*_i, i \in I'$, is fractional, then the mapping is separating.

Simple local cuts. The well-known result of Lenstra [25] provides a polynomial-time algorithm for solving MIP instances having a fixed number of integer variables. This suggests the following implicit mapping. Set M and R as identity matrices, and define $I' \subseteq I$ such that $|I'| \leq k$ for some

fixed k. We show in a computational study that this simple process can produce effective cuts, with k as small as four.

5 The local-cuts procedure

We are now in position to describe the full scheme for MIP local cuts. We start with a given MIP problem P_{IP}, a linear relaxation P_{LP}, and a fractional basic optimal solution x^* to P_{LP}. We choose some linear mapping π and an image space P'_{IP}, for which we provide an oracle description $\mathscr{O}_{P'_{IP}}$. We then call Algorithm 1 as $\texttt{SEP}(\mathscr{O}_{P'_{IP}}, \pi(x^*), \emptyset, \emptyset)$. If the algorithm finds a separating inequality, we proceed to call Algorithm 2 using as input the separating inequality found by Algorithm 1, and using as P_o the set of points satisfying the separating inequality at equality. We then take the resulting inequalities and map them back to the original space and add them to our current relaxation.

In our implementation we try six mappings before re-solving the LP relaxation. Also, instead of asking for facets of the mapped problem, we ask only for faces with dimension at least ten, but keeping all intermediate constraints produced by algorithm `FACET`.

Precision. Any error in the solution of the separation LP, or in the solutions provided by the oracle, may lead to invalid inequalities. Applegate et al. [5] deal with this issue by using integer representations of the inequalities and of solutions, and by using an oracle whose solutions are always integer. We follow their strategy, using rational representations and the exact rational LP solver described in [6].

A point to note when working with rational inequalities is that their representation may grow as we go along in the algorithm. From our experiments we know that even LP instances with simple coefficients can generate solutions that need a long encoding to be represented, and it seems to be the case that as we allow more and more complicated inequalities, the situation only worsens. To settle this point, we choose to accept only cuts such that the denominator and numerator of each coefficient can be expressed using at most some fixed number of bits. In our tests we set this limit at 3200 bits.

Improving oracles. In the tilting procedure described in Algorithm 3, at each iteration we have a candidate inequality and we ask whether or not it is valid for the polyhedron. If it is not valid, then it is sufficient to find a point in P that violates the candidate inequality. In particular, an optimal LP solution is not required. We have seen in practice that working with such an *improving oracle* speeds-up the overall performance of the local-cut procedure.

5.1 Computational results

The goal of our computational study is to demonstrate the feasibility of carrying out the local-cuts process in general MIP instances, and to evaluate if effective cutting planes can be obtained via the simple mappings described in Section 4.

Recall that simple mappings select a small number k of integer variables and relax the integrality conditions for all other x_i, $i \in I$. To carry out the local-cuts process, we need to create an optimization oracle for these relaxations. For this purpose, we have implemented an exact rational branch-and-cut MIP code on top of the exact LP solver. The implementation includes a form of

pseudo-cost branching as discussed in [26, 2, 1], K-cuts as explained in [17], GMI cuts derived from the aggregation of two tableau rows, and super additive lifted cover inequalities as in both [23, 7].

To test the effectiveness of the local cuts, we first computed default LP relaxations by repeatedly applying the cutting-plane routines listed above. A great advantage of working in rational arithmetic is that we can run these routines until no violated cuts are produced, rather than terminating them after several rounds in an effort to limit numerical difficulties. Each round of cutting works by sequentially looking for lifted cover inequalities, then K-cuts, and then Gomory cuts derived from the aggregation of two tableau rows. If one of the procedures is successful, then we add up to 500 of the cuts, re-solve the resulting LP, and start the process anew.

When testing local cuts, we also use our default cutting-plane-generation methods, that is, the local cuts are generated after default cuts have failed. If acceptable local cuts are found, our default routines are first tried again during the next round of cutting, until none of the cutting-plane schemes can produced acceptable cuts.

To select the k variables in the simple mappings, we order the integer variables x_i according to the distance of x_i^* to the nearest integer value, where x^* is the current optimal LP solution. The first mapping uses the k most fractional variables, and each subsequent mapping is obtained by randomly choosing one of the k variables and replacing it by a random choice among the remaining fractional variables.

Test bed. Since our procedure is fully rational, there are no rounding errors or problems caused by cutting off feasible solutions. This advantage comes at a price: our computations can be very time consuming. For this this reason we use as our test bed a set of smaller instances obtained from the MIPLIB 3.0 [10], MIBLIB 2003 [3], and Mittleman [28] collections. From this full set, we removed any instance that could not be solved in under five minutes with CPLEX 10.0 [24]. Furthermore, to restrict to instances where cutting is needed to obtain a good relaxation, we also eliminated the instances having an integrality gap of less than 1%, that is, the difference between the MIP optimal value and the optimal value of the original LP relaxation is less than 1% of the MIP value. The remaining 38 instances are listed in Table 1.

aflow30a	air05	bc1	blend2	fiber	fixnet6
flugpl	gt2	l152lav	lseu	mas76	misc03
misc07	mod008	neos10	neos1	neos20	neos21
neos7	nug08	p0033	p0201	p0282	pk1
pp08aCUTS	pp08a	qiu	qnet1	qnet1_o	rentacar
rgn	rout	set1ch	stein27	stein45	swath1
swath2	vpm2				

Table 1: Test set instances

Improved LP bounds. We tested the following three configurations.

- LO: Our default set of cutting planes.
- LS_4: Local cuts from simple mappings with $k = 4$ integer variables.
- LS_6: Local cuts from simple mappings with $k = 6$ integer variables.

For each configuration and all 38 test instances, we computed LP relaxations (no branching allowed) using up to seven days computing time for each instance and recording the final LP bound. If the configuration did run to termination, then we recorded the LP value found at the end of the seven-day period. The tests were carried on computes nodes equipped with a 2.33 GHz Intel Xeon E5345 processor and 8GB of random-access memory.

To compare the results, for each instance and each configuration, we computed the ratio of the LP bound obtained versus the best of the LP bounds obtained by all three configurations. The full set of ratios is plotted in the performance profile presented in Figure 5. In this plot, the horizontal

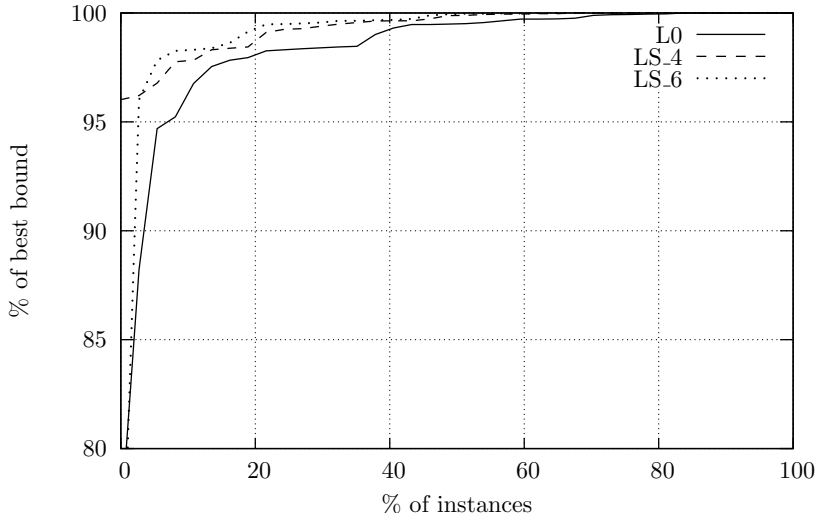

Figure 5: Performance profile for percentage of best LP values obtained

axis gives the percentage of test instances and the vertical axis gives the LP-bound ratios scaled by 100. For each of the three configurations, a point (x,y) in the plot means that on all but x% instances the LP bound for this configuration was at least y% of the best of the three LP bounds. Thus a higher curve gives a better performing configuration.

It can be seen from the plot that simple local cuts with $k = 4$ attained a reasonable overall improvement over the default cutting routine, while the runs with $k = 6$ were superior on a number of instances. On average, L0 attained 97.6% of the best lower bound, LS_4 attained 99.2%, and LS_6 attained 97.8%. The lower value for LS_6 compared with LS_4 is due to the fact that many more of the LS_6 runs did not complete in the allowed time period. If we restrict our results to the 12 instances where all three configurations ran to completion, then the default code attained 96.6% of the best lower bound, LS_4 attained 99.0%, and LS_6 attained 99.9%.

The improved LP bounds obtained by simple local cuts come at increased computational costs and we cannot make a claim that the method is one that can be adopted in the practical solution of large MIP instances. The positive results do however suggest that the overall local-cuts process is one that could be considered in settings where the best-possible LP relaxation is required, for example, in a preliminary step to a large-scale parallel branch-and-cut search, where individual processors can be devoted to evaluating potential mappings.

5.2 Implementation Issues

In several instances we were unable to add cuts to our LP relaxation because the encoding of the inequalities exceeded our bit-length limits. This raises the question of whether the problem of long encodings is common and unavoidable, or if there are techniques (or special relaxations) that naturally yield inequalities with short descriptions. The question is not just a rhetorical one, since in the case of the TSP the local-cuts procedure usually finds inequalities with short descriptions. An interesting research topic is to find mappings that are both good in this numerical sense and achieve strong cuts.

References

[1] T. ACHTERBERG, *SCIP: solving constraint integer programs*, Mathematical Programming Computation, 1 (2009), pp. 1–41.

[2] T. ACHTERBERG, T. KOCH, AND A. MARTIN, *Branching rules revisited*, Operations Research Letters, 33 (2005), pp. 42–54.

[3] T. ACHTERBERG, T. KOCH, AND A. MARTIN, *MIPLIB 2003*, Operations Research Letters, 34 (2006), pp. 1–12.

[4] E. ALTHAUS, T. POLZIN, AND S. V. DANESHMAND, *Improving linear programming approaches for the Steiner tree problem*, in Experimental and Efficient Algorithms, Second International Workshop, WEA 2003, K. Jansen, M. Margraf, M. Mastrolilli, and J. D. P. Rolim, eds., Springer, 2003, pp. 1–14.

[5] D. APPLEGATE, R. E. BIXBY, V. CHVÁTAL, AND W. COOK, *The Traveling Salesman Problem: A Computational Study*, Princeton University Press, Princeton, New Jersey, 2006.

[6] D. APPLEGATE, W. COOK, S. DASH, AND D. ESPINOZA, *Exact solutions to linear programming problems*, Operations Research Letters, 35 (2007), pp. 693–699.

[7] A. ATAMTÜRK, *Sequence independent lifting for mixed–integer programming*, Operations Research, 52 (2004), pp. 487–490.

[8] E. BALAS, S. CERIA, AND G. CORNUÉJOLS, *A lift-and-project cutting plane algorithm for mixed 0-1 programs*, Mathematical Programming, 58 (1993), pp. 295–324.

[9] E. BALAS, S. CERIA, G. CORNUÉJOLS, AND N. NATRAJ, *Gomory cuts revisited*, Operations Research Letters, 19 (1996), pp. 1–9.

[10] R. E. BIXBY, E. A. BOYD, AND R. R. INDOVINA, *MIPLIB: A test set of mixed integer programming problems*, SIAM News, 25 (1992), p. 16.

[11] R. E. BIXBY, M. FENELON, Z. GU, E. ROTHBERG, AND R. WUNDERLING, *Mixed-integer programming: A progress report*, in The Sharpest Cut: The Impact of Manfred Padberg and His Work, M. Grötschel, ed., SIAM, Philadelphia, 2004, pp. 309–325.

[12] E. A. BOYD, *Generating Fenchel cutting planes for knapsack polyhedra*, SIAM Journal on Optimization, 3 (1993), pp. 734–750.

[13] ——, *Fenchel cutting planes for integer programs*, Operations Research, 42 (1994), pp. 53–64.

[14] C. BUCHHEIM, F. LIERS, AND M. OSWALD, *Local cuts revisited*, Operations Research Letters, 36 (2008), pp. 430–433.

[15] ——, *Speeding up IP-based algorithms for constrained quadratic 0–1 optimization*, Tech. Rep. zaik2008-578, Zentrum für Angewandte Informatik Köln, Germany, 2008.

[16] A. CAPRARA AND M. FISCHETTI, *0,1/2-Chvátal-Gomory cuts*, Mathematical Programming, 74 (1996), pp. 221–235.

[17] G. CORNUÉJOLS, Y. LI, AND D. VANDENBUSSCHE, *K-cuts: A variation of Gomory mixed integer cuts from the LP tableau*, INFORMS J. on Computing, 15 (2003), pp. 385–396.

[18] H. P. CROWDER, E. L. JOHNSON, AND M. W. PADBERG, *Solving large-scale zero-one linear programming problems*, Operations Research, 31 (1983), pp. 803–834.

[19] D. G. ESPINOZA, *On Linear Programming, Integer Programming and Cutting Planes*, PhD thesis, School of Industrial and Systems Engineering, Georgia Institute of Technology, March 2006.

[20] R. E. GOMORY, *An algorithm for the mixed integer problem*, Tech. Rep. RM-2597, RAND Corporation, 1960.

[21] M. GRÖTSCHEL, L. LOVÁSZ, AND A. SCHRIJVER, *Geometric Algorithms and Combinatorial Optimization, 2nd Edition*, Springer, Berlin, Germany, 1993.

[22] Z. GU, G. L. NEMHAUSER, AND M. W. P. SAVELSBERGH, *Lifted cover inequalities for 0-1 integer programs*, Mathematical Programming, 85 (1999), pp. 437–467.

[23] ——, *Sequence independent lifting in mixed integer programming*, Journal of Combinatorial Optimization, 4 (2000), pp. 109–129.

[24] ILOG, *User's Manual, ILOG CPLEX 10.0*, ILOG CPLEX Division, Incline Village, Nevada, 2006.

[25] H. W. LENSTRA JR., *Integer programming with a fixed number of variables*, Mathematics of Operations Research, 8 (1983), pp. 538–548.

[26] J. T. LINDEROTH AND M. W. P. SAVELSBERGH, *A computational study of search strategies for mixed integer programming*, INFORMS J. on Computing, 11 (1999), pp. 173–187.

[27] H. MARCHAND AND L. A. WOLSEY, *Aggregation and mixed integer rounding to solve MIPs*, Operations Research, 49 (2003), pp. 363–371.

[28] H. MITTELMANN, *Mixed integer linear programming benchmark (free codes)*, //http://plato.asu.edu/ftp/milpf.html, (2008).