Multi-Row Intersection Cuts based on the Infinity Norm

Álinson S. Xavier*

Energy Systems Division
Argonne National Laboratory, USA

Ricardo Fukasawa† Laurent Poirrier‡

Department of Combinatorics and Optimization
University of Waterloo, Canada

July 16, 2019

Abstract

When generating multi-row intersection cuts for Mixed-Integer Linear Optimization problems, an important practical question is deciding which intersection cuts to use. Even when restricted to cuts that are facet-defining for the corner relaxation, the number of potential candidates is still very large, specially for instances of large size. In this paper, we introduce a subset of intersection cuts based on the infinity norm that is very small, works for relaxations having arbitrary number of rows, and, unlike many subclasses studied in the literature, that takes into account the entire data from the simplex tableau. We describe an algorithm for generating these inequalities and run extensive computational experiments in order to evaluate their practical effectiveness in real-world instances. We conclude that this subset of inequalities yields, in terms of gap closure, around 50% of the benefits of using all valid inequalities for the corner relaxation simultaneously, but at a small fraction of the computational cost, and with a very small number of cuts.

Keywords Mixed-Integer Linear Programming (MIP) · Intersection Cuts · Cutting Planes

1 Introduction

Cutting planes are one of the main techniques currently used to solve large-scale Mixed-Integer Linear Optimization problems (MIPs). One strategy for obtaining general-purpose cuts is to

*axavier@anl.gov
†rfukasawa@uwaterloo.ca
‡lpoirrier@uwaterloo.ca

consider inequalities that are valid for the *continuous corner relaxation* of a simplex tableau, given by

$$\begin{aligned} x &= f + \sum_{j=1}^{m} r^j s_j \\ x &\in \mathbb{Z}^n \\ s &\in \mathbb{R}^m_+, \end{aligned} \tag{C}$$

where $f \in \mathbb{Q}^n \setminus \{0\}$ and $r^1, \ldots, r^m \in \mathbb{Q}^n \setminus \{0\}$. For simplicity, we assume throughout this work that $\text{cone}(r^1, \ldots, r^m) = \mathbb{R}^n$. In this relaxation, x and s correspond, respectively, to the basic and non-basic variables, while f corresponds to the right-hand side of the tableau. The corner relaxation was first introduced by Gomory [9] in the late 1960s and, following Andersen, Louveaux, Weismantel and Wolsey [1], has received considerable attention in more recent years (see [5] for a survey). It is well known that valid inequalities for (C) can be constructed from *lattice-free sets* — convex sets that do not contain any integer points in their interior. More precisely, if $B \subseteq \mathbb{R}^n$ is a lattice-free set containing f in its interior, and if ψ is the gauge function of $B - f$, then the inequality

$$\sum_{j=1}^{m} \psi(r^j) s_j \geq 1, \tag{1}$$

is valid for (C). This approach of producing cutting planes for the continuous corner relaxation through lattice-free sets was first introduced by Balas [2], and cuts obtained in such way are usually called *intersection cuts*. When generating cutting planes for MIPs, we are typically interested in inequalities from (C) that cut off the fractional basic solution associated with the tableau, given by $(x, s) = (f, 0)$. It can be easily verified, however, that any intersection cut generated as described above satisfies this requirement. Even when restricted to intersection cuts that induce facets of (C), the number of suitable cuts is still very large, specially for instances of large size. An important question, therefore, is to decide which intersection cuts from (C) to use in a practical cutting-plane algorithm. Various factors must be considered, including not only the speed in which these cuts can be generated, but also the total number of cuts ultimately selected to be added to the linear relaxation of the MIP. In the following, we present some cut selection strategies that have been proposed in the literature, as well as their impact on solving real-world MIPs.

In the first extensive computational experiments on the impact of using multi-row intersection cuts, Espinoza [8] considers intersection cuts generated from three simple families of bounded, maximal lattice-free polyhedra. The lattice-free sets considered have fixed shape, and, in particular, do not change according to the values r^j. The impact of each family of cuts, for a different number of tableau rows, was measured individually on instances from MIPLIB 3.0 and MIPLIB 2003. The cuts improved both the LP bound obtained at the root node of the branch-and-bound tree, as well as the total running time of the algorithm. The best

configurations, compared against CPLEX 11.0 defaults, increased the gap closed at the root node by 2.5 percentage points and reduced the overall running time by 5%. Interestingly, the family of intersection cuts that presented the best results was never facet-defining for (C). The paper concludes that even simple subclasses of intersection cuts from (C) can have positive impact on the performance of branch-and-cut algorithms, and points to identifying additional important subclasses of such cuts, along with good computational implementation choices, as an interesting research question.

Musalem [13] focuses on generating multi-row intersection cuts $\pi^T s \geq 1$ that minimize the 1-norm and the 2-norm of the vector π. The problem of finding these cuts is modeled as a MIP and solved using column generation. Computational experiments are conducted on instances from MIPLIB 3.0, and the results are compared against the split cut separator implemented by Balas and Saxena [3]. Results are negative, as the 1-norm cut separator was only able to close 34% of the integrality gap at the root node, on average, under a two-hour time limit, while the split cut separator is able to close 86% of the gap, under a 10-hour time limit. Yamangil [15] also performs computational experiments on cuts minimizing the 1-norm, for relaxations with 2 to 5 rows. Compared against CPLEX defaults, these cuts closed, on average, 60% of the integrality gap at the root node, while CPLEX closed 52%.

Dey, Lodi, Tramontani and Wolsey [6, 7] focus on the specific case where (C) is a two-row relaxation. They evaluate the impact of cuts generated from maximal lattice-free triangles and from two-row splits, compared against, and in conjunction with, Gomory Mixed-Integer (GMI) cuts. Because the number of suitable triangles, even when restricted to facet-defining ones, is very large, they develop a heuristic procedure to select a small, but effective subset of triangles. Experiments are conducted both on a set of randomly generated instances and on a subset of MIPLIB 3.0 instances. While both families of two-row intersection cuts, used together, proved useful for all sets of instances, most of the improvement for real-world instances came from two-row split cuts.

Basu, Bonami, Cornuéjols and Margot [4] also focus on the two-row case. They generate cuts from relaxations where one component of f is integral and the other is fractional. This typically occurs when (C) comes from a tableau whose basis is degenerate. They measure the impact of a restricted subset of lattice-free triangles, carefully chosen so that the separation and trivial lifting procedure can be done through simple closed formulas. To measure the strength of these cuts, they use the method *diving towards a feasible solution*, described by Margot [12]. The conclusion is that the family of two-row cuts tested is not competitive with GMI cuts.

Louveaux and Poirrier [10] also focus on the two-row case, but instead of considering lattice-free sets with specific shapes, they develop an exact separator that, given a point $(\bar{x}, \bar{s})$, either finds an intersection cut from (C) that cuts off this point, or proves that no such intersection cut exists. In this way, they can evaluate the impact of using all facet-defining intersection cuts for (C) simultaneously, without explicitly enumerating them. By using techniques to reduce the complexity of the polar set of (C), they are able to obtain a method that works well in practice,

despite having no guarantee of polynomial running time. They conclude that the gap closed by two-row cuts, on top of single-row cuts, is considerable, although much of that closure can be achieved from split cuts. In later experiments, Louveaux, Poirrier and Salvagnin [11] develop another exact separator that, although much slower than the former, can separate over several variations of (C), and can handle relaxations with arbitrary numbers of rows.

Following the direction proposed by Espinoza [8], our goal in this paper is to identify additional subclasses of intersection cuts generated from (C) that can be easily computed in practice and that, when used as cutting planes for solving MIPs, provide benefits comparable to using all the facet-defining inequalities for (C). While many strategies have been proposed for two-row relaxations and triangles, other subclasses of intersection cuts, specially for relaxations with more than two rows, have received much less attention.

In this paper, we introduce a new subset of intersection cuts from (C), based on the infinity norm, which has many interesting properties. First, it is very small, with exactly one cut per continuous multi-row relaxation. Its cuts are minimal, which implies that they are individually undominated. Unlike the classes introduced by [6, 7] and [4], it works for relaxations with an arbitrary number of rows, and unlike the subclasses introduced by [8], it takes into account the coefficients r^j of the variables s_j. Finally, these cuts have an interesting geometrical interpretation, which we exploit in order to compute them more efficiently.

We start in Section 2, by giving a precise definition of these cuts and proving some of their basic properties. In Section 3 we describe an algorithm for generating them. Finally, in Section 4, we run extensive computational experiments to evaluate their strength. We conclude that these cuts yield on average, in terms of gap closure, about 50% of the benefits of using an exact separator, but at a small fraction of the computational cost, and with a significantly smaller number of cuts.

2 Definition of infinity cuts

In this paper, we assume that relaxation (C) has at least one feasible solution, in which case it can be proved that every valid inequality that cuts off the point $(f, 0)$ can be written as

$$\sum_{i=1}^{m} \pi_i s_i \geq 1,$$

where $\pi_1, \ldots, \pi_m \geq 0$. Since the variables s_i are non-negative, we are generally interested in cuts that have small π_i coefficients. A natural idea, therefore, is to consider cuts that minimize $\|\pi\|$ for different norms. Musalem [13] and Yamangil [15] performed computational experiments with cuts minimizing the 1-norm and the 2-norm. Here, we study intersection cuts that minimize the infinity norm.

Given a vector $\pi \in \mathbb{R}^m$, we recall that the infinity norm of π is given by

$$\|\pi\|_\infty = \max\{|\pi_1|, \dots, |\pi_m|\}.$$

Since this norm only takes into account the coefficient with largest magnitude, cuts that minimize it, in a sense, try to assign, with equal priority, good coefficients to all the variables. Next, we describe a simple geometric way of obtaining a cut that minimizes the infinity norm. Consider the inequality

$$\sum_{i=1}^{m} \varepsilon s_i \geq 1, \tag{2}$$

where $\varepsilon > 0$.

Using the framework of intersection cuts, we can prove that inequality (2) is valid for (C) by verifying that a certain convex set is lattice-free, as the next lemma shows.

Lemma 1. *For every $\varepsilon > 0$, the convex set*

$$B(\varepsilon) = \text{conv}\left(\{f\} \cup \left\{f + \frac{1}{\varepsilon} r^i : i \in \{1, \dots, m\}\right\}\right).$$

is a lattice-free set if and only if (2) is valid for (C).

Proof. Let ψ be the gauge function of $B(\varepsilon) - f$. We recall that, by definition,

$$\psi(r) = \inf\left\{\varepsilon > 0 : \frac{r}{\varepsilon} + f \in B(\varepsilon)\right\} \qquad \forall r \in \mathbb{Q}^n.$$

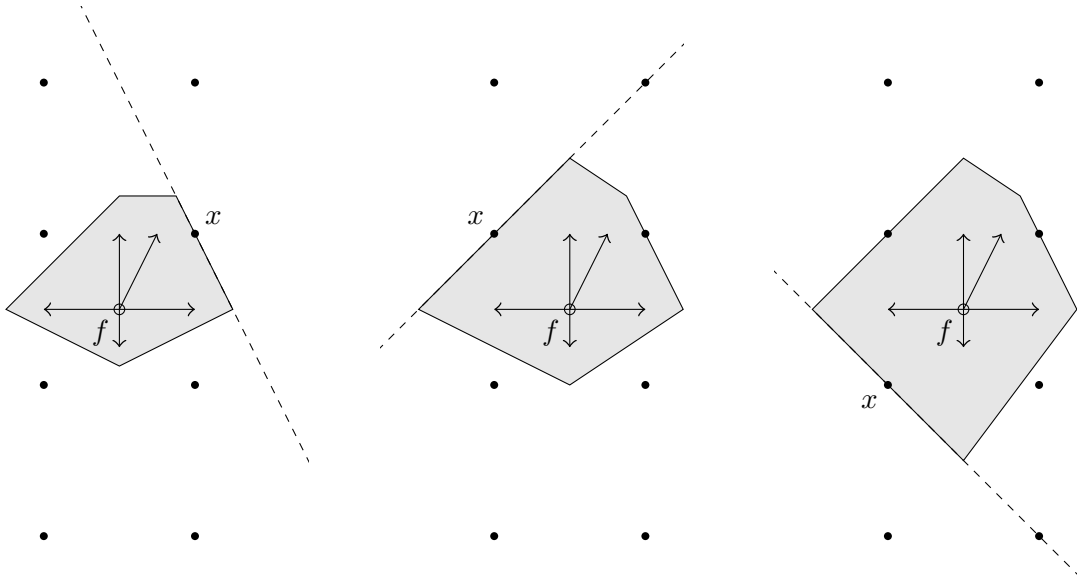
For every $i \in \{1, \dots, m\}$, since $f + \frac{1}{\varepsilon} r^i \in B(\varepsilon)$, then $\psi(r^i) \leq \varepsilon$. From the theory of intersection cuts, we know that $\sum_{i=1}^m \psi(r^i) s_i \geq 1$ is a valid inequality for (C). Since this inequality is at least as strong as (2), we conclude that (2) is also a valid inequality for (C). The converse follows from similar arguments. □

Indeed, if we pick the smallest ε such that (2) is valid, then (C) minimizes the infinity norm.

Proposition 2. *Let ε^* be the smallest ε such that (2) is valid for (C). Then $\sum_{i=1}^m \varepsilon^* s_i \geq 1$ is an intersection cut for (C) with smallest infinity norm.*

Proof. Let $\sum_{i=1}^m \pi_i s_i \geq 1$ be an intersection cut with smallest infinity norm. Let $\varepsilon := ||\pi||_\infty = \max\{\pi_1, \dots, \pi_m\}$. Then note that $\sum_{i=1}^m \varepsilon s_i \geq 1$ is also valid for (C), thus $\varepsilon^* \leq \varepsilon$. But also, since the infinity norm of the coefficients of $\sum_{i=1}^m \varepsilon^* s_i \geq 1$ is ε^*, we get $\varepsilon^* \geq \varepsilon$. □

From the Lemma 1 and from the fact that $f \notin \mathbb{Z}^n$, it is clear that, for a large enough ε, the set $B(\varepsilon)$ is lattice-free, and therefore (2) is valid. One strategy for obtaining a cut that minimizes the infinity norm, therefore, is to start with a very large value of ε and slowly decrease it, while making sure that the set $B(\varepsilon)$ is lattice-free. We stop when an integer point touches the boundary of $B(\varepsilon)$ since, for any larger ε, $B(\varepsilon)$ is not lattice-free. Although a cut that



(a) Level 1 (b) Level 2 (c) Level 3

Figure 1: Example of infinity cut for a two-row relaxation with 5 rays.

minimizes the infinity norm can be easily obtained in this way, we note that such cut is not necessarily minimal, and, in most cases, can be further strengthened. This is illustrated in the next example.

Example 3. *Consider the continuous relaxation*

$$\begin{aligned} & x = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix} s_1 + \begin{pmatrix} \frac{1}{4} \\ \frac{1}{2} \end{pmatrix} s_2 + \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix} s_3 + \begin{pmatrix} 0 \\ -\frac{1}{4} \end{pmatrix} s_4 + \begin{pmatrix} -\frac{1}{2} \\ 0 \end{pmatrix} s_5 \\ & x \in \mathbb{Z}^2 \\ & s \in \mathbb{R}^5_+ \end{aligned} \tag{3}$$

The set $B\left(\frac{2}{3}\right)$ *is illustrated in Figure 1a. Since this set is lattice-free and the point* $x = (1, 1)$ *belongs to its boundary, we conclude that*

$$\frac{2}{3}s_1 + \frac{2}{3}s_2 + \frac{2}{3}s_3 + \frac{2}{3}s_4 + \frac{2}{3}s_5 \geq 1$$

is an intersection cut that minimizes the infinity norm. Note, however, that this cut is not minimal. Indeed, it is dominated by the cut

$$\frac{1}{2}s_1 + \frac{2}{3}s_2 + \frac{2}{3}s_3 + \frac{1}{4}s_4 + \frac{1}{2}s_5 \geq 1,$$

which can be obtained from the lattice-free set illustrated in Figure 1c.

As the previous example illustrates, although the cuts obtained using the previous strategy

minimize the infinity norm, they are not necessarily minimal. In the following, we present a more restricted subset of cuts that are both minimal and that minimize the infinity norm. We start with the definition of a *tight* ray. Intuitively, a ray r is tight in a cut if its cut coefficient cannot be decreased without generating an invalid inequality.

Definition 4. *Given an inequality $\sum_{i=1}^{m} \pi_i s_i \geq 1$ that is valid for (C) and a ray r^k, where $k \in \{1, \ldots, m\}$, we say that r^k is* tight *with respect to this inequality if, for any $\bar{\pi}_k < \pi_k$, the inequality*

$$\sum_{\substack{i=1 \\ i \neq k}}^{m} \pi_i s_i + \bar{\pi}_k s_k \geq 1$$

is not valid for (C).

Next, we present the precise definition of the subset of cuts we will study in this paper. The definition is recursive, and leads naturally to an algorithm for obtaining these cuts. The idea is to decrease the cut coefficients of all the variables simultaneously, until some rays become tight and their coefficients cannot be decreased any further. The coefficients for these tight rays are then fixed, and the remaining coefficients are then lowered simultaneously. The process repeats, until all the rays become tight.

Definition 5.

(i) The inequality

$$\sum_{i=1}^{m} \varepsilon s_i \geq 1 \tag{4}$$

is an infinity cut (level 1) if $\varepsilon \geq 0$ is the smallest number such that (4) is valid.

(ii) Suppose

$$\sum_{i=1}^{m} \pi_i s_i \geq 1 \tag{5}$$

is an infinity cut (level k), for some $k \in \{1, 2, \ldots\}$ and some $\pi \in \mathbb{R}^m_+$. Let $T \subseteq \{1, \ldots, m\}$ be the set of indices i such that r^i is tight with respect to (5), and let $N = \{1, \ldots, m\} \setminus T$ be the set of the remaining indices. The inequality

$$\sum_{i \in T} \pi_i s_i + \sum_{i \in N} \varepsilon s_i \geq 1 \tag{6}$$

is an infinity cut (level $k+1$) if $\varepsilon \geq 0$ is the smallest number such that (6) is valid.

Figure 1 shows the infinity cuts (levels 1, 2 and 3) for the relaxation in Example 3. The next proposition shows that the infinity cut (level m) is indeed minimal, in the sense that no coefficient can be made individually stronger without generating an invalid cut.

Proposition 6. *If $\sum_{i=1}^{m} \pi_i s_i \geq 1$ is an infinity cut (level m), then this inequality is minimal.*

Proof. It can be easily proved by induction that at least k rays are tight with respect to the infinity cut (level k), for any $k \in \{1, \dots, m\}$. It follows that all the rays are tight with respect to the infinity cut (level m), and therefore the cut is minimal. □

We end this section with the observation that, from Definition 5, it is also clear that the infinity cut (level m) is unique for each continuous corner relaxation. This subset of intersection cuts, therefore, is very small, which makes it attractive computationally.

3 Computation of infinity cuts

In this section, we present an algorithm for computing the infinity cut (level m) for a given finite continuous multi-row relaxation. As in the previous section, we take a geometric approach, and focus on scaling up a certain lattice-free set.

First observe that, if (6) is an infinity cut (level k), then it can also be written as

$$\sum_{i=1}^{m} \max\{\beta_i, \varepsilon\} s_i \geq 1, \tag{7}$$

for some $\beta_1, \dots, \beta_m \geq 0$. Indeed, one valid choice of β is $\beta_i = \pi_i$ for every $i \in T$, and $\beta_i = 0$ for every $i \in N$. This form is more convenient than (6) since we do not need to deal with the sets T and N; all we need to characterize the cut are the values β_i. Similarly to the previous section, we can prove that (7) is a valid inequality by verifying that a certain convex set is lattice-free, as shown by the next lemma.

Lemma 7. *For every $\varepsilon > 0$ and $\beta_1, \dots, \beta_m \geq 0$, if the convex set*

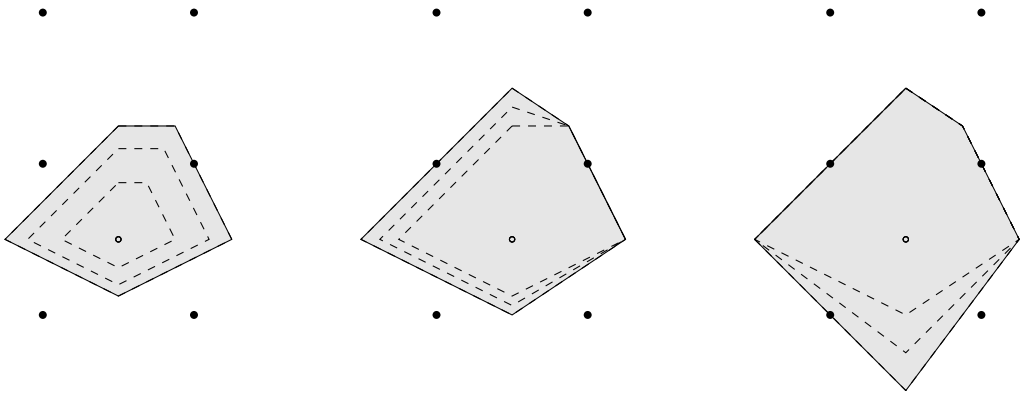
$$B(\varepsilon, \beta) = \operatorname{conv}\left(\{f\} \cup \left\{f + \frac{1}{\max\{\varepsilon, \beta_i\}} r^i : i \in \{1, \dots, m\}\right\}\right)$$

is lattice-free, then inequality (7) is valid for (C).

Proof. Similar to the proof of Lemma 1. □

A brief outline of the algorithm is as follows. First, all the variables β_i are set to zero. Then, the algorithm tries to find the smallest number $\varepsilon > 0$ such that $B(\varepsilon, \beta)$ is lattice-free. At this point, inequality (7) corresponds to the desired infinity cut. Next, for every ray r^i that has become tight with respect to (7), the algorithm updates β_i to ε. The process then repeats, and the algorithm tries to find, once again, the smallest $\varepsilon > 0$ such that $B(\varepsilon, \beta)$ is lattice-free. The algorithm stops after m iterations, since it is guaranteed, by then, that every ray has become tight. We illustrate the algorithm by a small example, before presenting its precise description.

Example 8. *Consider the continuous relaxation presented in Example 3. The algorithm starts by settings β to $(0, 0, 0, 0, 0)$. Figure 2a shows the set $B(\varepsilon, \beta)$ for values $\varepsilon = \frac{4}{3}, \frac{4}{5}, \frac{2}{3}$. Note that, as we decrease ε, the set $B(\varepsilon, \beta)$ grows. It is clear that, for any $\varepsilon < \frac{2}{3}$, the point $(1, 1)$ belongs*



(a) $\varepsilon \geq \frac{2}{3}$ (b) $\frac{2}{3} \geq \varepsilon \geq \frac{1}{2}$ (c) $\frac{1}{2} \geq \varepsilon \geq \frac{1}{4}$

Figure 2: Example of $B(\varepsilon, \beta)$ for different values of ε

to the interior of $B(\varepsilon, \beta)$, therefore the desired ε for the first iteration is $\frac{2}{3}$. The algorithm then sets β_2 and β_3 to $\frac{2}{3}$, and continues to decrease ε. Figure 2b shows $B(\varepsilon, \beta)$ for values $\varepsilon = \frac{2}{3}, \frac{4}{7}, \frac{1}{2}$. Note that, even as ε decreases, the vertices corresponding to rays r^2 and r^3 remain fixed. Since any $\varepsilon < \frac{1}{2}$ causes the point $(0, 1)$ to fall in the interior of $B(\varepsilon, \beta)$, the desired ε for the second iteration is $\frac{1}{2}$. The algorithm sets β_1 and β_5 to $\frac{1}{2}$ and continues decreasing ε. Figure 2c shows $B(\varepsilon, \beta)$ for values $\varepsilon = \frac{1}{2}, \frac{1}{3}, \frac{1}{4}$. Since $(0, 0)$ belongs to the interior of $B(\varepsilon, \beta)$ for any $\varepsilon < \frac{1}{4}$, the desired ε for the third iteration is $\frac{1}{4}$. The algorithm sets β_4 to $\frac{1}{4}$. In the following iterations, the set $B(\varepsilon, \beta)$ is lattice-free for any choice of ε, and no new rays become tight. The algorithm terminates then with $\beta = (\frac{1}{2}, \frac{2}{3}, \frac{2}{3}, \frac{1}{4}, \frac{1}{2})$, which gives us the coefficients for the infinity cut (level 5).

A precise description of our method is presented in Algorithm 9. The algorithm relies on a function Bound, which, given $x \in \mathbb{Z}^n$ and $\beta_1, \ldots, \beta_m \geq 0$, either returns the largest $\varepsilon_x > 0$ such that $B(\varepsilon_x, \beta)$ contains x, given that this number exists, or returns zero in case it does not. In the following, we prove that InfinityCut correctly computes the infinity cut (level m). First, we prove a technical lemma which shows that, at the end of each iteration of the loop in line 3, the value of the variable ε^k decreases.

Lemma 10. *If $\varepsilon^1, \ldots, \varepsilon^m$ are defined as in Algorithm 9, then*

$$\varepsilon^1 \geq \varepsilon^2 \ldots \geq \varepsilon^m.$$

Proof. Let $k \in \{2, \ldots, m\}$. Suppose, by contradiction, that $\varepsilon^{k-1} < \varepsilon^k$. Since $\varepsilon^{k-1} \geq 0$, we must have $\varepsilon^k > 0$. This implies that $\varepsilon^k = \varepsilon^k_x = \textsc{Bound}(\beta_1^{k-1}, \ldots, \beta_m^{k-1}, x)$, for some $x \in \mathbb{Z}^n$. By the definition of Bound, this implies $x \in B(\varepsilon^k, \beta_1^{k-1}, \ldots, \beta_m^{k-1})$. Next, we prove that

Algorithm 9

function InfinityCut
  $\beta_i^0 \leftarrow 0$, for every $i \in \{1, \dots, m\}$
  for $k \in \{1, \dots, m\}$ **do**
    $\varepsilon^k \leftarrow 0$
    for $x \in \mathbb{Z}^n$ **do**
      $\varepsilon_x^k \leftarrow \text{Bound}(\beta_1^{k-1}, \dots, \beta_m^{k-1}, x)$
      $\varepsilon^k \leftarrow \max\{\varepsilon^k, \varepsilon_x^k\}$
    $T^k \leftarrow \{i \in \{1, \dots, m\} : r^i \text{ is tight for } \sum_{i=1}^m \max\{\beta_i^{k-1}, \varepsilon^k\} s_i \geq 1\}$
    $\beta_i^k \leftarrow \max\{\beta_i^{k-1}, \varepsilon^k\}$, for every $i \in T^k$
    $\beta_i^k \leftarrow \beta_i^{k-1}$, for every $i \in \{1, \dots, m\} \setminus T^k$
  return $\beta_1^m, \dots, \beta_m^m$

$B(\varepsilon^k, \beta_1^{k-1}, \dots, \beta_m^{k-1}) = B(\varepsilon^k, \beta_1^{k-2}, \dots, \beta_m^{k-2})$. Indeed,

$$
\begin{aligned}
&B(\varepsilon^k, \beta_1^{k-1}, \dots, \beta_m^{k-1}) \\
&\quad = \text{conv}\left(\{f\} \cup \left\{f + \frac{1}{\max\{\varepsilon^k, \beta_i^{k-1}\}} r^i\right\}_{i=1}^m\right) \\
&\quad = \text{conv}\left(\{f\} \cup \left\{f + \frac{1}{\max\{\varepsilon^k, \varepsilon^{k-1}, \beta_i^{k-1}\}} r^i\right\}_{i \in T^{k-1}} \cup \left\{f + \frac{1}{\max\{\varepsilon^k, \beta_i^{k-2}\}} r^i\right\}_{i \in \{1,\dots,m\} \setminus T^{k-1}}\right) \\
&\quad = \text{conv}\left(\{f\} \cup \left\{f + \frac{1}{\max\{\varepsilon^k, \beta_i^{k-2}\}} r^i\right\}_{i=1}^m\right) \\
&\quad = B(\varepsilon^k, \beta_1^{k-2}, \dots, \beta_m^{k-2}).
\end{aligned}
$$

This implies that $x \in B(\varepsilon^k, \beta_1^{k-2}, \dots, \beta_m^{k-2})$. Therefore, if $\varepsilon_x^{k-1} = \text{Bound}(\beta_1^{k-2}, \dots, \beta_m^{k-2}, x)$, then we must have $\varepsilon_x^{k-1} \geq \varepsilon^k$. On the other hand, we know that $\varepsilon^{k-1} \geq \varepsilon_x^{k-1}$. A contradiction. □

Theorem 11. *If $k \in \{1, \dots, m\}$ and if $\beta_1^k, \dots, \beta_m^k$ and ε^k are defined as in Algorithm 9, then the inequality*

$$\sum_{i=1}^m \max\{\beta_i^{k-1}, \varepsilon^k\} s_i \geq 1 \tag{8}$$

is the infinity cut (level k).

Proof. First, we prove that the result holds at the end of the first iteration. In this case, equation (8) is given by

$$\sum_{i=1}^m \max\{\beta_i^0, \varepsilon^1\} s_i = \sum_{i=1}^m \varepsilon^1 s_i \geq 1.$$

By Lemma 7, in order to prove that this inequality is valid, it is sufficient to show that $B(\varepsilon^1, \beta^0)$ is lattice-free. Let $x \in \mathbb{Z}^n$ be an arbitrary lattice point and let $\varepsilon_x^1 = \text{Bound}(\beta^0, x)$. We prove that x is not in the interior of $B(\varepsilon^1, \beta^0)$. If $\varepsilon_x^1 = 0$, then there does not exist $\bar{\varepsilon} > 0$ such that

$x \in B(\bar{\varepsilon}, \beta^0)$. In particular, $x \notin B(\varepsilon^1, \beta^0)$, and we are done. Suppose, on the other hand, that $\varepsilon^1_x > 0$. By definition of BOUND, we known that $B(\varepsilon^1_x, \beta^0)$ does not contain x in its interior. Also, from the description of the algorithm, we have $\varepsilon^1 \geq \varepsilon^1_x$. Therefore, $B(\varepsilon^1, \beta^0)$ does not contain x in its interior either. Since the choice of x was arbitrary, we conclude that $B(\varepsilon^1, \beta^0)$ is lattice-free. Now we prove that ε^1 is as small as possible. If $\varepsilon^1 = 0$, there is nothing to prove. If $\varepsilon^1 > 0$, then, by the description of the algorithm, there exists $x \in \mathbb{Z}^n$ such that $\varepsilon^1 = \varepsilon^1_x = \text{BOUND}(\beta^0, x)$. By definition of BOUND, for any $\bar{\varepsilon}$ such that $0 < \bar{\varepsilon} < \varepsilon^1$, the set $B(\bar{\varepsilon}, \beta^0)$ contains x in its interior. Therefore, ε^1 cannot be decreased without generating an invalid cut.

Now let $k \in \{2, \dots, m\}$ and suppose that

$$\sum_{i=1}^{m} \max\{\beta_i^{k-2}, \varepsilon^{k-1}\} s_i \geq 1 \tag{9}$$

is the infinity cut (level $k-1$). We prove that

$$\sum_{i=1}^{m} \max\{\beta_i^{k-1}, \varepsilon^{k}\} s_i \geq 1 \tag{10}$$

is the infinity cut (level k). By Lemma 10, $\varepsilon^k \geq \varepsilon^{k+1}$. Consider a ray r^i such that $i \in T^{k-1}$. By definition, $\beta_i^{k-1} = \max\{\beta_i^{k-2}, \varepsilon^{k-1}\}$, which implies that $\max\{\beta_i^{k-1}, \varepsilon^k\} = \max\{\beta_i^{k-2}, \varepsilon^{k-1}, \varepsilon^k\} = \max\{\beta_i^{k-2}, \varepsilon^k\}$. Suppose, on the other hand, that r^i is such that $i \notin T^{k-1}$. Since the set of tight rays only increases after each iteration, $i \notin T^1, \dots, T^{k-2}$ and we have $\beta_i^{k-1} = 0$. Therefore, $\max\{\beta_i^{k-1}, \varepsilon^k\} = \varepsilon^k$. Equation (10) can be rewritten as

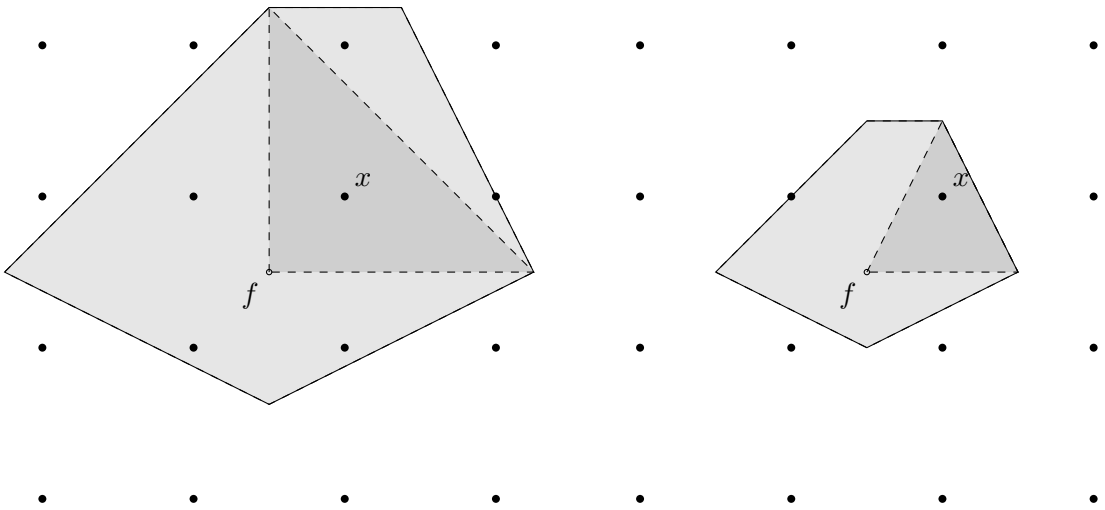
$$\sum_{i \in T^{k-1}} \max\{\beta_i^{k-2}, \varepsilon^{k-1}\} s_i + \sum_{\substack{i=1 \\ i \notin T^{k-1}}}^{m} \varepsilon^k s_i \geq 1. \tag{11}$$

This form matches item (ii) of Definition 5. More precisely, notice that the coefficients corresponding to the tight rays T^{k-1} have been kept unchanged when going from (9) to (11). Also, all the coefficients for the remaining rays are equal to some $\varepsilon^k \in \mathbb{R}_+$. It remains to show that (11) is valid and that ε^k cannot be made any smaller. The proof is similar to the base case. □

Before we continue, we make a small modification to Algorithm 9 in order to make it finite. Note that the algorithm currently never terminates because of the loop in Line 5. If ε^k is initialized to a very small but strictly positive constant η, instead of zero, then this loop can be rewritten as

$$\textbf{for } x \in \mathbb{Z}^n \cap B(\varepsilon^k, \beta^{k-1})$$

Since $B(\varepsilon^k, \beta^{k-1})$ is bounded, there are a finite number of lattice points contained in it, and the loop now terminates in finite time. Another advantage is that, whenever ε^k gets updated inside of the loop, the set $B(\varepsilon^k, \beta^{k-1})$ shrinks, causing the loop to terminate earlier. We note



(a) First iteration (b) Second iteration

Figure 3: Examples for Bound

that this modification does change the output of the algorithm, and causes the algorithm to produce a slightly weaker cut whenever the infinity cut for the relaxation turns out to be a split. However, we can choose the constant η to be small enough, so that the practical impact is negligible. In our experiments, this constant was set to 10^{-3}.

Next, we present an algorithm for evaluating the function Bound. With our modification, this function now behaves as follows. Given $x \in \mathbb{Z}^n$ and $\beta_1, \ldots, \beta^m \geq 0$, if there exists $\varepsilon \geq \eta$ such that $B(\varepsilon, \beta)$ contains x, the function returns the largest such ε. If no such ε exists, the function returns η. An outline of the algorithm is as follows. We start by setting $\varepsilon = \eta$ and verifying whether $B(\beta, \varepsilon)$ contains x in its interior. If so, there must exist a subset of linearly independent rays $r^{i_1}, \ldots, r^{i_s}$, where $s \leq n$, such that x belongs to the relative interior of the set

$$S(r^{i_1}, \ldots, r^{i_s}, \beta_{i_1}, \ldots, \beta_{i_s}, \varepsilon) = \operatorname{conv}\left(\{f\} \cup \left\{f + \frac{1}{\max\{\varepsilon, \beta_{i_j}\}} r^{i_j} : j \in \{1, \ldots, s\}\right\}\right).$$

Note that this set was obtained from $B(\beta, \varepsilon)$ by dropping some of the rays r^i, and therefore has a simpler structure. The algorithm then computes ε such that x belongs to the boundary of this simplified set, which can be done easily. Next, the algorithm verifies whether x still belongs to the interior of the original set $B(\beta, \varepsilon)$, and, if so, the process repeats. The algorithm stops when x is no longer in the interior of $B(\varepsilon, \beta)$. We show the execution of this algorithm in a small example.

Example 12. *Consider, once again, the continuous relaxation presented in Example 3. Let* $\beta = (0, 0, 0, 0, 0), x = (1, 1)$ *and suppose that we want to compute* Bound(β, x). *The algorithm starts by setting* ε *to some very small* η. *The set* $B(\varepsilon, \beta)$ *is shown in Figure 3a. Clearly,* x *belongs to the interior of* $B(\varepsilon, \beta)$. *Notice that* x *also belongs to the relative interior of* $S(r^1, r^3, \beta_1, \beta_3, \varepsilon)$ *for any* $\varepsilon < \frac{1}{2}$. *The algorithm then sets* ε *to* $\frac{1}{2}$ *and verifies that* x *still belongs to the interior*

of $B(\varepsilon, \beta)$, as shown in Figure 3b. Furthermore, x belongs to the interior of $S(r^2, r^3, \beta_2, \beta_3, \varepsilon)$ for any $\varepsilon < \frac{2}{3}$. After setting ε to $\frac{2}{3}$, it can be verified that x no longer belongs to the interior of $B(\varepsilon, \beta)$, and the algorithm stops.

A precise description of BOUND is given in Algorithm 13. In order to find the desired subset of linearly independent rays $r^{i_1}, \ldots, r^{i_s}$, or to prove that no such subset exists, BOUND calls the function FINDVIOLATEDCONE, which is described precisely in Algorithm 14, and in order to find ε such that x belongs to the boundary of $S(r^{i_1}, \ldots, r^{i_s}, \beta^{i_1}, \ldots, \beta^{i_s}, \varepsilon)$, BOUND calls the function CONEBOUND, which is described precisely in Algorithm 15. In the following, we prove that these algorithms indeed work as intended. We start with FINDVIOLATEDCONE.

Algorithm 13

function BOUND($\beta_1, \ldots, \beta_m, x$)
  $\varepsilon \leftarrow \eta$
  loop
    $Q \leftarrow$ FINDVIOLATEDCONE($\beta_1, \ldots, \beta_m, \varepsilon, x$)
    if $Q = \emptyset$ **then**
      break
    else
      Let $Q = \{i_1, \ldots, i_s\}$, where $\beta_{i_1} \geq \ldots \geq \beta_{i_s}$
      $\varepsilon \leftarrow \max\{\varepsilon,$ CONEBOUND($r^{i_1}, \ldots, r^{i_s}, \beta_{i_1}, \ldots, \beta_{i_s}, x$)$\}$
  return ε

Algorithm 14

function FINDVIOLATEDCONE($\beta_1, \ldots, \beta_m, \varepsilon, x$)
  Let λ^* be an optimal basic feasible solution for

$$\begin{aligned} \textbf{minimize} \quad & \sum_{j=1}^{m} \lambda_j \\ \textbf{subject to} \quad & x = f + \sum_{j=1}^{m} \frac{\lambda_j}{\max\{\varepsilon, \beta_j\}} r^j \\ & \lambda_r \geq 0 \end{aligned}$$

  if $\sum_{j=1}^{m} \lambda_j^* \geq 1$ **then return** $\emptyset$
  else return $\{i \in \{1, \ldots, m\} : \lambda_i^* > 0\}$

Proposition 16. *Let $\varepsilon > 0$. If* FINDVIOLATEDCONE *returns a non-empty set $\{i_1, \ldots, i_s\} \subseteq \{1, \ldots, m\}$, then*

(i) $r^{i_1}, \ldots, r^{i_s}$ are linearly independent

(ii) x belongs to the relative interior of $S(r^{i_1}, \ldots, r^{i_s}, \beta_{i_1}, \ldots, \beta_{i_s}, \varepsilon)$

If FINDVIOLATEDCONE *returns $\emptyset$, then no subset $\{i_1, \ldots, i_s\} \subseteq \{1, \ldots, m\}$ satisfies (i) and (ii).*

Algorithm 15

function ConeBound($q^1, \ldots, q^s, \beta^1, \ldots, \beta^s, x$)
  Let $\lambda \in \mathbb{R}^s$ such that $x = f + \sum_{i=1}^{s} \lambda_i q^i$
  for $t = 0$ **to** $s - 1$ **do**
    Let $\varepsilon_t \leftarrow \dfrac{1 - \sum_{i=1}^{t} \lambda_i \beta_i}{\sum_{i=t+1}^{s} \lambda_i}$
    if $\varepsilon_t > \beta_{t+1}$ **then return** ε_t
  return 0

Proof. Suppose FindViolatedCone returns a non-empty subset $\{i_1, \ldots, i_s\} \subseteq \{1, \ldots, m\}$, and let λ^* be defined as in the algorithm. Property (i) follows immediately from the fact that λ^* is a basic feasible solution. Now we prove that x belongs to the relative interior of $S := S(r^{i_1}, \ldots, r^{i_s}, \beta_{i_1}, \ldots, \beta_{i_s}, \varepsilon)$. If we denote by v^j the point $f + \frac{1}{\max\{\beta_j, \varepsilon\}} r^j$, for $j \in \{1, \ldots, m\}$, then the vertices of S are $f, v^{i_1}, \ldots, v^{i_s}$. It suffices to prove that x is a strict convex combination of these vertices. Recall that $\sum_{j=1}^{m} \lambda_j^* < 1$. Since each $\lambda_j^* > 0$, this also implies $\lambda_j^* < 1$, for every $j \in \{1, \ldots, m\}$. Furthermore, if we define $\lambda_0^* = 1 - \sum_{j=1}^{m} \lambda_j^*$, then $0 > \lambda_0^* > 1$. We have

$$\begin{aligned} x &= f + \sum_{j=1}^{m} \frac{\lambda_j^*}{\max\{\varepsilon, \beta_j\}} r^j \\ &= f + \sum_{j=1}^{s} \lambda_{i_j}^* \left(v^{i_j} - f \right) \\ &= \lambda_0^* f + \lambda_{i_1}^* v^{i_1} + \ldots + \lambda_{i_s}^* v^{i_s}. \end{aligned}$$

We conclude that (ii) is satisfied.

Now suppose FindViolatedCone returns the empty set. In this case, $\sum_{j=1}^{m} \lambda_j^* \geq 1$. Furthermore, assume by contradiction that there exists a non-empty set $\{i_1, \ldots, i_s\} \subseteq \{1, \ldots, m\}$ satisfying (i) and (ii). Let the points v^j be as defined previously. We know that there exist $0 > \lambda_0, \lambda_{i_1}, \ldots, \lambda_{i_s} > 1$ such that

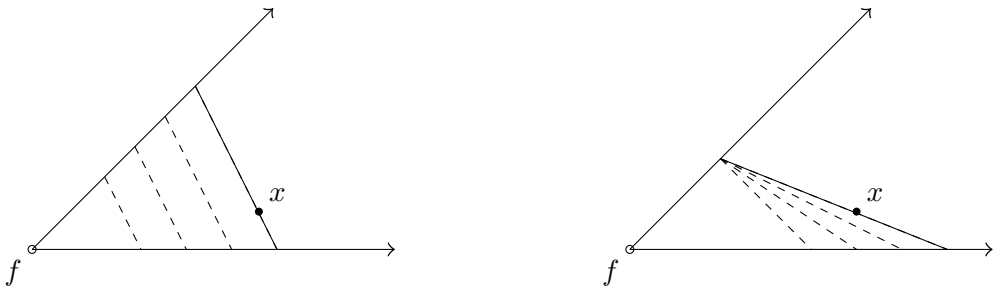
$$\begin{cases} x = \lambda_0 f + \lambda_{i_1} v^{i_1} + \ldots + \lambda_{i_s} v^{i_s} \\ \lambda_0 + \lambda_{i_1} + \ldots + \lambda_{i_s} = 1. \end{cases}$$

Let $\lambda_j = 0$ for every $j \in \{1, \ldots, m\} \setminus \{i_1, \ldots, i_s\}$. It is not hard to verify that

$$x = f + \sum_{j=1}^{m} \frac{\lambda_j}{\max\{\varepsilon, \beta_j\}} r^j.$$

Therefore, $\lambda_1, \ldots, \lambda_m$ is a feasible solution to the LP in the algorithm. Furthermore, $\sum_{j=1}^{m} \lambda_j = 1 - \lambda_0 < 1 \leq \sum_{j=1}^{m} \lambda_j^*$. This implies that λ^* is not an optimal solution, a contradiction. We conclude that no subset $\{i_1, \ldots, i_s\} \subseteq \{1, \ldots, m\}$ satisfying (i) and (ii) exists. □

Next, we focus on the function ConeBound. First, we present a technical lemma that



(a) Computation of ε_0 (b) Computation of ε_1

Figure 4: Illustration for ConeBound

justifies the formula for ε_t appearing in the algorithm. Then, we prove that ConeBound finds the correct answer.

Lemma 17. *Let $x \in \mathbb{Z}^n$, let $\beta_1, \ldots, \beta_s \in \mathbb{R}$ such that $\beta_1 \geq \ldots \geq \beta_s \geq 0$, let $q^1, \ldots, q^s \in \mathbb{R}^n$ be linearly independent vectors such that x belongs to the relative interior of $f + \operatorname{cone}(q^1, \ldots, q^s)$, and let $\lambda_1, \ldots, \lambda_s > 0$ such that $x = f + \sum_{i=1}^s \lambda_i q^i$. Let $t \in \{0, \ldots, s-1\}$, and define*

$$\varepsilon_t = \frac{1 - \sum_{i=1}^t \lambda_i \beta_i}{\sum_{i=t+1}^s \lambda_i}.$$

If $\beta_1, \ldots, \beta_t > 0$ and $\varepsilon_t > 0$, then ε_t is the largest $\varepsilon \in \mathbb{R}$ such that

$$x \in \operatorname{conv}\left(\{f\} \cup \left\{f + \frac{1}{\beta_i} q^i\right\}_{i=1}^t \cup \left\{f + \frac{1}{\varepsilon} q^i\right\}_{i=t+1}^s\right) =: S_t(\varepsilon).$$

Proof. Let $v^i = f + \frac{1}{\beta_i} q^i$ for $i \in \{1, \ldots, t\}$ and $v^i = f + \frac{1}{\varepsilon_t} q^i$ for $i \in \{t+1, \ldots, s\}$. We prove that x is a convex combination of $v^1, \ldots, v^s$. This implies that $x \in S_t(\varepsilon_t)$, and it is easy to see that, for any $\varepsilon > \varepsilon_t$, we have $x \notin S_t(\varepsilon)$. To prove this claim, let

$$\gamma_i = \begin{cases} \lambda_i \beta_i & \text{for } i \in \{1, \ldots, t\} \\ \lambda_i \varepsilon_t & \text{for } i \in \{t+1, \ldots, s\}. \end{cases}$$

By hypothesis, we have $\lambda_1, \ldots, \lambda_s > 0$ and $\beta_1, \ldots, \beta_t, \varepsilon_t > 0$. This implies that $\gamma_1, \ldots, \gamma_t > 0$. Furthermore, the sum of $\gamma_1, \ldots, \gamma_t$ equals one, since

$$\sum_{i=1}^s \gamma_i = \sum_{i=1}^t \lambda_i \beta_i + \varepsilon_t \left(\sum_{i=t+1}^s \lambda_i\right) = \sum_{i=1}^t \lambda_i \beta_i + 1 - \sum_{i=1}^t \lambda_i \beta_i = 1.$$

It only remains to prove that $x = \sum_{i=1}^s \gamma_i v^i$. We have:

$$\sum_{i=1}^s \gamma_i v^i = \sum_{i=1}^s \gamma_i f + \sum_{i=1}^t \lambda_i \beta_i \frac{1}{\beta_i} q^i + \sum_{i=t+1}^s \lambda_i \varepsilon_t \frac{1}{\varepsilon_t} q^i = f + \sum_{i=1}^s \lambda_i q^i.$$

□

Proposition 18. *Let $f, x, \beta_1, \dots, \beta_s$ and $q^1, \dots, q^s$ be defined as in Lemma 17. If there exists $\varepsilon > 0$ such that x belongs to the relative interior of $S(q^1, \dots, q^s, \beta_1, \dots, \beta_s, \varepsilon) := S(\varepsilon)$, then* ConeBound *returns the largest $\varepsilon \in \mathbb{R}$ such that $x \in S(\varepsilon)$. Otherwise,* ConeBound *returns zero.*

Proof. Suppose there exists $\tilde{\varepsilon} > 0$ such that x belongs to the interior of $S(\tilde{\varepsilon})$. First, we prove that there exists $\varepsilon^* > \beta_s$ such that $x \in S(\varepsilon^*)$. If $\tilde{\varepsilon} > \beta_s$, there is nothing to prove, so suppose $\tilde{\varepsilon} \leq \beta_s$. Since $\beta_s \leq \ldots \leq \beta_1$, this implies $S(\tilde{\varepsilon}) = S(\beta_s)$. Therefore, x belongs to the interior of $S(\beta_s)$. It is not hard to see that, for a very small $\theta > 0$, we still have $x \in S(\beta_s + \theta)$. Therefore, there exists $\varepsilon^* > \beta_s$ such that $x \in S(\beta_s + \theta)$. Furthermore, we may assume that ε^* is the maximum value satisfying this property. Indeed, if such maximum does not exist, then $x \in S(\varepsilon)$ for every $\varepsilon > \varepsilon^*$, which implies $f = x$, a contradiction. Next, we prove that ConeBound returns ε^*. Let $\beta_0 = \infty$. There exists $k \in \{0, \dots, s-1\}$ such that $\beta_{k+1} < \varepsilon^* \leq \beta_k$. By Lemma 17, we know that $\varepsilon^* = \varepsilon_k$. We prove that ConeBound returns ε_k. Since $\varepsilon_k = \varepsilon^* > \beta_{k+1}$, it is sufficient to show that the loop does not finish before considering ε_k. That is, that ConeBound does not return $\varepsilon_0, \dots, \varepsilon_{k-1}$. Suppose, by contradiction, that ConeBound returns ε_l, for some $l \in \{0, \dots, k-1\}$. Then $\varepsilon_l > \beta_{l+1} \geq \beta_k \geq \varepsilon^*$. Furthermore, $x \in S(\varepsilon_l)$, contradicting the maximality of ε^*. We conclude that the algorithm returns ε^*.

Now, suppose that ConeBound returns a non-zero value. Clearly, the returned value must be ε_k, for some $k \in \{0, \dots, s-1\}$. Furthermore, $x \in S(\varepsilon_k)$ and $\varepsilon_k > \beta_{k+1} \geq \beta_s$. Let $\tilde{\varepsilon} = \frac{\varepsilon_k + \beta_s}{2}$. It is not hard to see that $\tilde{\varepsilon} > 0$ and that x belongs to the interior of $S(\tilde{\varepsilon})$. We conclude that there exists $\varepsilon > 0$ such that x belongs to the interior of $S(\varepsilon)$.

□

We end this section with a proof of correctness for the function Bound.

Proposition 19. *Let $x \in \mathbb{Z}^n$, let $\beta_1, \dots, \beta_m \geq 0$ and let $\eta > 0$. If there exists $\varepsilon \geq \eta$ such that x belongs to the interior of $B(\beta_1, \dots, \beta_m, \varepsilon) =: B(\varepsilon)$, then* Bound *returns the largest $\varepsilon \geq \eta$ such that $x \in B(\varepsilon)$. Otherwise,* Bound *returns η.*

Proof. First, suppose there does not exist $\varepsilon \geq \eta$ such that x belongs to the interior of $B(\varepsilon)$. Then, clearly, x does not belong to the interior of $B(\eta)$ and, by Proposition 16, the function FindViolatedCone returns the empty set. The loop then breaks during the first iteration. Therefore, Bound returns η.

Now suppose there exists $\varepsilon \geq \eta$ such that x belongs to the interior of $B(\varepsilon)$. Since $B(\varepsilon) \subseteq B(\eta)$, then x also belongs to the interior of $B(\eta)$. This implies that the value ε^* returned by Bound equals the value returned by ConeBound, for some subset of rays $r^{i_1}, \dots, r^{i_s}$. By Proposition 18, we have $x \in B(\varepsilon^*)$. Now suppose, by contradiction, that there exists $\tilde{\varepsilon} > \varepsilon^*$ such that $x \in B(\tilde{\varepsilon})$. This implies that x belongs to the interior of $B(\varepsilon^*)$, a contradiction. We conclude that ε^* is the largest $\varepsilon \geq \eta$ such that $x \in B(\varepsilon)$. □

4 Computational experiments

In order to evaluate the strength of infinity cuts, we implemented a cut generator and tested it with problems from the MIPLIB. Our main performance indicator was the integrality gap closed at the root node of the branch-and-bound tree. We compared our cuts against Gomory Mixed-Integer (GMI) cuts, and also against an exact separator for continuous multi-row intersection cuts implemented by Louveaux, Poirrier and Salvagnin [11]. Our main goal was to determine how close to the maximum theoretical gap closure would infinity cuts bring us. Another side goal was to evaluate the efficiency our cut generator.

The cut generator was implemented in C and compiled with GNU GCC 5.4.0. The generator makes use of IBM ILOG CPLEX 12.7.0 as an LP/MIP solver. The complete source code has been made available online [14]. We conducted our experiments on an Intel Xeon E5-2630v2 2.6 GHz with 64GB of memory, with a time limit of 15 CPU minutes.

4.1 Generating infinity cuts from the tableau

When generating a multi-row intersection cut from the simplex tableau, the first step is to select a suitable combination of tableau rows that will be used to derive the cut. Suppose that the simplex tableau is given by

$$x_i + \sum_{j \in N} \bar{a}_{ij} x_j = b_i, \qquad i \in B$$

where x_i, for $i \in B$, are the basic variables, and x_j, for $j \in N$, are the non-basic variables. In principle, any subset of tableau rows $I \subseteq B$ can be used, as long as two conditions are satisfied: first, only rows corresponding to integral basic variables are selected; and second, at least one row with a fractional right-hand side is included in the subset. In practice, however, the number of subsets satisfying these two conditions is excessively large for all but very small instances. Our cut generator, therefore, filtered down the list of row subsets even further, as described next.

As a first step, the rows of the simplex tableau were sorted according to the fractionality of their right-hand side. More specifically, the rows were sorted increasingly according to $\left|\hat{b}_i - \frac{1}{2}\right|$, where $\hat{b}_i$ is the fractional part of b_i. In this way, rows having right-hand sides closer to $\frac{1}{2}$ received higher priority, while rows having right-sides closer to 0 or 1 were given lower priority. Then, only 300 tableau rows with the most fractional right-hand sides were kept, while the remaining rows were discarded.

Next, for each pair of remaining tableau rows, we computed how similar their support was. More specifically, given two tableau rows $x_p + \sum_{j \in N} \bar{a}_{pj} x_j = b_p$ and $x_q + \sum_{j \in N} \bar{a}_{qj} x_j = b_q$, their *affinity* is given by the formula

$$\textsc{Affinity}(p, q) = \frac{2 \times |\{j : \bar{a}_{pj} \neq 0 \text{ and } \bar{a}_{qj} \neq 0\}|}{|\{j : \bar{a}_{pj} \neq 0\}| + |\{j : \bar{a}_{qj} \neq 0\}|.}$$

Note that, for every pair of tableau rows, this formula returns one if $\bar{a}_p$ and $\bar{a}_q$ have exactly the same support and zero if their supports are completely disjoint. This heuristic was also used by Louveaux, Poirrier and Salvagnin [11]. Pairs of tableau rows with affinity less than a certain threshold were considered incompatible. The generator derived one infinity cut for each subset of tableau rows containing only pairwise compatible rows.

After selecting a subset $I = \{i_1, \ldots, i_k\} \subseteq B$ to generate the cut, another issue was to decide what set of rays to use when generating the infinity cut. On one hand, we wanted the set of rays to be as close as possible to the original columns of the simplex tableau. On the other hand, the running time of the algorithm described in the previous section is highly sensitive on the number of rays. For instances having a large number of variables, it proved to be too slow to be useful in practice. Our separator, therefore, performed the following heuristic filtering to reduce the number of rays considered. For each non-basic variable x_j, a ray $r^j = (\bar{a}_{ij} : i \in B)$ was created. Rays having zeros in all components were discarded. Then, for each pair of rays that were very similar to each other, only one was kept. More specifically, for every pair of rays r^1, r^2, we computed $\|r^1 - r^2\|$, and, whenever this value was smaller than a certain threshold, one of these rays was discarded. The threshold was initially set to infinity and then progressively decreased until at most 100 rays were selected.

4.2 Cut generating procedure

Our cut generator performed the following steps. First, the linear relaxation of the problem was solved, and a certain basic solution with value z_{LP} was obtained. The optimal simplex tableau was stored. Although this LP relaxation was later strengthened with additional constraints (cuts) and solved again, only this first optimal simplex tableau was used to generate all subsequent cutting planes. We thus obtain only rank-1 cuts. Next, one GMI cut was generated from each suitable tableau row, and added to the relaxation. The relaxation was solved again, and a certain solution with value z_{GMI} was found. Then, one infinity cut was generated at a time, for each suitable subset of tableau rows, as described in the previous subsection. If the infinity cut did not cut the current basic optimal solution, the cut was discarded. Otherwise, the infinity cut was added to the relaxation, the relaxation was solved once again, and the current basic solution was updated. At the end of the procedure, a new solution with value z_{INF} was obtained. Finally, we also ran, independently from our generator, the exact separator of [11] for continuous multi-row cuts (i.e., disregarding the integrality of nonbasic variables), and obtained a solution with value z_{EXACT}. Note that the latter separator (i) considers all suitable pairs of rows, and (ii) runs multiple round of rank-1 cuts until none can be found, whereas (i) infinity cuts were generated only for the heuristic selection of multirow models described above, and (ii) only one infinity cut exists for each such multirow model. On the other hand, because different bases would yield slightly different results, we were careful to use the same original basis for all cut generators.

The set of instances was the MIPLIB 3.0, which contains 65 instances coming from real-

world applications. We chose this version of the MIPLIB so that the slower exact separator could produce more reliable bounds. Out of the 65 instances, we eliminated those that were infeasible, that had no known optimal solution, or whose LP relaxation had the same objective value as the original problem. Three instances were also eliminated because of their size. For each of the remaining 56 instances, we computed:

- G-GMI, the gap closed by the inclusion of GMI cuts:

$$\text{G-GMI} = \frac{z_{GMI} - z_{LP}}{z_{OPT} - z_{LP}},$$

- G-INF, the gap closed by the inclusion of infinity cuts, in addition to the GMI cuts:

$$\text{G-INF} = \frac{z_{INF} - z_{LP}}{z_{OPT} - z_{LP}},$$

- G-EXACT, the gap closed by using the exact separator for continuous multi-row cuts:

$$\text{G-EXACT} = \frac{z_{EXACT} - z_{LP}}{z_{OPT} - z_{LP}},$$

- REL, a relative measure of strength, in terms of gap closure, when compared to the exact separator, given precisely by:

$$\text{REL} = \frac{z_{INF} - z_{GMI}}{z_{EXACT} - z_{GMI}}.$$

- T-TIME, the total CPU running time spent by the infinity cut generator (min:sec),
- C-TIME, the CPU running time that the infinity cut generator, on average, took to generate a single cut, in milliseconds,
- N-CUTS, the number of infinity cuts added to the relaxation.

We note that the number of calls to the cut generator is given by $\frac{T-TIME}{C-TIME}$ (after putting both in the same time unit), though that number may be slightly off due to rounding of either T-TIME or C-TIME.

4.3 Results and discussion

Table 1 shows the computational results for infinity cuts generated from two rows of the simplex tableau. We show both the strength of non-lifted infinity cuts, where no attempt was made to strengthen the coefficients of the non-basic variables, and for lifted infinity cuts, where the coefficients for the integral non-basic variables were strengthened using the trivial lifting procedure. Remark that, in theory, only non-lifted infinity cuts are directly comparable to the results we present with the exact separator, since lifted cuts are valid for a stronger multirow

			Non-Lifted 2-Row					Lifted 2-Row				
Instance	G-GMI	G-EXACT	G-INF	REL (%)	T-TIME (m:s)	C-TIME (ms)	N-CUTS	G-INF	REL (%)	T-TIME (m:s)	C-TIME (ms)	N-CUTS
10teams	57.1	57.1	57.1	—	04:18	52	0	57.1	—	15:00	527	0
air03	100.0	100.0	100.0	—	01:53	51	0	100.0	—	15:00	1410	0
bell3a	39.0	52.5	48.1	66.9	04:36	882	6	49.0	74.0	00:33	104	8
bell5	14.5	17.8	14.6	1.2	00:00	2	2	14.6	1.2	00:03	10	2
blend2	16.0	18.3	16.0	0.0	00:00	5	0	16.1	1.7	00:00	16	1
cap6000	41.6	41.6	41.6	—	00:00	651	0	41.6	—	00:02	2633	0
danoint	1.7	1.7	1.7	—	00:06	14	3	1.7	—	00:09	8	1
dcmulti	48.1	50.2	49.1	47.4	00:01	3	8	49.3	58.8	00:03	5	9
egout	65.1	82.3	75.9	63.0	00:00	0	2	75.9	63.0	00:00	2	2
fiber	70.3	71.6	71.6	99.7	00:02	1	3	73.9	274.9	00:36	26	12
fixnet6	22.6	26.6	27.3	119.2	00:00	2	9	30.9	209.1	00:01	20	16
flugpl	10.8	13.7	10.8	0.0	00:00	0	0	10.8	0.0	00:00	1	0
gen	1.3	6.8	8.1	123.3	00:00	1	10	6.2	88.9	00:00	8	15
gesa2	27.7	45.0	29.2	8.9	00:00	1	22	29.0	7.6	00:02	4	19
gesa2_o	29.2	56.8	34.8	20.2	00:01	1	30	34.8	20.4	00:06	5	28
gesa3	19.5	46.4	30.8	42.2	01:01	18	15	30.5	40.7	00:33	10	16
gesa3_o	59.1	72.4	64.4	40.0	00:05	1	15	65.0	44.3	00:56	13	14
gt2	59.9	100.0	89.7	74.4	00:00	1	2	90.6	76.7	00:07	29	2
harp2	24.1	26.8	24.2	1.9	00:02	5	3	24.2	2.0	01:14	155	6
khb05250	74.3	79.9	74.3	0.0	00:00	19	0	74.3	0.0	00:00	16	0
l152lav	12.9	12.9	12.9	—	01:10	21	0	13.0	—	15:00	518	6
lseu	37.0	37.3	37.0	0.0	00:01	6	0	39.0	732.1	00:05	21	4
markshare1	0.0	0.0	0.0	—	00:00	2	0	0.0	—	00:00	14	1
markshare2	0.0	0.0	0.0	—	00:00	2	0	0.0	—	00:00	27	0
mas74	6.7	6.7	6.7	0.0	00:00	4	0	7.0	675.8	00:04	67	10
misc03	15.0	15.0	15.0	—	00:02	2	0	15.0	—	00:55	29	0
misc06	31.4	37.6	37.6	99.7	00:00	4	5	41.3	157.9	00:00	4	5
misc07	0.7	0.7	0.7	—	00:09	2	0	0.7	—	03:19	46	0

Table 1: Strength of 2-row infinity cuts

			Non-Lifted 2-Row					Lifted 2-Row				
Instance	G-GMI	G-EXACT	G-INF	REL (%)	T-TIME (m:s)	C-TIME (ms)	N-CUTS	G-INF	REL (%)	T-TIME (m:s)	C-TIME (ms)	N-CUTS
mitre	84.0	84.0	84.0	—	00:02	2	1	84.0	—	01:30	94	3
mkc	8.7	10.6	8.7	0.0	00:02	11	0	8.7	0.0	00:41	216	1
mod008	21.6	21.6	21.6	—	00:00	8	0	21.6	—	00:02	167	0
mod010	100.0	100.0	100.0	—	00:45	15	0	100.0	—	15:00	486	0
mod011	33.0	34.6	33.0	0.0	00:06	13	0	33.0	0.0	00:12	26	0
modglob	17.3	26.5	17.5	2.4	00:01	6	5	17.5	2.7	00:02	12	5
nw04	66.1	66.1	66.1	—	00:57	350	0	66.1	—	15:09	11968	0
p0033	34.4	35.0	34.4	0.0	00:00	0	0	34.8	70.6	00:00	7	2
p0201	6.5	6.5	6.5	—	00:02	1	0	6.5	—	01:03	27	0
p0282	3.2	8.2	6.2	60.8	00:11	6	4	6.2	60.8	00:40	21	10
p0548	61.7	63.2	61.7	0.0	00:01	5	0	61.7	0.3	00:06	31	2
p2756	58.0	58.0	58.0	—	00:02	14	3	58.0	—	00:08	40	1
pk1	0.0	0.0	0.0	—	00:00	3	0	0.0	—	00:01	17	0
pp08a	54.9	66.7	67.1	103.3	00:00	2	26	67.0	103.2	00:00	3	26
pp08aCUTS	33.8	39.3	37.7	70.0	00:03	6	11	37.8	73.0	00:04	7	12
qiu	1.4	1.7	1.7	95.8	00:05	7	9	1.7	94.2	00:06	7	11
qnet1	11.7	14.6	12.0	8.4	00:53	11	5	14.7	106.5	15:00	222	14
qnet1_o	19.8	25.4	22.3	45.2	00:04	5	17	23.6	67.6	01:20	95	27
rentacar	19.8	20.3	19.8	0.0	00:00	14	0	19.8	0.0	00:01	21	0
rgn	12.0	12.0	12.0	—	00:00	1	0	15.1	—	00:00	9	4
rout	0.3	5.2	0.3	0.0	00:34	34	0	0.3	0.0	01:43	103	3
set1ch	29.4	55.6	44.0	55.7	00:00	2	87	44.0	55.7	00:00	4	87
seymour	6.6	6.6	6.6	—	01:13	15	0	6.6	—	14:59	209	0
stein27	0.0	0.0	0.0	—	00:02	1	0	0.0	—	00:26	5	0
stein45	0.0	0.0	0.0	—	00:04	1	0	0.0	—	00:47	9	0
swath	8.7	10.7	8.7	0.0	00:46	21	4	10.6	93.3	15:00	792	5
vpm1	29.9	31.3	30.7	54.8	00:00	1	5	33.8	280.9	00:00	5	5
vpm2	13.9	19.4	15.6	31.2	00:00	1	5	18.6	83.9	00:00	5	17
	29.0	33.9	31.3	47.8	00:20	41.2	5.7	31.8	57.8	02:27	363.1	7.4

Table 1: Strength of 2-row infinity cuts (continued)

			Non-Lifted 3-Row					Lifted 3-Row				
Instance	G-GMI	G-EXACT	G-INF	REL (%)	T-TIME (m:s)	C-TIME (ms)	N-CUTS	G-INF	REL (%)	T-TIME (m:s)	C-TIME (ms)	N-CUTS
10teams	57.1	57.1	57.1	—	15:00	116	0	57.1	—	15:00	557	0
air03	100.0	100.0	100.0	—	15:00	212	0	100.0	—	15:00	1465	0
bell3a	39.0	56.0	48.2	53.8	05:23	87	9	49.0	58.7	03:29	57	8
bell5	14.5	18.0	14.6	1.2	01:01	27	2	14.6	1.2	02:40	70	2
blend2	16.0	18.8	16.0	0.0	00:02	42	0	16.1	1.4	00:06	98	1
cap6000	41.6	41.6	41.6	—	00:10	10416	0	41.6	—	00:11	11880	0
danoint	1.7	1.7	1.7	—	01:50	70	3	1.7	—	03:33	47	1
dcmulti	48.1	52.5	49.1	23.4	01:56	23	12	49.6	36.2	03:34	43	31
egout	65.1	84.0	76.3	59.2	00:00	11	4	76.3	59.2	00:00	17	4
fiber	70.3	71.6	71.6	99.7	03:13	42	4	74.1	295.2	13:24	176	19
fixnet6	22.6	26.8	27.8	124.3	00:09	31	22	30.9	200.0	00:48	166	20
flugpl	10.8	62.7	39.1	54.6	00:02	30	3	39.1	54.6	00:03	36	3
gen	1.3	18.1	18.6	103.1	00:43	92	21	18.6	103.1	01:08	147	31
gesa2	27.7	45.0	30.1	14.0	01:03	19	36	30.3	14.9	02:30	45	40
gesa2_o	29.2	56.8	35.0	21.0	02:14	23	36	35.1	21.3	05:12	54	41
gesa3	19.5	46.4	31.6	44.9	02:07	21	19	31.6	45.0	02:13	21	20
gesa3_o	59.1	72.4	64.4	40.0	01:19	10	15	65.0	44.3	06:03	46	14
gt2	59.9	100.0	89.7	74.4	01:25	56	2	90.6	76.7	04:53	192	2
harp2	24.1	26.9	24.2	1.8	05:15	97	5	24.2	1.8	15:00	853	6
khb05250	74.3	80.4	74.3	0.0	00:01	17	0	74.3	0.0	00:01	19	0
l152lav	12.9	12.9	12.9	—	15:00	143	0	12.9	—	15:00	557	2
lseu	37.0	37.6	37.3	49.9	00:57	42	4	39.0	349.7	03:33	157	5
markshare1	0.0	0.0	0.0	—	00:02	81	0	0.0	—	00:06	178	1
markshare2	0.0	0.0	0.0	—	00:06	118	0	0.0	—	00:13	247	0
mas74	6.7	6.8	6.7	0.0	00:52	185	0	7.1	327.8	02:20	490	17
misc03	15.0	15.0	15.0	—	02:28	40	0	15.0	—	07:49	128	0
misc06	31.4	38.2	37.6	91.9	00:28	47	5	41.3	145.6	00:37	61	5
misc07	0.7	0.7	0.7	—	07:23	61	0	0.7	—	15:00	163	2

Table 2: Strength of 3-row infinity cuts

			Non-Lifted 3-Row					Lifted 3-Row				
Instance	G-GMI	G-EXACT	G-INF	REL (%)	T-TIME (m:s)	C-TIME (ms)	N-CUTS	G-INF	REL (%)	T-TIME (m:s)	C-TIME (ms)	N-CUTS
mitre	84.0	84.0	84.0	—	01:07	27	1	84.0	—	15:00	559	4
mkc	8.7	10.6	8.7	0.0	01:06	89	0	8.7	0.0	15:00	1336	1
mod008	21.6	21.6	21.6	—	00:05	164	0	21.6	—	00:27	779	0
mod010	100.0	100.0	100.0	—	06:30	84	0	100.0	—	15:00	503	0
mod011	33.0	35.1	33.0	0.0	02:50	34	0	33.0	0.0	04:20	52	0
modglob	17.3	33.6	17.5	1.4	00:15	19	5	17.5	1.5	00:24	30	5
nw04	66.1	66.1	66.1	—	15:00	802	0	66.1	—	15:09	13373	0
p0033	34.4	35.2	34.5	8.7	00:11	28	1	35.2	105.4	00:26	65	6
p0201	6.5	6.5	6.5	—	06:51	63	0	6.5	—	15:00	155	4
p0282	3.2	9.9	6.2	45.3	03:31	35	4	6.2	45.3	08:32	84	10
p0548	61.7	74.2	61.7	0.0	00:19	42	0	61.7	0.1	01:15	164	3
p2756	58.0	58.9	58.0	0.5	00:14	30	3	58.0	0.0	00:50	104	1
pk1	0.0	0.0	0.0	—	01:31	163	0	0.0	—	02:48	301	0
pp08a	54.9	66.7	68.1	112.4	00:05	18	37	68.1	112.3	00:07	22	36
pp08aCUTS	33.8	40.7	38.0	60.8	04:35	62	23	38.0	61.3	06:29	88	21
qiu	1.4	1.7	1.7	104.5	08:39	110	11	1.7	102.6	10:21	132	13
qnet1	11.7	16.8	13.8	41.0	15:00	121	13	14.7	59.9	14:59	250	13
qnet1_o	19.8	30.5	23.8	36.9	12:46	133	27	24.8	45.9	15:00	651	32
rentacar	19.8	20.3	19.8	0.0	00:02	21	0	19.8	0.0	00:03	23	0
rgn	12.0	12.0	12.0	—	00:15	45	0	16.2	—	00:29	88	7
rout	0.3	5.2	0.3	0.0	12:56	153	0	0.3	0.0	15:00	460	3
set1ch	29.4	72.2	44.0	34.1	00:08	19	87	44.0	34.1	00:14	33	87
seymour	6.6	6.6	6.6	—	14:59	108	0	6.6	—	14:59	209	0
stein27	0.0	0.0	0.0	—	01:27	15	0	0.0	—	02:08	22	0
stein45	0.0	0.0	0.0	—	07:38	46	0	0.0	—	13:07	79	0
swath	8.7	10.7	8.7	0.1	13:41	124	7	10.6	93.3	14:59	837	5
vpm1	29.9	33.6	30.9	25.4	00:04	27	8	33.9	105.8	00:09	58	8
vpm2	13.9	23.5	15.7	19.5	00:14	33	8	19.0	53.6	00:22	54	26
	29.0	36.1	32.2	44.8	03:52	263.8	7.8	32.7	52.2	06:06	687.2	10.0

Table 2: Strength of 3-row infinity cuts (continued)

model. However, we provide results with lifting enabled, because the trivial lifting is relatively fast to compute, and would be a natural next step in practice.

For the set of 56 instances processed, the average gap closed by GMI cuts, by non-lifted infinity cuts, by lifted infinity cuts, and by the exact separator was, respectively, 29.0%, 31.3%, 31.8% and 33.9%. Therefore, in terms of gap closure, non-lifted infinity cuts provided, on average, 47.8% of the benefits of adding the entire class of continuous 2-row intersection cuts, while lifted infinity cuts provided, on average 57.8%.

The total CPU running time necessary to generate the non-lifted infinity cuts was on average very low, at 20 seconds, while the median was below 1 second. After lifting, the average running time increased to 2 minutes and 27 seconds. In comparison, the average running time for the exact separator was 1 hour and 26 minutes, with many instances reaching the time limit of 4 hours. Because of this time limit, the infinity cut generator was able to obtain a better gap closure than the exact separator in several instances. The infinity cut generator added, on average, 5.6 non-lifted cuts and 7.4 lifted cuts per instance. In comparison, the exact separator added, on average 42 cuts per instance.

For 21 out of the 56 instances processed, the exact separator was not able to improve upon the gap closed by the GMI cuts, while the non-lifted infinity cuts were not able to improve upon the gap for 32 instances. We note however that, for these instances, the total running time to generate these cuts was only 22 seconds and the number of cuts added was only 0.3, on average. Therefore, even for instances where non-lifted infinity cuts were not helpful, they would also probably not significantly affect the total running time of the branch-and-bound algorithm. Lifted infinity cuts were not able to improve the gap for 25 instances. The total running time for these instances was 3 minutes and 26 seconds, while the number of added cuts was 0.5, on average.

The experiments show that, while generating each cut individually is not an expensive procedure, most of the time is spent trying to find a good candidate set of tableau rows to use. This is evidenced if one inspects the total number of attempted cuts ($\frac{T-TIME}{C-TIME}$) and the actual number of cuts generated (N-CUTS).

We also ran computational experiments for lifted and non-lifted infinity cuts from 3 rows of the simplex tableau. Table 2 shows these results. Here, we compare against the exact separator for continuous 3-row intersection cuts. For the set of 56 instances processed, the average gap closed by GMI, by non-lifted infinity cuts, by lifted infinity cuts, and by the exact separator was, respectively, 29.0%, 32.2%, 32.7% and 36.1%. Non-lifted infinity cuts provided, on average, 44.9% of the benefits of the exact separator, while lifted infinity cuts provided 52.2%. This suggests that the relative strength of infinity cuts is preserved for relaxations with more rows.

The total CPU running time per instance was 3 minutes and 52 seconds for the non-lifted infinity cuts, and 6 minutes and 6 seconds for the lifted ones, on average. In comparison, the exact separator spent 2 hours and 27 minutes, on average. The infinity cut generator added,

on average, 7.8 non-lifted cuts and 10.0 lifted cuts — numbers considerably lower than the 103 cuts added on average by the exact separator.

5 Conclusion

In this paper, we introduced a new subset of multi-row intersection cuts based on the infinity norm. This subset is very small, which makes it attractive computationally, and its cuts are minimal, meaning that they are individually undominated. Unlike the other classes of multi-row intersection cuts in the literature, our cuts work for relaxations with arbitrary numbers of rows and they take into account the columns of the simplex tableau. We developed a geometric algorithm to generate them and ran computational experiments to evaluate their impact when used as cutting planes for solving real-world instances. We concluded that this subset of valid inequalities yields, in terms of gap closure, around 50% of the benefits of using all the valid inequalities for the continuous multi-row relaxation, but at a small fraction of the computational cost, and with a significantly smaller number of cuts.

We highlight two main directions for future work. First, we observed that our cut generator does not necessarily produce maximal lattice-free sets. While this has no effect on the coefficients of the continuous non-basic variables, it does negatively impact the coefficients for the integral non-basic variables (obtained through lifting). It would be interesting to have a practical algorithm for converting non-maximal lattice-free sets into maximal ones. Secondly, in Section 4, we described several heuristics for selecting a subset of rows that would be used to generate infinity cuts. This was not the main focus of this work, but it was clearly what made the running time of the code increase significantly. Thus, better ways to pick such subsets of rows are needed.

References

[1] Kent Andersen, Quentin Louveaux, Robert Weismantel, and Laurence Wolsey. Inequalities from two rows of a simplex tableau. In Matteo Fischetti and David Williamson, editors, *Integer Programming and Combinatorial Optimization*, volume 4513 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin / Heidelberg, 2007. doi:http://dx.doi.org/10.1007/978-3-540-72792-7_1.

[2] Egon Balas. Intersection cuts – a new type of cutting planes for integer programming. *Operations Research*, 1(19):19–39, 1971.

[3] Egon Balas and Anureet Saxena. Optimizing over the split closure. *Mathematical Programming*, 113(2):219–240, 2008. doi:10.1007/s10107-006-0049-5.

[4] Amitabh Basu, Pierre Bonami, Gérard Cornuéjols, and François Margot. Experiments with two-row cuts from degenerate tableaux. *INFORMS Journal on Computing*, 23:578–590, 2011.

[5] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer programming*, volume 271. Springer, 2014.

[6] Santanu S. Dey, Andrea Lodi, Andrea Tramontani, and Laurence A. Wolsey. Experiments with two row tableau cuts. In Friedrich Eisenbrand and F. Bruce Shepherd, editors, *Integer Programming and Combinatorial Optimization, 14th International Conference, IPCO 2010, Lausanne, Switzerland, June 9-11, 2010. Proceedings*, volume 6080 of *Lecture Notes in Computer Science*, pages 424–437. Springer, 2010.

[7] Santanu S. Dey, Andrea Lodi, Andrea Tramontani, and Laurence A. Wolsey. On the practical strength of two-row tableau cuts. *INFORMS Journal on Computing*, 26(2):222–237, 2014. doi:10.1287/ijoc.2013.0559.

[8] Daniel G. Espinoza. Computing with multi-row Gomory cuts. *Operations Research Letters*, 38(2):115 – 120, 2010. doi:10.1016/j.orl.2009.10.016.

[9] Ralph E. Gomory. Some polyhedra related to combinatorial problems. *Linear Algebra and its Applications*, 2(4):451–558, 1969.

[10] Quentin Louveaux and Laurent Poirrier. An algorithm for the separation of two-row cuts. *Mathematical Programming*, 143(1-2):111–146, 2014. doi:10.1007/s10107-012-0597-9.

[11] Quentin Louveaux, Laurent Poirrier, and Domenico Salvagnin. The strength of multi-row models. *Mathematical Programming Computation*, 7(2):113–148, 2015.

[12] François Margot. Testing cut generators for mixed-integer linear programming. *Mathematical Programming Computation*, 1(1):69–95, 2009. doi:10.1007/s12532-009-0003-7.

[13] Felipe Serrano Musalem. Some experiments on separation with multi-row cuts. Master's thesis, Universidad de Chile, 2012.

[14] Álinson S. Xavier, Laurent Poirrier, and Ricardo Fukasawa. Multi-Row Intersection Cuts based on the Infinity Norm: Source Code, July 2019. doi:10.5281/zenodo.3275892.

[15] Emre Yamangil. *Valid Inequalities for Mixed-Integer Linear Programming Problems*. PhD thesis, Rutgers, The State University of New Jersey, 2015.