Sparsity of Lift-and-Project Cutting Planes

Diplomarbeit

von

Matthias Walter

angefertigt am Institut für Mathematische Optimierung
der Otto-von-Guericke Universität Magdeburg

unter Anleitung von
Herrn Professor Dr. Volker Kaibel

Magdeburg, September 2011

Preface

During my internship at the IBM T.J. Watson Research Center in fall 2010 I studied recent papers that may be suitable for an implementation for CPLEX, one of the state-of-the-art optimization suites. The main topic was cutting planes because an implementation was relatively easy to create without requiring specific (and secret) internal information. The first article which I worked on was by Andersen and Weismantel and I knew it from Kent's talk at the Integer Programming and Combinatorial Optimization Conference 2010 in Lausanne. A very fascinating property of their cut generation method was that they had complete control over zero coefficients. One year later I must admit that this is not quite true.

Eventually, Andrea Lodi convinced me to implement the lift-and-project method in a way which Bonami discovered in 2010. In the remainder of the internship I became familiar with the method and got a feeling about decent split selection strategies and solving time. The results were very typical: A significantly smaller branch & bound tree, but longer computation times. One reason was that some of the generated cuts were relatively dense.

These two highlights inspired me to write my thesis about sparsity. It satisfied my desire for a topic which requires programming and was also promising because of its suspected importance (as a reason for poor performance of my cuts) as well as missing previous work in literature.

I owe many thanks to my internship advisor Laci Ladanyi who guided me through all the pitfalls of coding with CPLEX as well as Tobias Achterberg, Andrea Lodi, Andrea Tramontani, and Roland Wunderling for their helpful comments. I also thank Mary Fenelon for organizing my intership.

Just as much as the above people I want to thank Prof. Dr. Volker Kaibel. As my main advisor in Magdeburg he did not only make the internship possible. He also trusted me to select a good topic on my own. Although the affinity to his favorite subjects is

relatively small our discussions about my work were very fruitful.

Working on the topic was sometimes frustrating and challenging on the one hand, but often very satisfying on the other hand. I'm also very thankful to my girlfriend Sarah for suffering from my permanent desire to work on my computer. The same goes for Stefanie Kabelitz who had to listen to the mathematical details a couple of times. Finally, I want to thank my parents Gabi and Karl-Heinz for enabling me to study and in addition to them, Julia Gebhardt, Georg Naumann, Bernhard Spitz, and Tobias Walter for helpful comments on earlier versions of this thesis.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This thesis is about a numerical property of a certain cutting plane method in mixed-integer (linear) programming. A *linear program* (LP) is a problem for which we want to optimize a linear objective function over an intersection of (finitely many) linear inequalities and equations. The set of feasible points is a *polyhedron*, i.e., the intersection of finitely many halfspaces. Every $\geq$-inequality can be scaled by -1 and every equation can be modeled by two inequalities. If not stated differently, we can assume without loss of generality that polyhedra are given in the form

$$P = P^{\leq}(A, b) := \{x \in \mathbb{R}^n : Ax \leq b\}$$

Here, $\mathbb{R}$ denotes the real numbers, A is a real $m \times n$ matrix over $\mathbb{R}$, $b \in \mathbb{R}^m$ and $\leq$ has to be read component-wise. In other contexts, one may also demand nonnegativity constraints, i.e., $x \in \mathbb{R}^n_+$. For LPs we always assume a minimization problem of the following type.

$$\min c^\intercal x \text{ such that } x \in P$$

Assuming only rational input there are several algorithms to solve LPs. The ellipsoid method runs in polynomial time, while for the de-facto standard method, the famous simplex algorithm, no polynomial time version is known. A third kind, so-called barrier methods can also be made to run in polynomial time. Efficient in practice are only the last two. For further reading about polyhedra we recommend [38] and for complexity theory [29].

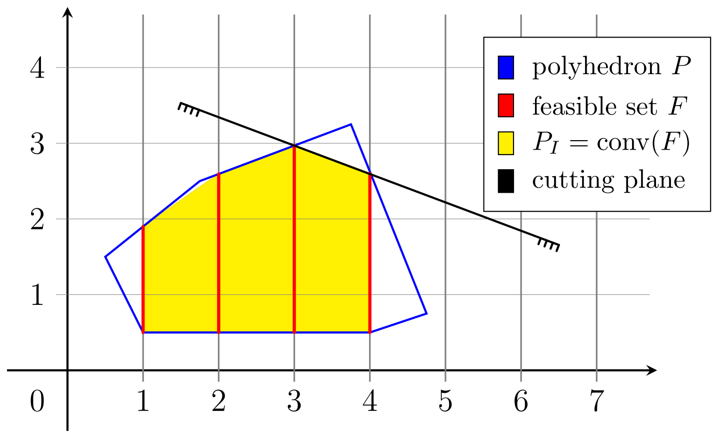


Figure 1.1: A 2-dimensional mixed-integer program (MIP).

A *mixed-integer program (MIP)* is an LP with integrality restrictions imposed on a subset I of $[n] := \{1, \ldots, n\}$. We call a feasible point x *integral* if $x_i \in \mathbb{Z}$ for $i \in I$, although this notation is not quite right. A very important observation is that the convex hull of all feasible points, denoted by P_I, is a polyhedron again (see Figure 1.1). P_I is (usually strictly) contained in P. This class of problems does not belong to the class of convex optimization problems and is $\mathcal{NP}$-hard in general. Nevertheless, there are efficient methods to solve problems of decent size in a couple of minutes. The most important concept is that of a *linear relaxation* which means that we drop the integrality constraints. This relaxation can be strengthened by reducing the set of feasible points without loosing an integer point. A widely used way of strengthening is via *cutting planes*. They are additional inequalities which cut off points which are feasible for the LP but infeasible for the MIP. An important concept is that of a *separation problem*. It means that given a MIP and some LP-feasible but nonintegral point, we have to find a valid cutting plane which cuts off this point or we have to determine that no such cut exists (from a given class of cuts).

The third concept is that of *sparsity*. An LP row (an inequality or equation) is said to be sparse if her coefficient vector a has not too many nonzero entries. Mathematically speaking, the support $\text{supp}(a) := \{i \in [n] : a_i \neq 0\}$ shall contain only few indices. The importance of sparsity comes from practice. LP solvers (which are invoked many times while solving MIPs) simply exploit sparsity by “looking” at nonzero entries only. Hence, a *dense* cutting plane may strengthen the relaxation, but slow down the future solving process, making its use unattractive.

As the reader may have guessed from the title we will focus on lift-and-project cutting planes, although other classes are considered later too. Lift-and-project cuts were introduced by Egon Balas, Sebastián Ceria, and Gérard Cornuéjols in their seminal paper in 1993. Before, some basic concepts were already known but many experts in this field did not believe in the practical usefulness of cutting planes. Although Gomory Mixed-Integer (GMI) cuts were more successful for practical implementations, lift-and-project was never forgotten and experienced a boom in 2003. In that year, Egon Balas had a Ph.D. student, namely Michael Perregaard, who was working on lift-and-project. They were able to show a precise correspondence between lift-and-project and other classes of cuts. This led to much insight and some very attractive features of lift-and-project. For the history-interested reader we want to mention Sebastián Ceria's nice article about the history of lift-and-project [19].

Outline. Our main work is separated into four chapters. In Chapter 2 we start with the lift-and-project method itself. We will go along the historic path until we have the so-called Cut Generating LP (CGLP) available. Then we thoroughly investigate the normalization constraints which are necessary to use the CGLP for cut separation. After deriving some basic properties we will present different formulations which can be found in literature and we will show how the obtained cuts can be strengthened further. Chapter 3 connects lift-and-project with other classes of cuts and the well-known relaxation hierarchies. It was put separate because of its minor importance for our sparsity investigations. We present some important techniques which are helpful for implementations in Chapter 4. Among them are ways to make certain features like cut strengthening work or to reduce the computational effort. In Chapter 5 we eventually focus on the sparsity structure. Our analysis is based on several experiments which have the purpose to:

- verify a correlation between LP solving time and sparsity,
- measure the sparsity of lift-and-project cutting planes, taking different parameters into account.
- point out potential improvements which might help to make cuts sparser or show limitations.

The experiments are described and the results are displayed and interpreted.

Notations. We now introduce some necessary mathematical concepts. For any nonnegative number $n \in \mathbb{Z}_+$ we denote by $[n]$ the set $\{1, \ldots, n\}$ and by $[n]_0$ the augmented set $[n] \cup \{0\}$. As we do not define the groundset of all matrices and vectors everywhere we assume them to be of appropriate sizes. The same holds for $\mathbb{O}$ (the zero vector) and $\mathbb{1}$ (the vector consisting of only 1's). For an $m \times n$ matrix A we denote by $A_{i,j}$ the entry in the i-th row and j-th column. We furthermore allow the placeholder $*$ for i or j, meaning the complete row or column. For example, $A_{i,*}$ is the i-th row (as a row vector). By $A^{\intercal}$ we denote transposition of matrices and vectors.

We will use L_p norms, denoted by $||.||_p$, for $p \in \{1, 2, \infty\}$. For any $x \in \mathbb{R}$, $|x|$, $\lfloor x \rfloor$, and $\lceil x \rceil$, denote the absolute value, x rounded down to the next integer, and x rounded down, respectively.

By $\mathrm{conv}(S) := \{\lambda s_1 + (1 - \lambda) s_2 : s_1, s_2 \in S \ \wedge \ \lambda \in [0, 1]\}$ we denote the convex hull of some set S and by $\mathrm{cone}(S) := \{\lambda s : s \in S \ \wedge \ \lambda \geq 0\}$ the cone generated by S.

Chapter 2

The Lift-and-Project Method

In this chapter we focus on the lift-and-project method itself. The first two sections are concerned with the derivation of the method. First we describe the original lift-and-project approach followed by another description using disjunctive programming. Then we specify the set of valid inequalities more precisely. As the latter set is a cone it must be truncated by a so-called normalization constraint. There are several of these and we present them along with their geometrical meaning in the dual space. Furthermore, we propose another one which is supposed to yield sparser cutting planes. This is our first own contribution in this work. In literature certain formulations of the cone are used and we link them together in the fourth section. We finish this chapter by presenting the standard way of strengthening the cutting planes.

2.1 Derivation of the Method

First we derive lift-and-project via the original idea of lifting the LP relaxation in a higher dimensional space where one is able to tighten it conveniently. After this step one can either work in this space or project it back. Two such approaches for 0-1-mixed-integer programming (a MIP where all integer variables are binary) have been suggested before lift-and-project came up.

In the first approach Sherali and Adams propose to multiply $Ax \leq b$ by *several* 0-1 variables x_j or their complements $(1 - x_j)$ and project back to the original space (see [39], [40]). The second is by Lovász and Schrijver. For every $j \in I$ they consider all the inequalities $x_j(Ax) \leq x_j b$, $(1 - x_j)(Ax) \leq (1 - x_j)b$ and collect them. After linearizing the number of 0-1-variables is squared and the number of constraints also increased very much. Their main result in [33] is that applying this process n times yields P_I.

The lift-and-project method is equivalent to the Lovász-Schrijver approach except that it is only applied to a single variable x_j. Hence, the number of variables in the larger space is only twice the original number. It was introduced in [14] for 0-1-mixed-integer problems. We will work on our polyhedron $P := P^{\leq}(A, b)$ with $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and a set $I \subseteq [n]$ of binary variables. We assume that the (relaxed binary) constraints $0 \leq x_i \leq 1$ $(i \in I)$ are contained in $Ax \leq b$.

Then the following procedure strengthens the LP relaxation, i.e., produces a polyhedron P_k with $P_I \subseteq P_k \subseteq P$.

Algorithm 2.1 The Historical Lift-and-Project Method

1. Select an index $k \in I$ among the binary variables of $P^{\leq}(A, b)$.
2. Replace $(Ax - b) \leq 0$ by the following (nonlinear) system.
$$\begin{array}{rcl} x_k(Ax - b) & \leq & 0 \\ (1 - x_k)(Ax - b) & \leq & 0 \end{array} \tag{2.1}$$
3. Linearize (2.1) by substituting y_i for $x_k \cdot x_i$ for all $i \in [n]$ with $i \neq k$ and x_k for x_k^2. Call the resulting polyhedron $M^{(k)} = M^{(k)}(P)$.
4. Project $M^{(k)}(P)$ onto the x variables and call the resulting polyhedron $P^{(k)}$.

Our first observation is that this method does not remove feasible points. In fact, x_k and $(1 - x_k)$ are both nonnegative and thus (2.1) is still a relaxation of P_I. Obviously, substitution of y_i does not change any structure. Hence, the only part where some (fractional) points may become infeasible is substitution of x_k^2 by x_k. The latter is valid for binary values which implies $P_I \subseteq P^{(k)}$. The other important fact is $P^{(k)} \subseteq P$ and is proved formally in the next section.

We will provide more details about the polyhedral properties of the constructed polyhedron $P^{(k)}$ in Chapter 3. There we will show that the same property holds which is was also proved for the Lovász-Schrijver construction. We investigate the strength of the procedure (applied to all $k \in I$) compared to other cutting plane closures.

2.2 Disjunctive Programming

We now present another characterization of $P^{(k)}$ from the first section. It also appeared in [14] in Theorem 2.1. In the remainder of this section we show how to get a compact formulation of the set of valid inequalities for $P^{(k)}$. The latter part is known as disjunctive programming and was introduced by Balas in [8],[9].

We start by stating the announced characterization of $P^{(k)}$ and repeating the proof by Balas et al..

Theorem 2.1 (Balas, Ceria, and Cornuéjols, 1993). *Let $P^{(k)}$ be as in Algorithm 2.1, and let $P_0 := P \cap \{x_k = 0\}$, $P_1 := P \cap \{x_k = 1\}$. Then $P^{(k)} = \operatorname{conv}(P_0 \cup P_1)$.*

Proof. For the "$\supseteq$" part let $\overline{x}$ be an arbitrary point in $P \cap \{x_k = 0, 1\}$. Define $\overline{y}_i := \overline{x}_i \cdot \overline{x}_k$ for $i \neq k$. Then $(\overline{x}, \overline{y})$ is valid for $M^{(k)}(P)$ since $\overline{x}_k^2 = \overline{x}_k$.

To show "$\subseteq$" we distinguish three cases:

Case 1: $P_0 = \emptyset$. As $P^{(k)}$ is closed the inequality $x_k > \varepsilon$ is valid for $P^{(k)}$ for some $\varepsilon > 0$. This implies that $(1 - x_k)(x_k - \varepsilon) \geq 0$ is satisfied by any x satisfying (2.1). Replacing $x_k^2 = x_k$ it follows that $x_k \geq 1$ is valid for $P^{(k)}$. This together with $P^{(k)} \subseteq P$ implies $P^{(k)} \subseteq P_1 = \operatorname{conv}(\emptyset \cup P_1)$.

Case 2: $P_1 = \emptyset$. In an analogous way we can prove the validity of $x_k \leq 0$ for $P^{(k)}$. This implies $P^{(k)} \subseteq \operatorname{conv}(P_0 \cup \emptyset)$.

Case 3: $P_0 \neq \emptyset$ and $P_1 \neq \emptyset$. Let $\alpha^\intercal x \leq \beta$ be a valid inequality for $\operatorname{conv}(P_0 \cup P_1)$. Since $\alpha^\intercal x \leq \beta$ is valid for P_0 there exists $\lambda \geq 0$ such that $\alpha^\intercal + \lambda x_k \leq \beta$ is valid for P (lifting of inequalities). Similarly, there exists $\mu \geq 0$ such that $\alpha^\intercal x + \mu(1 - x_k) \leq \beta$ is valid for P. Now any x satisfying (2.1) must also satisfy $(1-x_k)(\alpha^\intercal x+\lambda x_k-\beta) \leq 0$ and $x_k(\alpha^\intercal x+\mu(1-x_k)-\beta) \leq 0$. Adding both inequalities yields $\alpha^\intercal x+(\lambda+\mu)(x_k-x_k^2) \leq \beta$. After setting $x_k^2 = x_k$ the validity of $\alpha^\intercal x \leq \beta$ for $M^{(k)}(P)$ and for $P^{(k)}$ is clear.

Both directions together finish the proof. $\square$

For the remainder of this thesis we can now forget about the nonlinear construction in Algorithm 2.1. An example is depicted in Figure 2.1. Furthermore, we can work in the more general context where a polyhedron $P^{\leq}(A, b)$ (not necessarily with $[0, 1]$-variables) is given. Instead of an *elementary split disjunction* $(x_k \leq 0) \vee (x_k \geq 1)$ we now allow arbitrary *split disjunctions* $(\pi^\intercal x \leq \pi_0) \vee (\pi^\intercal x \geq \pi_0 + 1)$ for $\pi \in \mathbb{Z}^n$ where $\pi_j = 0$ for all $j \notin I$ and any $\pi_0 \in \mathbb{Z}$. This results in P being split into $P_0 := P \cap \{\pi^\intercal x \leq \pi_0\}$ and $P_1 := P \cap \{\pi^\intercal x \geq \pi_0 + 1\}$. Then our new relaxation is $\operatorname{conv}(P_0 \cup P_1)$.

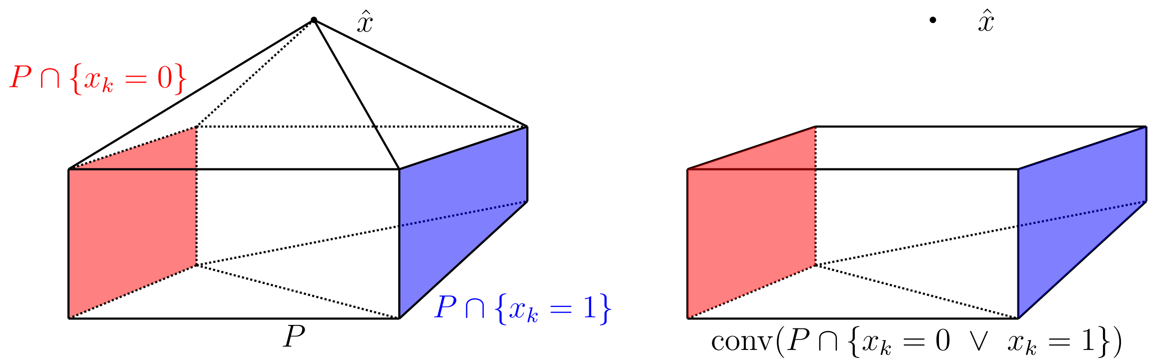


Figure 2.1: The set $\operatorname{conv}(P_0 \cup P_1)$ for a disjunction.

Disjunctive programming in our context is the optimization over unions of polyhedra. In their seminal paper Balas, Ceria, and Cornuéjols referred to the following lifting theorem from [8], specialized to the union of two polyhedra.

Theorem 2.2 (Balas, 1974)**.** *Let $P_i = P^{\leq}(C^{(i)}, d^{(i)})$ for $i = 1, 2$ be two nonempty polyhedra in $\mathbb{R}^n$. Then $\operatorname{conv}(P_0 \cup P_1)$ is the set of $x \in \mathbb{R}^n$ for which there exist vectors $x^{(1)}, x^{(2)} \in \mathbb{R}^n$ and $y^{(1)}, y^{(2)} \in \mathbb{R}_+$ such that*

$$\begin{array}{rcl} C^{(1)}x^{(1)} & \leq & d^{(1)}y^{(1)} \\ C^{(2)}x^{(2)} & \leq & d^{(2)}y^{(2)} \\ x^{(1)} + x^{(2)} & = & x \\ y^{(1)} + y^{(2)} & = & 1 \end{array} \tag{2.2}$$

Note, however, that we assumed $P_0, P_1 \neq \emptyset$ which does not always hold. Instead of applying more techniques in order to verify the correctness of the formulation in this case we will now derive a compact description of the projection onto the x variables. Later we will analyze this projection thoroughly and also prove that it yields valid cutting planes even if P_0 or P_1 are empty.

For this we first apply Theorem 2.2 to our split polyhedron.

Corollary 2.3. *Let $P = P^{\leq}(A, b)$ be a polyhedron and $(\pi^\intercal x \leq \pi_0) \vee (\pi^\intercal x \geq \pi_0 + 1)$ be a split disjunction. Then $\operatorname{conv}(P_0 \cup P_1)$ equals the projection of*

$$\begin{array}{rcl} Ax^{(1)} & \leq & by^{(1)} \\ \pi^\intercal x^{(1)} & \leq & \pi_0 y^{(1)} \\ Ax^{(2)} & \leq & by^{(2)} \\ -\pi^\intercal x^{(2)} & \leq & -(\pi_0+1)y^{(2)} \\ x^{(1)} + x^{(2)} & = & x \\ y^{(1)} + y^{(2)} & = & 1 \\ y^{(1)}, y^{(2)} & \geq & 0 \end{array} \quad \Leftrightarrow \quad \begin{array}{rcl} Ax^{(1)} - by^{(1)} & \leq & 0 \\ \pi^\intercal x^{(1)} - \pi_0 y^{(1)} & \leq & 0 \\ Ax^{(2)} - by^{(2)} & \leq & 0 \\ -\pi^\intercal x^{(2)} + (\pi_0+1)y^{(2)} & \leq & 0 \\ x - x^{(1)} - x^{(2)} & = & 0 \\ y^{(1)} + y^{(2)} & = & 1 \\ -y^{(1)} & \leq & 0 \\ -y^{(2)} & \leq & 0 \end{array} \tag{2.3}$$

on the x variables.

Our next goal is to describe the projection. For that we need some standard machinery. We start with the Farkas' Lemmata which give characterizations whether a certain polyhedron is empty or not.

Lemma 2.4 (Farkas' Lemmata)**.** *The following statements hold for $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$:*

(a) The system $Ax \leq b$ has a solution $x \in \mathbb{R}^n$ if and only if for every $y \in \mathbb{R}^m_+$ with $y^\intercal A = 0$ the inequality $y^\intercal b \geq 0$ holds.

(b) The system $Ax \leq b$ has a solution $x \in \mathbb{R}^n_+$ if and only if for every $y \in \mathbb{R}^m_+$ with $y^\intercal A \geq 0$ the inequality $y^\intercal b \geq 0$ holds.

(c) Either the system $Ax \leq b$ has a solution $x \in \mathbb{R}^n$ or there exists $y \in \mathbb{R}^m_+$ with $y^\intercal A = 0$ and $y^\intercal b = -1$.

Proofs can be found in [38] where (a) corresponds to Corollary 7.1e, (b) to Corollary 7.1f, and (c) is just a reformulation of (a). While (c) will be helpful later we now use (a) for the following projection theorem. Part (b) is not used directly, but illustrates the result for the variables $y^{(1)}, y^{(2)}$ in (2.3).

Theorem 2.5. *Let $P := \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^p : Gx + Hy \leq s\}$. The projection onto the x-space is given by $\{x \in \mathbb{R}^n : \lambda^\intercal Gx \leq \lambda^\intercal s$ for all $\lambda \in \mathbb{R}^m_+$ with $\lambda^\intercal H = 0\}$*

Proof. We write the projection as $\{x \in \mathbb{R}^n : \exists y \in \mathbb{R}^m$ with $Gx + Hy \leq s\}$. Let $x \in \mathbb{R}^n$. By Lemma 2.4 (a), $Hy \leq s - Gx$ has a solution $y \in \mathbb{R}^m$ if and only if $\lambda^\intercal(s - Gx) \geq 0$ for every $\lambda \in \mathbb{R}^m_+$ with $\lambda^\intercal H = 0$. □

The final step is to use Theorem 2.5 in order to project the polyhedron given by (2.3) on the x variables. The row multipliers are $(w^\intercal, w_0, v^\intercal, v_0, \alpha^\intercal, \beta, \lambda^1, \lambda^2)$. The valid inequalities are

$$\begin{pmatrix} w \\ w_0 \\ v \\ v_0 \\ \alpha \\ \beta \\ \lambda^1 \\ \lambda^2 \end{pmatrix}^\intercal \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ I_n \\ 0 \\ 0 \\ 0 \end{pmatrix} x \leq \begin{pmatrix} w \\ w_0 \\ v \\ v_0 \\ \alpha \\ \beta \\ \lambda^1 \\ \lambda^2 \end{pmatrix}^\intercal \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad \Leftrightarrow \quad \alpha^\intercal x \leq \beta \tag{2.4}$$

where $w, v \in \mathbb{R}^m_+$, $w_0, v_0, \lambda^1, \lambda^2 \in \mathbb{R}_+$ and $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}$ are unconstrained as they are multipliers for equations. The constraints for the multipliers are

$$\begin{pmatrix} w \\ w_0 \\ v \\ v_0 \\ \alpha \\ \beta \\ \lambda^1 \\ \lambda^2 \end{pmatrix}^\intercal \begin{pmatrix} A & -b & & \\ \pi^\intercal & -\pi_0 & & \\ & & A & -b \\ & & -\pi^\intercal & (\pi_0 + 1) \\ -I_n & & -I_n & \\ & 1 & & 1 \\ & -1 & & \\ & & & -1 \end{pmatrix} = 0. \tag{2.5}$$

After simplification and setting up the goal to maximize the violation of a given point $\hat{x}$

this yields the *cut generating linear program*:

$$\begin{array}{lrcl} \min & \beta & - & \alpha^\intercal \hat{x} \\ \text{s.t.} & \alpha^\intercal & = & w^\intercal A + w_0 \pi^\intercal \\ & \alpha^\intercal & = & v^\intercal A - v_0 \pi^\intercal \\ & \beta & \geq & w^\intercal b + w_0 \pi_0 \\ & \beta & \geq & v^\intercal b - v_0(\pi_0 + 1) \\ & w, v & \in & \mathbb{R}^m_+ \\ & w_0, v_0 & \in & \mathbb{R}_+ \end{array} \tag{CGLP}$$

The associated polyhedron is a cone $\mathcal{C}^=_\geq$ and it is unbounded once a nontrivial cut ($\alpha \neq \mathbb{O}$) exists. This makes sense since any inequality $\alpha^\intercal x \leq \beta$ can be scaled without changing its geometry. In order to optimize over this set we need to truncate it. This amounts to adding a normalization constraint and is discussed later.

In the above description we assumed nonempty P_0, P_1 in order to make the convexification work properly. We now consider the case where this does not hold and answer the question whether the constructed inequalities are still valid. The following lemma characterizes the region that is cut off.

Lemma 2.6. *Let $P = P^\leq(A, b)$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ be a given polyhedron, $\pi_0 \in \mathbb{Z}$, and $\pi \in \mathbb{Z}^n$ be a split. Let furthermore $\hat{x}$ be in the polyhedron P but cut off by some $\alpha^\intercal x \leq \beta$ from $\mathcal{C}^=_\geq$.*

Then $\pi^\intercal \hat{x} \in (\pi_0, \pi_0 + 1)$ and in the corresponding solution to (CGLP), w_0 and v_0 are positive.

Proof. Let $(w, w_0, v, v_0, \alpha, \beta)$ be a solution to the cut constraints. From $\alpha^\intercal x > \beta$ we derive

$$\begin{aligned} (w^\intercal A + w_0 \pi^\intercal)\, \hat{x} > \beta &\geq w^\intercal b + w_0 \pi_0 \\ (v^\intercal A - v_0 \pi^\intercal)\, \hat{x} > \beta &\geq v^\intercal b - v_0(\pi_0 + 1). \end{aligned}$$

As $\hat{x} \in P$ we have $A\hat{x} \leq b$, i.e., $-w^\intercal A\hat{x} \geq -w^\intercal b$ and $-v^\intercal A\hat{x} \geq -v^\intercal b$. Adding both inequalities to the above we observe

$$w_0 \pi^\intercal \hat{x} > w_0 \pi_0 \text{ and } - v_0 \pi^\intercal \hat{x} > -v_0(\pi_0 + 1)$$

and conclude that $w_0, v_0 > 0$. Dividing by both yields $\pi_0 < \pi^\intercal \hat{x} < \pi_0 + 1$. □

As the constraints $w_0, v_0 \geq 0$ were not used in the proof we can conclude:

Remark 2.7. *Lemma 2.6 also holds when the cut $\alpha^\intercal x \leq \beta$ is a solution to $\mathcal{C}_{\geq}^{=}$ where the nonnegativity constraints for w_0 and v_0 have been removed.*

Lemma 2.6 also proves the validity of the lift-and-project inequalities regardless of whether P_0 or P_1 may be empty sets.

Corollary 2.8. *The lift-and-project method only removes points in the interior of the split from the given polyhedron. Therefore it never cuts off an integer point.*

It is instructive to consider the case of $P_0 = \emptyset$ explicitly. Here, the Farkas Lemma (Lemma 2.4 (c)) provides us with nonnegative multipliers $\lambda \in \mathbb{R}^m_+$, $\mu \in \mathbb{R}_+$ with $\lambda^\intercal A + \mu\pi^\intercal = \mathbb{O}$ and $\lambda^\intercal b + \mu\pi_0 = -1$. That is, the inequality $\mathbb{O}x \leq -1$ can be derived as a valid inequality for P_0. We observe that $(\lambda^\intercal, \mu - 1, \mathbb{O}, 1, -\pi_0 - 1, -\pi^\intercal)$ is a feasible solution to (CGLP) showing that $\pi^\intercal x \geq \pi_0 + 1$ is a lift-and-project inequality:

$$\begin{aligned}
\alpha^\intercal &= w^\intercal A + w_0\pi^\intercal = \lambda^\intercal A + (\mu - 1)\pi^\intercal = -\mu\pi^\intercal + (\mu - 1)\pi^\intercal = -\pi^\intercal \\
\alpha^\intercal &= v^\intercal A - v_0\pi^\intercal = \mathbb{O} - \pi^\intercal = -\pi^\intercal \\
\beta &\geq w^\intercal b + w_0\pi_0 = \lambda^\intercal b + (\mu - 1)\pi_0 = -1 - \mu\pi_0 + \mu\pi_0 - \pi_0 = -\pi_0 - 1 \\
\beta &\geq v^\intercal b - v_0(\pi_0 + 1) = -\pi_0 - 1
\end{aligned}$$

It is easy to see that $\pi^\intercal x \leq \pi_0$ is a lift-and-project inequality if $P_1 = \emptyset$ by symmetric arguments. The resulting cut is depicted in Figure 2.2.

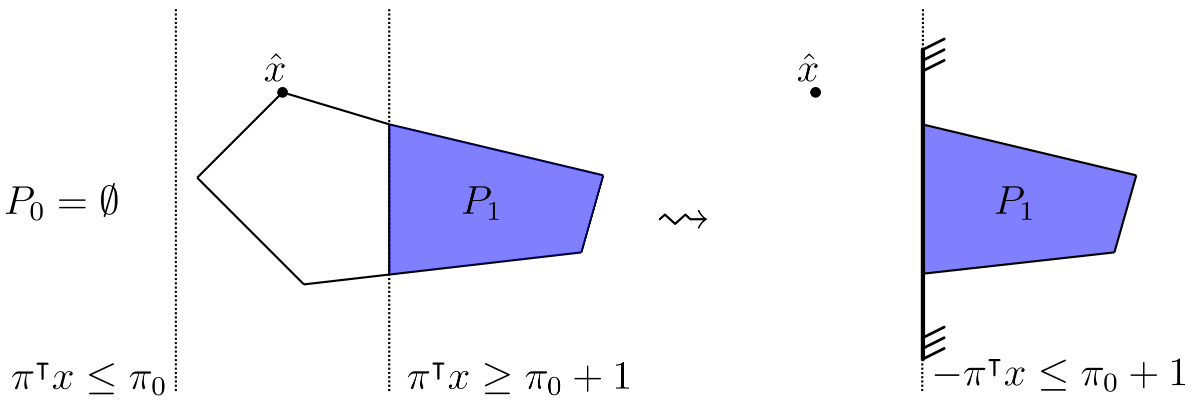

$\hat{x}$
$P_0 = \emptyset$
P_1
$\rightsquigarrow$
$\hat{x}$
P_1
$\pi^\intercal x \leq \pi_0$
$\pi^\intercal x \geq \pi_0 + 1$
$-\pi^\intercal x \leq \pi_0 + 1$

Figure 2.2: A lift-and-project inequality if $P_0 = \emptyset$.

2.3 Normalization Constraints

In order to truncate the feasible region of the CGLP one must add a so-called *normalization constraint*. The choice of a good normalization is very important because there is a large number of possible lift-and-project cutting planes (see Figure 2.3). We now present the most important ones that are studied in literature. We also suggest another one that (in theory) tries to generate sparser cuts.

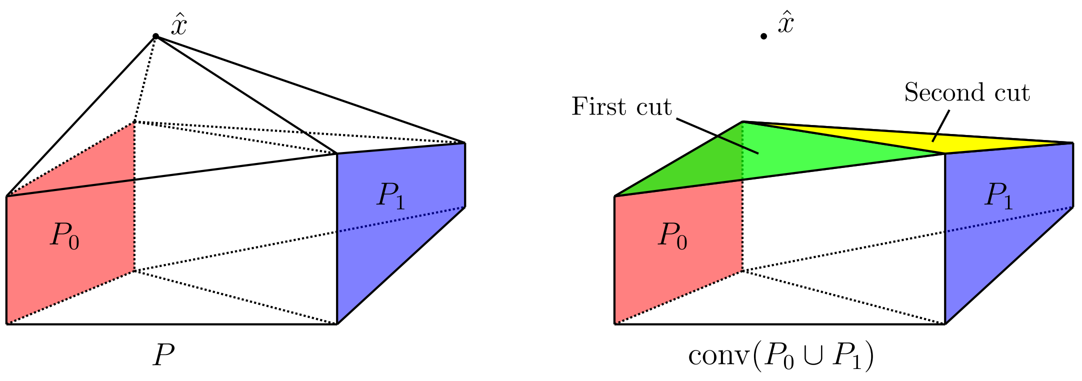


Figure 2.3: Different possible cuts for a disjunction.

We start by dualizing the CGLP which gives insightful geometric interpretations of the various normalization constraints. Note that the CGLP's variables are in fact row multipliers. This is why we call it the *dual* and call its dual linear program the *primal*.

$$\begin{array}{lrclcl}
\min & & & \beta - \hat{x}^\intercal \alpha & & \\
\text{s.t.} & -\alpha & + & A^\intercal w + \pi w_0 & = & 0 \\
& -\alpha & + & A^\intercal v - \pi v_0 & = & 0 \\
& \beta & - & b^\intercal w - \pi_0 w_0 & \geq & 0 \\
& \beta & - & b^\intercal v + (\pi_0 + 1) v_0 & \geq & 0 \\
& & & w, v & \in & \mathbb{R}^m_+ \\
& & & w_0, v_0 & \in & \mathbb{R}_+
\end{array} \tag{D}$$

Its dual is almost the same as the formulation in the lifted space except that the point $\hat{x}$ is the point which we want to separate. The primal problem reads

$$\begin{array}{llcl} \max & & 0 & \\ \text{s.t.} & Ax^{(1)} & \leq & by^{(1)} \\ & \pi^{\intercal}x^{(1)} & \leq & \pi_0 y^{(1)} \\ & Ax^{(2)} & \leq & by^{(2)} \\ & \pi^{\intercal}x^{(2)} & \geq & (\pi_0+1)y^{(2)} \\ & x^{(1)}+x^{(2)} & = & \hat{x} \\ & y^{(1)}+y^{(2)} & = & 1 \\ & y^{(1)},y^{(2)} & \in & \mathbb{R}_+ \end{array} . \tag{P}$$

Without normalizing the dual unboundedness is reflected in an infeasible primal problem (P) (see Figure 2.1 on page 8). Bounding the former corresponds to relaxing the latter in order to make it feasible. Different normalization constraints correspond to different types of relaxations on the primal side which we will present. The precise analysis of this correlation was done in [21].

2.3.1 α_p-Normalization Constraints

When they proposed lift-and-project cutting planes for the first time Balas, Ceria, and Cornuéjols [14] worked with normalizations that bound the coefficients or the right-hand side. We start with what they called the *LHS normalization*. The general α_p-normalization constraint reads

$$||\alpha||_p \leq 1. \tag{α_p-NC}$$

The only specializations which admit a *linear* formulation are for $p = 1$ and $p = \infty$.

$$\sum_{i=1}^{n} |\alpha_i| = 1 \tag{α_1-NC}$$

$$|\alpha_i| \leq 1 \quad (\forall i \in [n]) \tag{α_∞-NC}$$

According to [14], the extreme points of the truncated cone do not always correspond to extreme rays of $\mathcal{C}^{=}_{\geq}$. On the dual side we observe the following. To add the constraints $\sum |\alpha_i| \leq 1$ we have to introduce variables $\alpha^+, \alpha^- \in \mathbb{R}^n_+$ with $\alpha = \alpha^+ - \alpha^-$. The normalization is then $\mathbb{1}^\intercal \alpha^+ + \mathbb{1}^\intercal \alpha^- \leq 1$. In their article [21], Ceria and Soares refer to a theorem stating that dualizing an LP with an additional (not necessarily linear) constraint $||.|| \leq 1$ results in an objective function with a term $||.||_*$ where the $*$ indicates the *dual* norm. For us it is only relevant that $||.||_1$ and $||.||_\infty$ are dual to each other. We show only the specialized results here.

Proposition 2.9. *The relaxed version of* (P) *corresponding to* α_1*-NC reads*

$$\begin{array}{lrcl}
\min & & ||\hat{x} - x^*||_\infty & \\
s.t. & Ax^{(1)} & \leq & by^{(1)} \\
 & \pi^\intercal x^{(1)} & \leq & \pi_0 y^{(1)} \\
 & Ax^{(2)} & \leq & by^{(2)} \\
 & \pi^\intercal x^{(2)} & \geq & (\pi_0 + 1) y^{(2)} \\
 & x^{(1)} + x^{(2)} & = & x^* \\
 & y^{(1)} + y^{(2)} & = & 1 \\
 & y^{(1)}, y^{(2)} & \in & \mathbb{R}_+
\end{array}. \tag{2.6}$$

Proof. We add the constraints $\alpha = \alpha^+ - \alpha^-$ and $\mathbb{1}^\intercal \alpha^+ + \mathbb{1}^\intercal \alpha^- \leq 1$ with duals $\mu \in \mathbb{R}^n$ and $\pi \in \mathbb{R}_-$. When dualizing and making it a minimization problem $-\pi$ enters the objective while $x^{(1)} + x^{(2)} = \hat{x}$ is relaxed to $x^{(1)} + x^{(2)} = \hat{x} + \mu$. Furthermore, $I_n \mu + \mathbb{1}\pi \leq 0$ and $-I_n + \mathbb{1}\pi \leq 0$ are added. After scaling π by -1 and substituting $(\hat{x} + \mu)$ by x^* we obtain the result. □

Proposition 2.10. *The relaxed version of* (P) *corresponding to* α_∞*-NC reads*

$$\begin{array}{lrcl}
\min & & ||\hat{x} - x^*||_1 & \\
s.t. & Ax^{(1)} & \leq & by^{(1)} \\
 & \pi^\intercal x^{(1)} & \leq & \pi_0 y^{(1)} \\
 & Ax^{(2)} & \leq & by^{(2)} \\
 & \pi^\intercal x^{(2)} & \geq & (\pi_0 + 1) y^{(2)} \\
 & x^{(1)} + x^{(2)} & = & x^* \\
 & y^{(1)} + y^{(2)} & = & 1 \\
 & y^{(1)}, y^{(2)} & \in & \mathbb{R}_+
\end{array} \tag{2.7}$$

Proof. Here we add the duals $\lambda^+, \lambda^- \in \mathbb{R}^n_-$ for the constraints $\alpha \leq \mathbb{1}$ and $-\alpha \leq -\mathbb{1}$. Again, we dualize and make it a minimization problem. Then the objective is $-\mathbb{1}^\intercal\lambda^+ - \mathbb{1}^\intercal\lambda^-$ while $x^{(1)} + x^{(2)} = \hat{x}$ is relaxed to $x^{(1)} + x^{(2)} = \hat{x} + \lambda^+ - \lambda^-$. We substitute $(\hat{x} + \lambda^+ - \lambda^-)$ by x^* and obtain the result. $\square$

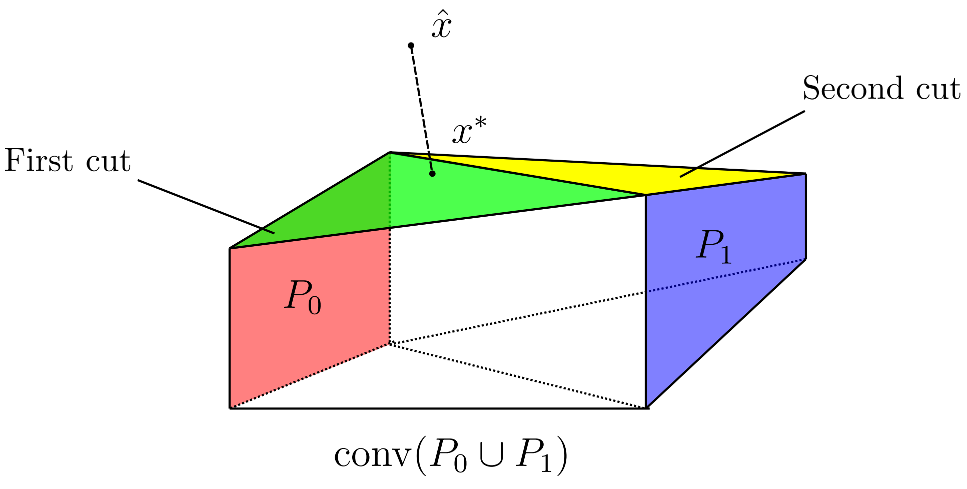


Figure 2.4: Find $x^* \in \mathrm{conv}\,(P_0 \cup P_1)$ with $\min ||\hat{x} - x^*||_q$.

The interpretation is the same for both. In order to make (P) feasible we relax the constraint that $\hat{x}$ must be in $\mathrm{conv}\,(P_0 \cup P_1)$. In fact, the LPs in Propositions 2.9 and 2.10 look for a point $x^* \in \mathrm{conv}\,(P_0 \cup P_1)$ such that the distance between $\hat{x}$ and x^* is minimized. The norm used for the distance is the dual to the norm used in the normalization constraint. Figure 2.4 illustrates the interpretation.

The derivation of a dual solution (a cut) from a given primal solution is not easily done for every normalization. Here, deriving a cut from x^* involves subgradients (see [21]) but is at least possible. But from this we can observe easily that such a cut is tight at x^*. With this in mind, in some sense, the optimal solutions to the corresponding CGLPs yield *geometrically deepest* cuts.

2.3.2 β-Normalization Constraint

The second kind of "primal" normalization constraints is the *β-normalization constraint*

$$|\beta| = 1. \tag{β-NC}$$

It was also introduced in [14] and its geometry analyzed in [21]. Again, to get some insight, we investigate the corresponding relaxation of (P). This is done for the constraint $\beta = \hat{\beta}$ with a dual variable $\lambda \in \mathbb{R}$. The next proposition states the resulting primal problem.

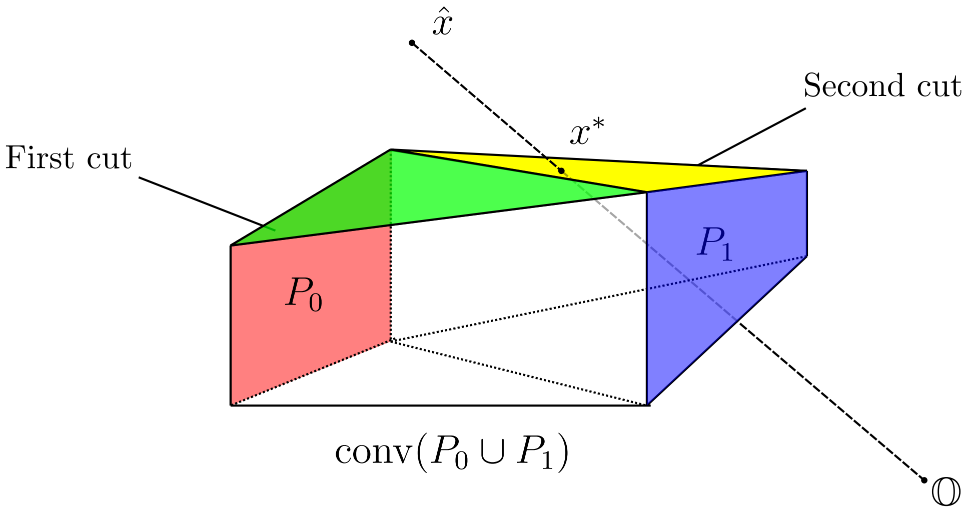


Figure 2.5: Find scaled $\hat{x}$ that is in conv $(P_0 \cup P_1)$.

Proposition 2.11. *The relaxed version of* (P) *corresponding to β-NC reads*

$$\begin{array}{lrcl}
\max & \multicolumn{3}{l}{\hat{\beta}(1 - y^{(1)} - y^{(2)})} \\
s.t. & Ax^{(1)} & \leq & by^{(1)} \\
 & \pi^\intercal x^{(1)} & \leq & \pi_0 y^{(1)} \\
 & Ax^{(2)} & \leq & by^{(2)} \\
 & \pi^\intercal x^{(2)} & \geq & (\pi_0 + 1) y^{(2)} \\
 & x^{(1)} + x^{(2)} & = & \hat{x} \\
 & y^{(1)}, y^{(2)} & \in & \mathbb{R}_+
\end{array} \tag{2.8}$$

Proof. We only have to introduce a single dual variable $\lambda \in \mathbb{R}$ for the equation $\beta = \hat{\beta}$. The original constraint $y^{(1)} + y^{(2)} = 1$ is relaxed to $y^{(1)} + y^{(2)} + \lambda = 1$ and the new objective is $\hat{\beta}\lambda$. Substituting λ finishes our proof. □

Obvious from Proposition 2.11 is that we are looking for a factor $\gamma = 1/\left(1 - y^{(1)} - y^{(2)}\right)$ such that $\gamma\hat{x} \in \operatorname{conv}(P_0 \cup P_1)$. Depending on the sign of $\hat{\beta}$ we are looking for a minimal $\gamma > 1$ or a maximal $\gamma < 1$. Also interesting is that the relaxed primal problem is still infeasible if $\hat{x} \notin \operatorname{cone}(\operatorname{conv}(P_0 \cup P_1))$.

2.3.3 Trivial Normalization Constraint

The other category of normalization constraints is based on the idea of bounding the multipliers w, w_0, v, and v_0. All of the following truncations have a very attractive property: There is a correspondence between the bases of (CGLP) and the bases of the original LP. We provide some more details in Section 4.4.5. The simplest version of this kind is the equation

$$w_0 + v_0 = 1 \tag{TNC}$$

and is called the *trivial normalization constraint*. The reason is that the *simple intersection cut* (see Section 3.2) is already an optimal solution. The latter can be obtained directly from the LP tableau. However, in Section 4.4.3 we will explain how Bonami was able to make it usable. This normalization is the last that was introduced by Balas, Ceria, and Cornuéjols in [14]. As in the previous cases we state the relaxed version of (P).

Proposition 2.12. *The relaxed version of* (P) *corresponding to TNC reads*

$$\begin{array}{lrcl}
\max & & \lambda & \\
s.t. & Ax^{(1)} & \leq & by^{(1)} \\
& \pi^\intercal x^{(1)} + \lambda & \leq & \pi_0 y^{(1)} \\
& Ax^{(2)} & \leq & by^{(2)} \\
& \pi^\intercal x^{(2)} - \lambda & \geq & (\pi_0 + 1) y^{(2)} \\
& x^{(1)} + x^{(2)} & = & \hat{x} \\
& y^{(1)} + y^{(2)} & = & 1 \\
& y^{(1)}, y^{(2)} & \in & \mathbb{R}_+
\end{array} \tag{2.9}$$

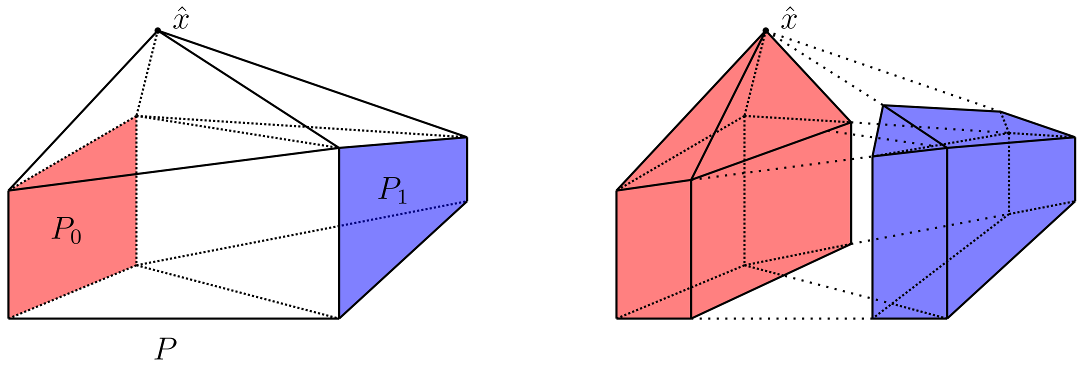


Figure 2.6: Shrink the split until $\hat{x}$ belongs to the convex hull.

The geometric meaning of this normalization gets clearer if we consider an elementary split disjunction, i.e., $\pi = e_k$ and $\pi_0 = 0$. The disjunction itself is relaxed to $x_k \leq -\lambda$ and $x_k \geq 1 + \lambda$ (note that here λ is usually negative.) However, $\hat{x}$ does not need to be in one of the two relaxed polyhedra (as Figure 2.6 might suggest) but in their convex hull.

2.3.4 Standard Normalization Constraint

We now turn to the state-of-the-art normalization constraint. It was suggested in [21] and first tested in [20].

$$\sum_{i=1}^{m} w_i + w_0 + \sum_{i=1}^{m} v_i + v_0 = 1 \tag{SNC}$$

Its associated primal problem is stated in the next proposition.

Proposition 2.13. *The relaxed version of* (P) *corresponding to SNC reads*

$$\begin{array}{lrcl}
\max & & \lambda & \\
s.t. & Ax^{(1)} + \lambda & \leq & by^{(1)} \\
& \pi^\intercal x^{(1)} + \lambda & \leq & \pi_0 y^{(1)} \\
& Ax^{(2)} + \lambda & \leq & by^{(2)} \\
& \pi^\intercal x^{(2)} - \lambda & \geq & (\pi_0 + 1) y^{(2)} \\
& x^{(1)} + x^{(2)} & = & \hat{x} \\
& y^{(1)} + y^{(2)} & = & 1 \\
& y^{(1)}, y^{(2)} & \in & \mathbb{R}_+
\end{array} . \tag{2.10}$$

This time all constraints of the two polyhedra P_0 and P_1 are relaxed at the same time until $\hat{x}$ is contained in one of the relaxations. The relaxation is illustrated in Figure 2.7.

The *standard normalization constraint* highly depends on the representation of a certain inequality $A_{i,*}x \leq b_i$ in that a scaled version of a row (scaled by some $\lambda > 1$) is preferred over the original row. This happens because it needs a smaller multiplier to get the same result.

If we interpret the values of the multipliers as a resource with capacity equal to 1 we will (on average) use approximately half of it for w and half of it for v. This means that the resulting cut is almost a convex combination of some inequalities of $Ax \leq b$ scaled by $1/2$. The scaling factor in turn means that incorporating a generated lift-and-project cut into another lift-and-project cut is penalized. This fact is considered as a reason that with the SNC the rank of the lift-and-project inequalities remains small even after several rounds of cut generation. Because typical MIPs usually have sparse rows rank 1 cuts with a small dual support (a small number of positive multipliers) are sparse as well.

2.3.5 Euclidean Normalization Constraint

Sometimes it is not helpful that the SNC depends on the representation of the polyhedron. Fischetti et al. introduced the so-called *Euclidean Normalization Constraint* in [27] to address this problem. They scaled each summand of the SNC by the norm of the corresponding inequality coefficient vector and obtained

$$\sum_{i=1}^{m} ||A_{i,*}||_2 (w_i + v_i) + ||\pi||_2 (w_0 + v_0) = 1. \tag{ENC}$$

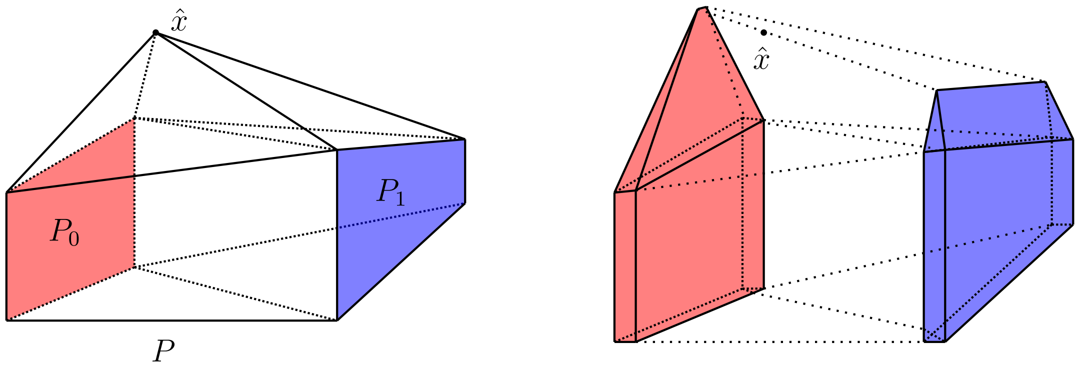


Figure 2.7: Enlarge the two polyhedra until $\hat{x}$ belongs to the convex hull.

Note that the geometric interpretation is the same as for the SNC except that the inequality-defining hyperplanes of the two polyhedra are "moved with a different speed".

2.3.6 Density Normalization Constraint

We present a variant of the Euclidean normalization constraint. Instead of making the choice of all constraints fair in the ENC we want to penalize dense constraints.

$$\sum_{i=1}^{m} |\text{supp}\,(A_{i,*})\,| \cdot ||A_{i,*}||_2(w_i + v_i) + ||\pi||_2(w_0 + v_0) = 1 \tag{DNC}$$

We do not call it Sparsity Normalization Constraint because of the ambiguity of (SNC). The interpretation in terms of resources is simple. The CGLP is allowed to incorporate two sparser inequalities instead of a single dense inequality with the same average of multipliers.

2.4 Different versions of the CGLP in Literature

When reading about lift-and-project cuts for the first time one may get confused because there are several different formulations for the CGLP. Therefore we present these variants how they correspond to each other and justify their existence. We start by restating the defining system for $\mathcal{C}^{=}_{\geq}$.

$$\begin{array}{rcl} \alpha^\intercal & = & w^\intercal A + w_0 \pi^\intercal \\ \alpha^\intercal & = & v^\intercal A - v_0 \pi^\intercal \\ \beta & \geq & w^\intercal b + w_0 \pi_0 \\ \beta & \geq & v^\intercal b - v_0(\pi_0 + 1) \\ w, v & \in & \mathbb{R}^m_+ \\ w_0, v_0 & \in & \mathbb{R}_+ \end{array} \tag{$\mathcal{C}^{=}_{\geq}$}$$

2.4.1 Equations for the Right-hand Side

We now assume that our polyhedron $P = P^{\leq}(A, b)$ is bounded in some direction and its reverse, i.e., there is some vector $c \in \mathbb{R}^n$ with $d_1 \leq c^\intercal x \leq d_2$ for all $x \in P$. We can combine the two inequalities and get $(c - c)^\intercal x \leq d_2 - d_1$. In other words, the trivial inequality $\mathbb{O}^\intercal x \leq 1$ is always valid for P. This simple fact can be used to enforce equality in the two β-equations in the CGLP:

$$\begin{array}{rcl} \alpha^\intercal & = & w^\intercal A + w_0 \pi^\intercal \\ \alpha^\intercal & = & v^\intercal A - v_0 \pi^\intercal \\ \beta & = & w^\intercal b + w_0 \pi_0 \\ \beta & = & v^\intercal b - v_0(\pi_0 + 1) \\ w, v & \in & \mathbb{R}^m_+ \\ w_0, v_0 & \in & \mathbb{R}_+ \end{array} \tag{$\mathcal{C}^{=}_{=}$}$$

Proposition 2.14. *The two versions ($\mathcal{C}^{=}_{\geq}$) and ($\mathcal{C}^{=}_{=}$) are equivalent up to scaling of α and β.*

Proof. Obviously, every solution to ($\mathcal{C}^{=}_{=}$) is also a feasible for ($\mathcal{C}^{=}_{\geq}$).

For the reverse direction let $(w, w_0, v, v_0, \alpha, \beta)$ be feasible for ($\mathcal{C}^{=}_{\geq}$). Furthermore, let $\lambda \in \mathbb{R}^m_+$ with $\lambda^\intercal A = \mathbb{O}$ and $\lambda^\intercal b = 1$ be multipliers for the normalized trivial inequality $\mathbb{O}^\intercal x \leq 1$. Then with $\beta_w := w^\intercal b + w_0 \pi_0$ and $\beta_v := v^\intercal b - v_0(\pi_0 + 1)$ we have that

$(w + \lambda(\beta - \beta_w), w_0, v + \lambda(\beta - \beta_v), v_0, \alpha, \beta)$ is feasible for ($\mathcal{C}^{=}_{=}$):

$$\begin{array}{lclcl}
(w + \lambda(\beta - \beta_w))^\intercal A + w_0\pi^\intercal & = & w^\intercal A + w_0\pi^\intercal & = & \alpha^\intercal \\
(v + \lambda(\beta - \beta_v))^\intercal A - v_0\pi^\intercal & = & v^\intercal A - v_0\pi^\intercal & = & \alpha^\intercal \\
(w + \lambda(\beta - \beta_w))^\intercal b + w_0\pi_0 & = & \beta_w + (\beta - \beta_w) & = & \beta \\
(v + \lambda(\beta - \beta_v))^\intercal b - v_0(\pi_0 + 1) & = & \beta_v + (\beta - \beta_v) & = & \beta
\end{array}$$

Scaling of the cut is necessary if we consider a multiplier-based normalization constraint (like TNC, SNC, ENC, or DNC). As we may add multiples of λ, the multiplier sum changes and can be made 1 again by scaling. □

Proposition 2.15. *If in an optimal solution to a normalized $\mathcal{C}^{=}_{\geq}$ the point $\hat{x}$ (resp. x^* for α_p-NC, β-NC) is not contained in the relaxed versions of P_0 or P_1 then the β-constraints are satisfied with equality.*

Proof. As $\hat{x}$ (resp. x^*) is not contained in P_0 or P_1 it must be a convex combination with nonzero multipliers. Hence, $y^{(1)}, y^{(2)} > 0$, and, by complementary slackness, the β-equations are satisfied with equality. □

Unfortunately, we do not know what happens if $y^{(1)} = 1, y^{(2)} = 0$, i.e., $\hat{x}$ is contained in the relaxed P_0. Hence, we have no proof that optimal solutions to $\mathcal{C}^{=}_{\geq}$ correspond to optimal solutions to $\mathcal{C}^{=}_{=}$, assuming the same normalization.

2.4.2 Inequalities for the Cut Coefficients

Another very prominent model is one where $P^{\leq}(A, b) \subseteq \mathbb{R}^n_+$ is required implicitly. Here, the constraints $-x_j \leq 0$ ($j \in [n]$) can be used in the same manner as the trivial inequality in the version above.

$$\begin{array}{rcl}
\alpha^\intercal & \leq & w^\intercal A + w_0\pi^\intercal \\
\alpha^\intercal & \leq & v^\intercal A - v_0\pi^\intercal \\
\beta & \geq & w^\intercal b + w_0\pi_0 \\
\beta & \geq & v^\intercal b - v_0(\pi_0 + 1) \\
w, v & \in & \mathbb{R}^m_+ \\
w_0, v_0 & \in & \mathbb{R}_+
\end{array} \qquad (\mathcal{C}^{\leq}_{\geq})$$

Proposition 2.16. *The CGLP cone $\mathcal{C}^{\leq}_{\geq}$ is equivalent to $\mathcal{C}^{=}_{\geq}$ if $P^{\leq}(A, b) \subseteq \mathbb{R}^n_+$ and the normalizations do not depend on the multipliers of the (possibly implicit) $-x_j \leq 0$ constraints.*

Proof. As in the proof of Proposition 2.14, we can use the $-x_j \leq 0$ constraints "for free" in order to decrease ${\alpha_w}^\intercal := w^\intercal A + w_0\pi^\intercal$ and ${\alpha_v}^\intercal := v^\intercal A - v_0\pi^\intercal$ element-wise until they equal $\alpha^\intercal$. □

This also means that if the nonnegativity constraints are contained in $Ax \leq b$ we can remove these rows from A and b before feeding them into $\mathcal{C}^{\leq}_{\geq}$. Again, if the normalization depends on the mentioned multipliers we may need to scale the solution afterwards.

2.4.3 Relaxed Multiplier Domains

As observed earlier, we can relax the nonnegativity constraints for w_0 and v_0. By Remark 2.7, a CGLP solution with negative objective value implies $w_0, v_0 > 0$ and it yields a valid cutting plane $\alpha^\intercal x \leq \beta$. Note that we must be careful with this relaxation as a nonnegative objective value may give an invalid inequality. An example can be found in Remark 4 of [17]. This relaxation can be applied to all CGLP cones mentioned earlier. We denote them by a zero index, i.e $\mathcal{C}^{=}_{\geq 0}$, $\mathcal{C}^{=}_{=0}$, $\mathcal{C}^{\leq}_{=0}$, and $\mathcal{C}^{\leq}_{\geq 0}$. We will use it in Section 4.4.3.

We now observe what happens on the primal side when w_0 or v_0 are unconstrained. In (P) we demand that $x^{(1)}$ (resp. $x^{(2)}$) lie on the hyperplanes $\pi^\intercal x = \pi_0$ (resp. $\pi^\intercal x = \pi_0 + 1$). If we want to separate a point that is *in* the split this is no restriction. Hence, this relaxation can be used for cut generation.

2.5 A Posteriori Cut Strengthening

Cut Strengthening was introduced in Balas and Jeroslow [11] and is a method to increase coefficients of a computed lift-and-project cut. The idea is that for an elementary split disjunction $\pi = e_k$ the cut only uses the integrality of one variable x_k. We therefore search for the best π for fixed multipliers (w, w_0, v, v_0).

To the best of our knowledge, cut strengthening only works for $P \subseteq \mathbb{R}_+$, i.e., our given multipliers are a solution to

$$\begin{aligned}
\alpha^\intercal &\leq w^\intercal A + w_0 e_k \\
\alpha^\intercal &\leq v^\intercal A - v_0 e_k \\
\beta &= w^\intercal b + w_0 \pi_0 \\
\beta &= v^\intercal b - v_0(\pi_0 + 1)
\end{aligned}$$

The cut coefficients verify $\alpha_j = \min\{w^\intercal A_{*,j} + w_0\pi_j^\intercal, v^\intercal A_{*,j} - v_0\pi_j^\intercal\}$. There are two "suggested" coefficients $w^\intercal A_{*,j}$ and $v^\intercal A_{*,j}$ which are modified by a scaled version of π_j. After the modification the minimum of both must be selected as the cut coefficient.

In order to increase that minimum we can change π_j. If done the right way, the smaller of the "suggested" coefficients increases while the other decreases.

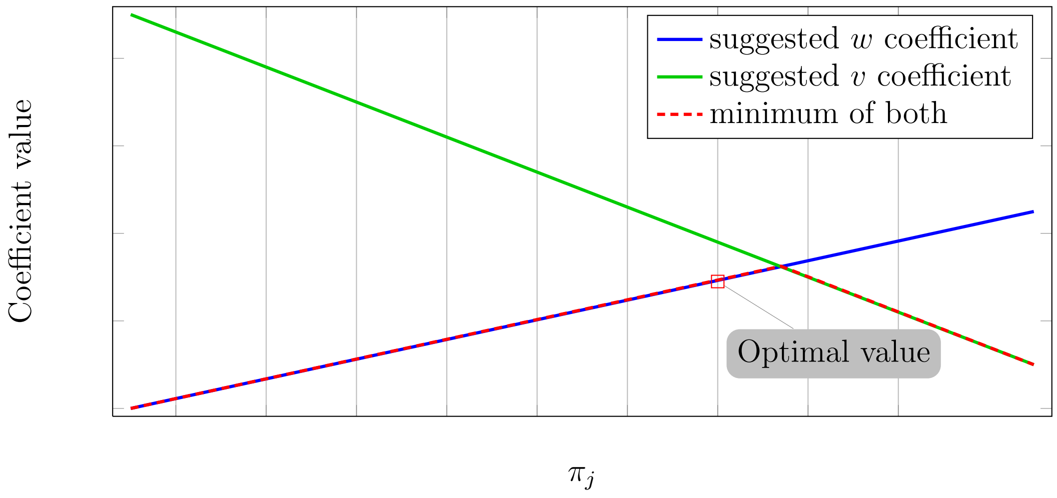


Figure 2.8: Finding an optimal disjunction for component j.

At $\pi_j = (v^\intercal A_{*,j} - w^\intercal A_{*,j}) / (w_0 + v_0)$ they coincide, i.e., $w^\intercal A_{*,j} + w_0\pi_j = v^\intercal A_{*,j} - v_0\pi_j$. If $\pi_j \notin \mathbb{Z}$, the optimal value is either $\lfloor \pi_j \rfloor$ or $\lceil \pi_j \rceil$ (see Figure 2.8). For $\lfloor \pi_j \rfloor$, the minimum of the coefficients equals $w^\intercal A_{*,j} + w_0\lfloor \pi_j \rfloor$ while for $\lceil \pi_j \rceil$ it is $v^\intercal A_{*,j} - v_0\lceil \pi_j \rceil$. Hence, we observe the following formulas for coefficient strengthening:

$$\begin{aligned} \hat{\alpha}_j &:= \max\{w^\intercal A_{*,j} + w_0\lfloor \hat{m}_j \rfloor, v^\intercal A_{*,j} - v_0\lceil \hat{m}_j \rceil\} && \text{for } j \in I \\ \hat{\alpha}_j &:= \min\{w^\intercal A_{*,j}, v^\intercal A_{*,j}\} && \text{for } j \notin I \end{aligned} \tag{2.11}$$

Chapter 3

Properties of the Lift-and-Project Cutting Planes

In the previous chapter we focused on the lift-and-project method and the necessary aspects for cut generation. The method and the cuts have certain properties which connect us with other interesting theory. As they are only of minor interest for our implementation-related purposes we collect them in a separate chapter.

We start by presenting a property already known in the very beginning of lift-and-project. We go on by showing the embedding of this specific type of cutting planes in the world of general-purpose cutting planes. The relationships also imply certain statements about complexity of cut separation. We finish this chapter by describing the known results and open problems in this field.

As these topics are only partially related to main goals of this work we omit proofs or even theorems but try to give a rough understanding instead. For the interested reader we highly recommend to read the mentioned literature.

3.1 Sequential Convexification

When comparing them with the convexification approach from [33] and [40], Balas et al. proved that the lift-and-project cuts have a theoretically attractive property. The next theorem and its corollary correspond to their Theorem 2.2 and Corollary 2.3 in [14]. They only hold for (mixed) 0-1 problems, that is, all integer variables must have 0-1 bounds.

Let $P^{(k_1,\dots,k_t)} := P^{(k_t)}\left(P^{(k_1,\dots,k_{t-1})}\right)$ denote the sequential application of the lift-and-project procedure to P using disjunctions $x_{k_i} \le 0 \quad \vee \quad x_{k_i} \ge 1$ for $i = k_1, \dots, k_t$.

Theorem 3.1 (Balas, Ceria, and Cornuéjols, 1993)**.** *Let $(k_1, \dots, k_t) \in [n]^t$ be a sequence where $0 \le x_{k_i} \le 1$ is valid for P for every $i \in [t]$. Then*

$$P^{(k_1,\dots,k_t)} = \operatorname{conv}\{x \in P : x_{k_i} \in \{0,1\} \ \forall i = 1, \dots, t\}. \tag{3.1}$$

As the proof is not very instructive we skip it here and refer to [14].

Corollary 3.2.

$$P^{(1,\dots,n)} = P_I \tag{3.2}$$

This property is known as *sequential convexification* and means the following: Once we add all lift-and-project cuts for some variable x_k we will never have to consider this variable again even after applying another lift-and-project iteration for another variable. Note that the number of lift-and-project cuts is usually exponential in the size of the problem. Furthermore, we would have to add *all* cuts and not only those violated by some LP solution.

3.2 Relation to Basic Intersection and GMI Cuts

In literature, the equivalence between strengthened lift-and-project cuts and Gomory mixed-integer (GMI) cuts is mentioned very often. Here, we must be very careful with the terminology as there are different types of GMI cuts. We introduce the necessary concepts and then state the equivalence results. We won't repeat the proofs as they are technical and do not contribute to our goal to give an overview. The most important fact is that we always consider *basic* cuts, i.e they are associated to a certain (not necessary feasible) basis of the LP relaxation.

For the remainder of this section we assume our polyhedra relaxation to be

$$P = \{x \in \mathbb{R}^n : Ax = b \quad \wedge \quad x \ge 0\}. \tag{3.3}$$

As always, $I \subseteq [n]$ denotes the set of variables supposed to be integral. We consider a row from a simplex tableau with respect to a simplex basis $B \subset [n]$ and $J = [n] \setminus B$. Without loss of generality, $k \in B$ shall index the row and the variable x_k which is represented as

$$x_k = \bar{b}_k - \sum_{j \in J} \bar{a}_{k,j} x_j. \tag{3.4}$$

Let $f_0 := \bar{b}_k - \lfloor \bar{b}_k \rfloor$ and $f_j := \bar{a}_{k,j} - \lfloor \bar{a}_{k,j} \rfloor$. Then the *basic simple intersection cut* from the convex set $\{x : \lfloor \bar{b}_k \rfloor \le x_k \le \lceil \bar{b}_k \rceil\}$ applied to (3.4) is

$$\sum_{j \in J} \max\{\bar{a}_{k,j}(1-f_0), -\bar{a}_{k,j} f_0\} x_j \ge f_0(1-f_0). \tag{3.5}$$

It originates from the disjunction itself. We observe

$$\begin{array}{lclcl} x_k \le \lfloor \bar{b}_k \rfloor & \Leftrightarrow & \bar{b}_k - \sum_{j \in J} \bar{a}_{k,j} x_j \le \bar{b}_k - f_0 & \Leftrightarrow & \sum_{j \in J} \bar{a}_{k,j} x_j \ge f_0 \\ x_k \ge \lceil \bar{b}_k \rceil & \Leftrightarrow & \bar{b}_k - \sum_{j \in J} \bar{a}_{k,j} x_j \ge \bar{b}_k + (1-f_0) & \Leftrightarrow & \sum_{j \in J} -\bar{a}_{k,j} x_j \ge 1 - f_0 \end{array}. \tag{3.6}$$

As $x \ge 0$, the two inequalities can be combined via a component-wise maximum. These cuts have been introduced in [7]. Such a cut can be strengthened using the integrality of other variables in $I \cap J$. Using the convex set $\{\lfloor \bar{b}_k \rfloor \le \pi^\intercal x \le \lceil \bar{b}_k \rceil\}$ with

$$\pi_j := \begin{cases} \lfloor \bar{a}_{k,j} \rfloor & j \in I \cap J \quad \wedge \quad f_j \le f_0 \\ \lceil \bar{a}_{k,j} \rceil & j \in I \cap J \quad \wedge \quad f_j > f_0 \\ 1 & j = k \\ 0 & \text{otherwise} \end{cases}, \tag{3.7}$$

we obtain the *basic Gomory mixed-integer cut* (3.8).

$$\sum_{\substack{j \in I \cap J \\ f_j \le f_0}} f_j(1-f_0)x_j + \sum_{\substack{j \in I \cap J \\ f_j > f_0}} (1-f_j) f_0 x_j + \sum_{j \in J \setminus I} \max\{\bar{a}_{k,j}(1-f_0), -\bar{a}_{k,j} f_0\} x_j \ge f_0(1-f_0) \tag{3.8}$$

A proof of the dominance over the basic simple intersection cut can be found in [5].

In his paper with Egon Balas (and later in his dissertation [37]), Michael Perregaard showed a correspondence between basic simple intersection cuts and lift-and-project cuts with the SNC and a simple disjunction $x_k \le 0 \quad \vee x_k \ge 1$. It was shown only for 0-1-problems but a generalization to general MIPs is possible. Furthermore, they proved that the two ways of strengthening are equivalent, i.e., the basic GMI cuts correspond to strengthened lift-and-project cuts. We provide an explanation which is not too technical. The details along with the proofs can be found in Theorems 4a, 4b and 5 of [13].

Crucial for the equivalence is an index pair (M_1, M_2) with $M_1, M_2 \subseteq [n]$ and $M_1 \cap M_2 = \emptyset$. For the intersection and GMI cuts it indexes a partition of the nonbasic variables $J = M_1 \cup M_2$ according to the sign of a certain tableau row expression. In the lift-and-project context M_1 (resp. M_2) indexes the basic variables of w (resp. v). It is possible that the M_i index variables *and* inequalities because the latter are all lower bounds and hence correspond to a single variable.

We sketch how to derive the intersection cut (resp.) GMI cut from a basic solution of the CGLP with $\pi = e_k$. Let $J := M_1 \cup M_2$ be as above. The theorems in [13] guarantee that $[n] \setminus J$ indexes a valid LP basis for the relaxation which contains k. Hence, x_k can be written as a linear combination of x_J. The simple intersection cut from this tableau row and the same disjunction is equivalent to the cut from the basic CGLP solution.

For the reverse direction we have to find a partition of the nonbasics J in M_1 and M_2 satisfying the mentioned sign pattern. Note that this partition does not need to be unique. Then the variables $w_{M_1}, w_0, v_{M_2}, v_0, \alpha, \beta$ form a basis of the CGLP. Furthermore, $\alpha^\intercal x \leq \beta$ is equivalent to the given intersection cut.

3.3 Elementary Closures and Separation Complexity

The notion of a cutting plane closure is important in order to compare different types cutting planes. Even though we saw that certain cutting planes are equivalent this is not always the case. An *elementary closure* of a certain family of cutting planes is defined as the intersection of all cuts in the family. We know several such families along with properties like being a polyhedron. Some of them yield the same closures while others are contained in another or turned out to be incomparable. In 2000 Cornuéjols and Li published a very good overview [24] about the elementary closures where they compared 18 closures from literature. They established almost the relationships between these families.

A first result which is now obvious for us is that the lift-and-project closure and the closure of basic intersection cuts are the same. Their strengthened versions yield a different closure defined by basic GMI cuts and strengthened lift-and-project cuts. Considering only tableaus from basic *feasible* solutions another result is that the elementary closure (with or without strengthening) is larger. On the other hand, if we generalize GMI cuts to *nonbasic* solutions (any row combination of tableau rows) then the obtained closure will be strictly smaller. By smaller (resp. larger) we mean that closure C_1 is contained in C_2 (resp. vice versa). A smaller closure means that for a certain $\hat{x}$ there may exist a cut which cuts deeper in the relaxation which in turn is what we try to accomplish.

The theoretical complexity of separating cuts is very important. An $\mathcal{NP}$-hard separation problem implies that the search for an efficient (in the sense of polynomial run time) separation algorithm is impossible if we believe in the conjecture "$\mathcal{P} \neq \mathcal{NP}$". On the other hand it does not necessarily mean that there is no method which does the separation quite fast in practice. But the existence of a polynomial time separation algorithm is a good motivation to look for the latter.

The separation of lift-and-project cuts is very interesting when it comes to complexity theory. Whether there exists a lift-and-project cut that separates a certain $\hat{x}$ can be decided by solving the CGLP for every fractional variable. As linear programming is possible in polynomial time (see [31]) we obtain a polynomial time separation algorithm. The complexity of the separation problem for strengthened lift-and-project cuts is an open question while the separation of the general GMI cuts (not necessarily associated to a basis) is an $\mathcal{NP}$-hard problem. Solving the CGLP for all fractional variables does not help because if there is no unstrengthened lift-and-project cut the decision whether some of the many valid lift-and-project inequalities can be strengthened to cut off $\hat{x}$ is nontrivial.

Chapter 4

Integrating Lift-and-Project in a MIP-Solver

This chapter is dedicated to the more practical side of lift-and-project. We start with an informal description of a MIP solver's structure and refer to more articles and documentation. In the next two sections we state the typical representation of polyhedra in practice and describe bound shifting and variable complementing. These are two transformations which are used to move a polytope into the first orthant. Here we also show how these affect the CGLP and its possible normalizations. As the computational effort to solve the CGLP in its most general form is very large we also describe some methods how to reduce this effort to a moderate amount. We close this chapter by stating the details about our implementation.

4.1 How a Modern MIP Solver Works

Currently, there are quite a few different MIP solvers available. An overview over the performance and quality can be found at H. Mittelmann's benchmark page at [35]. Almost all of them have common basics and we give a short overview here.

The backbone of a MIP solver is a branch & bound (B&B) algorithm. Typical branching types are splitting via a disjunction just like in the cutting case or branching on special ordered sets. Bounding is done by solving a linear relaxation over the restricted set of constraints. The solver maintains the best known feasible solution (primal bound) and the smallest objective value (note that we minimize) of all unprocessed branch & bound nodes (dual bound). If the objective value of a linear relaxation of some node is greater than the primal bound this node can be discarded. Of course, if the linear relaxation is infeasible the node is discarded, too.

In addition to this simple algorithm the state-of-the-art solvers have many more other ingredients to speed up the solving process. Most of them are *cut & branch* solvers as they add cutting planes in the root node, that is, before branching starts. Some are even *branch & cut* solvers which means that they also add cuts that are locally valid for a branching node and its children. Although we will provide some more details about our cutting plane implementation in Section 4.5 we refer to [41] for further reading about the general topic. Since some years, also constraint propagation is used to strengthen the linear relaxation (see [4]). Its idea is that changes in variable domains (e.g. during branching) are propagated through the problem to influence other variables' domains. In order to find a good feasible solution quickly primal heuristics are run frequently (see [16]). They use gathered information to guess solutions via rounding or solving smaller subproblems.

There are different modeling languages to translate (combinatorial) optimization problems into mixed-integer problems (see [30], [32], [28]). These languages often introduce redundant constraints or unnecessary variables. To cope with these unnecessarily large models all solvers have at least some mechanisms to simplify MIPs. In addition to the removal of redundancy unspecified (but implicitly valid) variable bounds are added.

For further reading we recommend [1] for a very good overview.

4.2 Representation of the Linear Relaxation

So far we assumed that our polyhedron is of the form

$$Ax \leq b \text{ and } x \in \mathbb{R}^n. \tag{4.1}$$

This is not a suitable representation for a practical implementation. For example, consider the case of binary variables. For every such variable, the two constraints $x_k \leq 1$ and $-x_k \leq 0$ have to be considered. These simple inequalities are called variable bounds for obvious reasons. As inequalities are modeled (though not explicitly rewritten) as equations with a slack variable a so-called *ranged row* $b_l \leq a^\intercal x \leq b_r$ can be used without more effort. The reason is that this row needs only one slack variable which has upper and lower bounds instead of just one of the two. In the same way equations can be handled by setting b_l equal to b_r. Note that we still assume a minimization problem. The resulting representation reads

$$\begin{array}{llcccc} \min & & & c^{\intercal}x & & \\ \text{s.t.} & b_l & \leq & Ax & \leq & b_r \\ & l & \leq & x & \leq & u \\ & & & x \in \mathbb{R}^n & & \end{array} . \tag{4.2}$$

Of course, we allow values $+\infty$ (resp. $-\infty$) for b_r and u (resp. b_l and l).

All modern implementations of the simplex algorithm actually implement the *revised simplex method*. Here, among other issues, sparsity is exploited heavily. Many real-world problem instances have a sparse matrix A, i.e., many entries are zero. For example, precedence constraints (for binary variables) have exactly two nonzero entries. Many combinatorial problems are graph-related and the corresponding linear constraints characterize a local property of the graph or some structure on the graph. This locality implies that only a small number of variables is concerned for each constraint.

For cutting planes the sparsity of A has an important impact. Many cutting plane methods derive cuts as linear combinations of already existing inequalities. If the method tends to select only a small number of inequalities for the linear combination the resulting cut is also (relatively) sparse. We call this feature *dual sparsity* because the row multipliers of the linear combination live in the dual space. Based on this motivation we will formally define (dual) sparsity later in Section 5.1.

4.3 Bound Shifting and Variable Complementing

Bound shifting means to replace some variable x_j by $\widetilde{x_j}+\gamma$. In the context of 0-1 problems, complementing a variable x_j means to replace it by $1-\widetilde{x_j}$. In a more general context we define it as flipping the sign of x_j as we may apply bound shifting afterwards.

Usually, these two concepts are applied to move a polyhedron $P^{\leq}(A,b)$ into the first orthant (see Sections 2.5 and 4.4.2). This is always possible if there is no free variable. Otherwise, one has to find a workaround. From a geometry point of view we do not expect serious changes for the CGLP because the original polyhedron is just mirrored or moved.

To complement a variable x_j the solver has to scale $A_{*,j}$ by -1. A bound shift manifests itself as a replacement of b by $(\widetilde{b} - \gamma A_j)$. Note that both procedures retain sparsity of A. Here, we show how they interact with the CGLP.

Lemma 4.1. *Complementing a variable x_j does not affect $\mathcal{C}_{\geq}^{=}$ if π_j is replaced by $-\pi_j$. None of the discussed normalization constraints is affected.*

Proof. We state the updated CGLP cone which differs from the original $(\mathcal{C}_{\geq}^{=})$ only by the scaled column $A_{*,j}$ and changed π_j.

$$\begin{array}{rcl} \widetilde{\alpha}_i & = & w^\intercal A_{*,i} + w_0\pi_i \quad (\forall i \neq j) \\ \widetilde{\alpha}_i & = & v^\intercal A_{*,i} - v_0\pi_i \quad (\forall i \neq j) \\ \widetilde{\alpha}_j & = & w^\intercal(-A_{*,j}) + w_0(-\pi_j) \\ \widetilde{\alpha}_j & = & v^\intercal(-A_{*,j}) - v_0(-\pi_j) \\ \beta & \geq & w^\intercal b + w_0\pi_0 \\ \beta & \geq & v^\intercal b - v_0(\pi_0+1) \\ w, v & \in & \mathbb{R}^m_+ \\ w_0, v_0 & \in & \mathbb{R}_+. \end{array} \tag{4.3}$$

We show that a tuple $(w, w_0, v, v_0, \alpha, \beta)$ is feasible for $(\mathcal{C}_{\geq}^{=})$ if and only if the tuple $(w, w_0, v, v_0, \widetilde{\alpha}, \beta,)$ is feasible for (4.3) where $\widetilde{\alpha}_i = \alpha_i$ for $i \neq j$, and $\widetilde{\alpha}_j = -\alpha_j$. The only changes are in the constraints for α_j, where every summand is just scaled by -1. As $\widetilde{x}_i^* = x_i^*$ for $i \neq j$ and $\widetilde{x}_j^* = -x_j^*$, the cuts $\alpha^\intercal x \leq \beta$ and $\widetilde{\alpha}^\intercal \widetilde{x} \leq \beta$ are the same as the only difference is $\alpha_j \cdot x_j = (-\alpha_j) \cdot (-x_j)$.

Because the multipliers and β remain unchanged the normalization constraints TNC, SNC, ENC, DNC, and β-NC are not affected. Furthermore, α_1-NC and α_∞-NC depend on $|\alpha_i|$ which also did not change. □

We now turn to the case of shifting bounds, i.e., replacing variable x_j by $\widetilde{x}_j + \gamma$.

Lemma 4.2. *Shifting a variable x_j by γ does not affect $\mathcal{C}_{\geq}^{=}$ if $\gamma\pi_j \in \mathbb{Z}$. Among the discussed normalization constraints only β-NC is affected.*

Proof. First, observe that $\widetilde{\pi}_0 := \lfloor \pi^\intercal \widetilde{x}^* \rfloor = \lfloor \pi^\intercal x^* \rfloor - \gamma\pi_j$ as $\gamma\pi_j \in \mathbb{Z}$. Again, we state the updated CGLP cone which differs from the original $(\mathcal{C}_{\geq}^{=})$ only by the changed right-hand side $(b - \gamma A_{*,j})$ and the different $\widetilde{\pi}_0$.

$$\begin{array}{rcc} \alpha^\intercal & = & w^\intercal A + w_0 \pi^\intercal \\ \alpha^\intercal & = & v^\intercal A - v_0 \pi^\intercal \\ \widetilde{\beta} & \geq & w^\intercal (b - \gamma A_{*,j}) + w_0(\pi_0 - \gamma \pi_j) \\ \widetilde{\beta} & \geq & v^\intercal (b - \gamma A_{*,j}) - v_0(\pi_0 - \gamma \pi_j + 1) \\ w, v & \in & \mathbb{R}^m_+ \\ w_0, v_0 & \in & \mathbb{R}_+. \end{array} \tag{4.4}$$

We show that a tuple $(w, w_0, v, v_0, \alpha, \beta)$ is feasible for $(\mathcal{C}^=_\geq)$ if and only if the tuple $(w, w_0, v, v_0, \alpha, \widetilde{\beta})$ is feasible for (4.4) where $\widetilde{\beta} = \beta - \gamma \alpha_j$. The right-hand sides are related as follows.

$$\begin{aligned} \widetilde{\beta} &= \max\{w^\intercal (b - \gamma A_{*,j}) + w_0(\pi_0 - \gamma \pi_j), v^\intercal (b - \gamma A_{*,j}) - v_0(\pi_0 - \gamma \pi_j + 1)\} \\ &= \max\{w^\intercal b + w_0 \pi_0 - \gamma(w^\intercal A_{*,j} + w_0 \pi_j), v^\intercal b - v_0(\pi_0 + 1) - \gamma(v^\intercal A_{*,j} - v_0 \pi_j)\} \\ &= \max\{w^\intercal b + w_0 \pi_0 - \gamma \alpha_j, v^\intercal b - v_0(\pi_0 + 1) - \gamma \alpha_j\} \\ &= \max\{w^\intercal b + w_0 \pi_0, v^\intercal b - v_0(\pi_0 + 1)\} - \gamma \alpha_j \\ &= \beta - \gamma \alpha_j \end{aligned}$$

The associated cuts are the same:

$$\begin{aligned} \alpha^\intercal x \leq \beta \quad &\Leftrightarrow \quad \alpha^\intercal x - \gamma \alpha_j \leq \beta - \gamma \alpha_j \\ &\Leftrightarrow \quad \sum_{i \neq j} \alpha_i x_i + \alpha_j (x_j - \gamma) \leq \widetilde{\beta} \\ &\Leftrightarrow \quad \alpha^\intercal \widetilde{x} \leq \widetilde{\beta} \end{aligned}$$

Because the multipliers and α remain unchanged the normalization constraints TNC, SNC, ENC, DNC, α_1-NC, and α_∞-NC are not affected. □

By Lemma 4.1 and Lemma 4.2 we can safely assume $P^\leq(A, b) \subseteq \mathbb{R}^n_+$ if we use any of the mentioned normalization constraints, except for β-NC.

We now discuss the case of β-NC. As the right-hand side of the cuts changes the β-normalization constraint may not be satisfied anymore after performing a bound shift.

Consider the polytope

$$P = \operatorname{conv}\left\{ \begin{pmatrix}0\\0\\4\end{pmatrix}, \begin{pmatrix}1\\0\\5\end{pmatrix}, \begin{pmatrix}1\\-1\\3\end{pmatrix}, \begin{pmatrix}0\\-1\\5\end{pmatrix}, \begin{pmatrix}1/2\\1/2\\6\end{pmatrix} \right\}. \tag{4.5}$$

If we remove $\hat{x} = (1/2, 1/2, 6)^\intercal$ from the defining set we get the extreme points of P_I, i.e., it is the convex hull of the first four integral points. $\hat{x}$ is the optimal solution of the linear program $\max\{z : (x, y, z) \in P\}$. P_I has exactly two facets which separate $\hat{x}$ and both of them can be obtained by an application of lift-and-project on variable y. They are

$$-x + y + z \leq 4 \quad \text{and} \quad 2x - 2y + z \leq 7 \tag{4.6}$$

and $\hat{x}$ violates both by 1. Normalized with $\beta = 1$ we have objective values of the CGLP of 1/4 and 1/7, respectively. The optimal solution is then the first inequality.

Now we consider the polytope $P + \{(-1, 1, 0)^\intercal\}$ (where + indicates the *Minkowski sum*). In other words, we moved P one unit in y direction and one unit in $-x$ direction. Now, the two possible cuts verify

$$-x + y + z \leq 6 \quad \text{and} \quad 2x - 2y + z \leq 3 \tag{4.7}$$

The normalized cut violations are now 1/6 and 1/3, that is, the second inequality is optimal now. Note that the distance from the hyperplanes is $1/\sqrt{3}$ and 1/3, respectively. Hence, P is not symmetric in the sense that the choice of the cut was arbitrary. This example illustrates that the normalization β-NC is not suitable if we investigate cut separation from a geometry point of view.

4.4 Reducing the Computation Time

The huge size of the CGLP comes from the fact that there are two variables for every constraint in the original problem. In the general case we have to consider bounds as usual inequalities, too. On the contrary, only a small number of those multiplier variables will be nonzero as such a nonzero variable must be basic.

This fact made people think about removing certain multipliers in the hope that they will be zero in the optimal solution anyway. We present two such ideas from literature.

When using the TNC along with $\mathcal{C}^{=}_{\geq 0}$ its dual shrinks to a smaller problem. Another way of working with an LP which is only as large as the original LP relaxation is to simulate CGLP pivots in the original tableau. How these two approaches work is shown in the last parts of this section.

4.4.1 Removing Certain Multipliers

In [12] the authors suggest to use the equivalence to intersection cuts (see Section 3.2) in order to compute a basic feasible solution to the CGLP. On the one hand it can be used to warm-start the CGLP to reduce the run time. On the other hand, Balas and Perregaard suggested to remove certain multipliers after producing the mentioned solution. They tried to only keep those with negative reduced costs (they used a maximization problem formulation). This is common practice as the reduced costs are one indicator used the by simplex algorithm implementation to decide which variables might enter the basis.

4.4.2 Solving in the Subspace of Fractional Variables

Balas et al. were able to prove that it is helpful to solve the CGLP in the space of fractional variables only. In [14] they described how to lift a locally valid cut to a higher dimensional space in order to make it globally valid. Such a solution of the original space CGLP can be obtained by a simple formula.

In practice we have bounds $l \leq x \leq u$ for $l \in \mathbb{R}^n \cup \{-\infty\}$ and $u \in \mathbb{R}^n \cup \{\infty\}$. The mixed-integer problem is then

$$\begin{array}{rrcl} \max & c^{\intercal}x & & \\ & Ax & \leq & b \\ & Ix & \leq & u \\ & -Ix & \leq & -l \end{array} \tag{4.8}$$

We define the CGLP cone $\mathcal{C}^{=}_{\geq}(X, U, L)$ for subsets of variables $X \subseteq [n]$ and subsets of upper (resp. lower) bounds U (resp. L) $\subseteq [n]$ by

$$\begin{array}{rcl} \alpha_X^\intercal & = & w^\intercal A_{*,X} + \overline{w}_U^\intercal I_{|U|} - \underline{w}_L^\intercal I_{|L|} + w_0 \pi_X^\intercal \\ \alpha_X^\intercal & = & v^\intercal A_{*,X} + \overline{v}_U^\intercal I_{|U|} - \underline{v}_L^\intercal I_{|L|} - v_0 \pi_X^\intercal \\ \beta & \geq & w^\intercal b + \overline{w}_U^\intercal u_U - \underline{w}_L^\intercal l_L + w_0 \pi_0 \\ \beta & \geq & v^\intercal b + \overline{v}_U^\intercal u_U - \underline{v}_L^\intercal l_L - v_0(\pi_0 + 1) \\ w, v & \in & \mathbb{R}_+^m \\ \overline{w}_U, \overline{v}_U & \in & \mathbb{R}_+^U \\ \underline{w}_L, \underline{v}_L & \in & \mathbb{R}_+^L \\ w_0, v_0 & \in & \mathbb{R}_+ \end{array} . \qquad (\mathcal{C}^{=}_{\geq}(X, U, L))$$

Note that $\mathcal{C}^{=}_{\geq}([n], [n], [n])$ equals ($\mathcal{C}^{=}_{\geq}$) for the above MIP.

Let $R \subseteq [n]$ be the set of variables $x_i \in R$ that are not tight at any bound, i.e., $l_i < x_i < u_i$ for all $i \in R$. By complementing variables and bound shifting we can assume without loss of generality that $0 = l_i = x_i$ holds for every $i \notin R$. We now investigate $\mathcal{C}^{=}_{\geq}(R, R, \emptyset)$ under this assumption.

$$\begin{array}{rcl} \alpha_R^\intercal & = & w^\intercal A_{*,R} + \overline{w}_R^\intercal I_{|R|} + w_0 \pi_R^\intercal \\ \alpha_R^\intercal & = & v^\intercal A_{*,R} + \overline{v}_R^\intercal I_{|R|} - v_0 \pi_R^\intercal \\ \beta & \geq & w^\intercal b + \overline{w}_R^\intercal u_R + w_0 \pi_0 \\ \beta & \geq & v^\intercal b + \overline{v}_R^\intercal u_R - v_0(\pi_0 + 1) \\ w, v & \in & \mathbb{R}_+^m \\ \overline{w}_R, \overline{v}_R & \in & \mathbb{R}_+^R \\ w_0, v_0 & \in & \mathbb{R}_+ \end{array} \qquad (\mathcal{C}^{=}_{\geq}(R, R, \emptyset))$$

Suppose we solve $\mathcal{C}^{=}_{\geq}(R, R, \emptyset)$ and obtain a cut $\alpha_R^\intercal x_R \leq \beta$. The next theorem corresponds to Theorem 3.2 from [14] and shows how to lift this cut back to the original variable space.

Theorem 4.3 (Balas et al., 1993). *Let $\alpha_R^\intercal x_R \leq \beta$ be an optimal cut obtained from $\mathcal{C}^{=}_{\geq}(R, R, \emptyset)$ with $R \supseteq \{i \in [n] : 0 < x_i < u_i\}$. Then $\alpha^\intercal x \leq \beta$ defined below is a cut that could have been obtained from $\mathcal{C}^{=}_{\geq}([n], [n], [n])$.*

$$\alpha_j = \min\{w^\intercal A_{*,j} + w_0 \pi_j, v^\intercal A_{*,j} - v_0 \pi_j\} \quad \forall j \notin R$$

Proof. Let $(w, \overline{w}_R, w_0, v, \overline{v}_R, v_0, \alpha_R, \beta)$ be the optimal solution of $\mathcal{C}^{=}_{\geq}(R, R, \emptyset)$ and let $\alpha_j^w :=$

$w^\intercal A_{*,j} + w_0\pi_j$ and $\alpha^v_j := v^\intercal A_{*,j} - v_0\pi_j$. We distinguish two cases for all $j \notin R$:

Case 1: $\boldsymbol{\alpha^w_j \leq \alpha^v_j}$. We set $\overline{w}_j := 0$, $\underline{w}_j := 0$, $\overline{v}_j := 0$, and $\underline{v}_j := \alpha^v_j - \alpha^w_j$. For the CGLP cone $\mathcal{C}^{=}_{\geq}([n],[n],[n])$ we observe

$$\begin{array}{rclcl} \alpha_j &=& w^\intercal A_{*,j} + \overline{w}_j - \underline{w}_j + w_0\pi_j &=& \alpha^w_j + 0 + 0 \\ \alpha_j &=& v^\intercal A_{*,j} + \overline{v}_j - \underline{v}_j - v_0\pi_j &=& \alpha^v_j + 0 - (\alpha^v_j - \alpha^w_j) \end{array}$$

Furthermore, the β-constraints are not changed as the only added nonzero multiplier is $\underline{v}_j$ which is multiplied by $l_j = 0$.

Case 2: $\boldsymbol{\alpha^w_j > \alpha^v_j}$. We set $\overline{w}_j := 0$, $\underline{w}_j := \alpha^w_j - \alpha^v_j$, $\overline{v}_j := 0$, and $\underline{v}_j := 0$. As above, for the CGLP cone $\mathcal{C}^{=}_{\geq}([n],[n],[n])$ we observe

$$\begin{array}{rclcl} \alpha_j &=& w^\intercal A_{*,j} + \overline{w}_j - \underline{w}_j + w_0\pi_j &=& \alpha^w_j + 0 - (\alpha^w_j - \alpha^v_j) \\ \alpha_j &=& v^\intercal A_{*,j} + \overline{v}_j - \underline{v}_j - v_0\pi_j &=& \alpha^v_j + 0 + 0 \end{array} \tag{4.9}$$

By the same arguments as in case 1 the β-constraints are not changed.

After setting the variables $\overline{w}, \underline{w}, \overline{v}$ and $\underline{v}$ on the variables $j \notin R$ and $\underline{w}_R = \underline{v}_R = \mathbb{O}$, we conclude that $(w, \overline{w}, \underline{w}, w_0, v, \overline{v}, \underline{v}, v_0, \alpha, \beta)$ is feasible for $\mathcal{C}^{=}_{\geq}([n],[n],[n])$. $\square$

As we figured out how to lift the cuts the most natural question is whether the lifted cuts are optimal for the CGLP in in the original space. In their article Balas et al. only investigated α_1-NC, α_∞-NC, and β-NC and were able to prove that the answer is positive for the β-normalization but is negative for the two α-normalizations. To the best of our knowledge, no analysis for the lifting with respect to the other normalization constraints was done so far.

In another article [10], Balas works on a different (larger) subspace which may also involve nonbasic variables that are in some predefined set $S \subset [n]$. In general, this lifting procedure is not only useful to keep the computational efforts for solving (CGLP) small but can also applied during branch & bound in order to make locally valid cuts valid for the global problem.

4.4.3 Shrinking in Case of the Trivial Normalization Constraint

In the next paragraphs we will specialize on the trivial normalization constraint TNC for the CGLP cone $\mathcal{C}^{=}_{\equiv 0}$ defined in Section 2.4.3. We work along the lines of Bonami's article [17] and start by restating the corresponding CGLP.

$$\begin{array}{lrcl}
\min & \beta & - & \alpha^\intercal \hat{x} \\
\text{s.t.} & \alpha^\intercal & = & w^\intercal A + w_0 \pi^\intercal \\
& \alpha^\intercal & = & v^\intercal A - v_0 \pi^\intercal \\
& \beta & = & w^\intercal b + w_0 \pi_0 \\
& \beta & = & v^\intercal b - v_0(\pi_0 + 1) \\
& 1 & = & w_0 + v_0 \\
& w, v & \in & \mathbb{R}^m_+
\end{array}$$

This linear system can be simplified by subtracting the corresponding equations.

$$\begin{array}{lrcll}
\min & & & w^\intercal b + w_0\pi_0 - (w^\intercal A + w_0 \pi^\intercal)\,\hat{x} & = \; w^\intercal(b - A\hat{x}) - (\pi^\intercal \hat{x} - \pi_0) w_0 \\
\text{s.t.} & 0 & = & (w-v)^\intercal A + (w_0 + v_0)\pi^\intercal & = \; (w-v)^\intercal A + \pi^\intercal \\
& 0 & = & (w-v)^\intercal b + (w_0 + v_0)\pi_0 + v_0 & = \; (w-v)^\intercal b + \pi_0 + 1 - w_0 \\
& w, v & \in & \mathbb{R}^m_+ &
\end{array}$$

We can now substitute w_0 in the objective and eventually dualize.

$$\begin{array}{lrcl}
\min & & & (b - A\hat{x})^\intercal w - (\pi^\intercal \hat{x} - \pi_0)(b^\intercal(w - v) + \pi_0 + 1) \\
\text{s.t.} & -\pi & = & A^\intercal (w - v) \\
& w, v & \in & \mathbb{R}^m_+
\end{array} \tag{4.10}$$

Let $f_\pi := \pi^\intercal \hat{x} - \pi_0$ be the "fractional component" of $\hat{x}$ with respect to the split direction π. The dual reads

$$\begin{array}{ll}
\max & -\pi - f_\pi(\pi_0 + 1) \\
\text{s.t.} & 0 \;\le\; Ay + f_\pi b \;\le\; b - A\hat{x} \\
& y \in \mathbb{R}^n
\end{array} \tag{MLP}$$

where the variables w (resp. v) are the duals for the constraints $0 \le Ay + f_\pi b$ (resp. $Ay + f_\pi b \le b - A\hat{x}$). This so-called *Membership Linear Program* is very compact. For every original constraint it only has two constraints and they have the same LP row. For an LP solver this means that the associated slack variable must have upper *and* lower bounds which can be handled much more efficiently than two different constraints. Hence, the Membership LP is almost of the same size as the original LP relaxation.

As mentioned in Section 2.3.3 the intersection cut is already an optimal solution to the CGLP normalized with TNC. This is true if and only if we try to separate a vertex of the corresponding polyhedron. Hence, one might think that lift-and-project cuts coming from the trivial normalization constraint are useless as they only produce intersection cuts. Bonami came up with the key idea in order to make the Membership LP (and thus the TNC) usable in practice. He started cut generation via the MLP after other cuts were added. When still using the original system (the one defining P) as the basis for the MLP $\hat{x}$ is inside the associated polyhedron. In other words, this disadvantage of the TNC does not come into play if we separate rank 1 cutting planes.

4.4.4 Generalizing the MLP for Other Normalizations

We just we motivated the MLP as an efficient alternative to the CGLP only for the TNC. Our suggestion is to examine whether we can solve a nonlinear problem on the MLP polyhedron instead of solving the large CGLP with a different normalization constraint. This idea comes from the following observation. Let the cut $\alpha^\intercal x \leq \beta$ with multipliers (w, w_0, v, v_0) be feasible for the CGLP under a normalization $N(w, w_0, v, v_0, \alpha, \beta) = 1$. Then $\mu\alpha^\intercal x \leq \mu\beta$ with $\mu \cdot (w_0 + v_0) = 1$ is feasible for the CGLP under the trivial normalization. On the other hand, solutions to CGLP with TNC correspond to solutions with normalization $N(w, w_0, v, v_0, \alpha, \beta) = 1$ when we use the following (nonlinear) objective function:

$$\frac{\beta - \alpha^\intercal \hat{x}}{N(w, w_0, v, v_0, \alpha, \beta)} \tag{4.11}$$

For a linear normalization constraint $N(w, w_0, v, v_0, \alpha, \beta) = 1$ this kind of problem is known as a *linear fractional program*. By certain substitutions it can be modeled as a linear program (see [22]). Unfortunately, applying this technique here introduces the normalization itself as a constraint again. More precisely, we end up with the original CGLP formulation with $N(\cdot) = 1$ as a normalization.

Our suggestion is to pivot in the MLP but move everything else to the above context. For example, from the dual solution we can obtain a primal CGLP solution easily. The latter can then be analyzed in order to find improving directions with respect to (4.11). Such an approach already works for the SNC as we will see now.

4.4.5 The Implicit Pivoting Algorithm by Balas and Perregaard

In Section 3.2 we explained the correspondence between basic intersection cuts and lift-and-project cuts. One may consider an algorithm which solves the CGLP and can switch back and forth between the tableau representation and the CGLP representation.

Balas and Perregaard were able to move everything necessary for a simplex pivot in the CGLP (with the SNC normalization) into the context of intersection cuts. Already present in the tableau is the information *which pivots* can occur. Feasibility is not important as very often the optimal lift-and-project cuts do not correspond to basic *feasible* intersection cuts. Also simple is the computation of the CGLP objective value as it corresponds to the (scaled) violation of the intersection cut. What they contributed is mainly the computation of the CGLP reduced costs. It is used to decide whether a variable enters the basis. After this decision they use a certain function which computes the actual change in the objective for every possible leaving variable. These values are necessary because the simplex algorithm does not work by performing a complete pivot and then computing the objective value as there are too many possible pivots.

As we did not describe the exact transformation between the two cut types in Section 3.2 we also do not go into details here. Instead we refer to [13] for further reading.

4.5 Our Implementation of Lift-and-Project

For our implementation of lift-and-project we used the SCIP framework [2]. The reason for choosing SCIP was its open and good design. It is well structured with different plug-ins for specific jobs. Hence, we had very good control over the functionality because all the plug-ins have several parameters which specify their behavior. More information about SCIP can be found at its documentation page[1].

Conforming to the goal of this work we did not attempt to implement a very efficient cut separator. As the main ingredient for lift-and-project seems to be the normalization constraint we implemented all normalizations from Section 2.3 in their native form. All of them are straight-forward to implement, except for β-NC. For this constraint we added a binary indicator variable (in order to model $|\beta|$) and converted the LP to a MIP. Note that the resulting run time penalty is negligible. For the CGLP we used the original cone, i.e., $\mathcal{C}^{=}_{\geq}$. Furthermore, we treated lower and upper bounds like regular inequalities.

Significant impact on our work was imposed by the decision against an implementation which is based on bound shifting and variable complementing. One reason was that we were curious on how the β-NC performed with respect to sparsity. Our observation (see Section 4.3) that the β-normalization constraint is incompatible with bound shifting influenced this decision. Another reason is that it would have made the implementation much more complicated. Unfortunately, this missing feature has a great disadvantage, too. Cut strengthening would have been very interesting which now was not possible to implement. Though we did not carry out experiments with strengthened cuts the topic is discussed briefly in Section 5.6.

For all experiments we added cuts only in the root node. Furthermore, we did not generate cuts for all elementary disjunctions $\pi = e_k$ but only for a fixed number. In order to select this amount of cuts a priori we used a heuristic. The latter works by examining the fractionality. According to our previous experience, this heuristic is far from yielding the strongest cuts when we demand only a small number. But it chooses good candidates for giving a cut at all. More precisely, the elementary disjunctions which do not yield a negative CGLP objective value tend to have a fractionality which is almost zero or one.

As the testbed we used all instances from the MIPLIB 3.0 and MIPLIB 2003 (see [34] and [3]). For different experiments we often had to restrict ourselves to a subset of instances with a certain property, e.g. the presence of equations in the model. All problems were run on the same machine, namely one with 4 64-bit AMD Opteron 6176 SE processors, each with 12 cores, 300 GB memory equipped with an openSUSE Linux.

Chapter 5

Sparsity of Cutting Planes

This chapter subsumes the results of our main practical work on sparsity. First we give a definition which is not a hard task but has some pitfalls which one needs to avoid. In Section 5.2 we answer the question about the importance of sparsity. We present an experiment specifically designed to measure the influence of nonzero entries on the run time of a state-of-the-art dual simplex implementation. Then we move on to lift-and-project. Here we measure the actual sparsity of cuts from optimal CGLP solutions and go on to measure the sparsity that one could get by doing extra work. We finish the lift-and-project specific topic by presenting ideas of how to improve the sparsity. In order to get an impression of the theoretical limitations associated to sparse cutting planes we analyze the problem of making an LP row sparse by using other valid equations. The last section is about sparsity of other classes of cutting planes, especially the related ones (intersection and GMI cuts).

5.1 Searching for a Definition

At first glance, a definition of sparsity or density is very simple. Of course, an LP row should be called sparse if it has only a few nonzero coefficients and dense in the other case. During our search for a definition we found a one which is not suitable. In [6] the authors state

> A first realization is that such cuts must be sparse, i.e., the cuts must have many zero coefficients.

The problem arises because their polyhedral description of a MIP relaxation is in equation form, i.e.,

$$P = \{x \in \mathbb{R}^n : Ax = b \quad \wedge \quad x \geq 0\} . \tag{5.1}$$

Later, Andersen and Weismantel introduce the concept of a *zero coefficient cut* which is reasonable in the sense that those cuts are optimal with respect to a certain cut separation problem. Unfortunately, in a practical MIP, their zero coefficients do not necessarily correspond to zero coefficients in the cut that is generated. The reason is as follows. By using the equation form they introduce a slack variable for every inequality. On the one hand, the definition of sparsity is applied to the whole equation including slack variables. On the other hand, the LP solver does not introduce slack variables explicitly.

Consider, for example, a problem with several knapsacks of size 100 over distinct sets of variables.

$$\begin{array}{rcrcl} a_{1,1}x_1 & +\ldots+ & a_{1,100}x_{100} & \leq & b_1 \\ a_{2,1}x_{101} & +\ldots+ & a_{2,100}x_{200} & \leq & b_2 \\ \vdots & & \vdots & & \vdots \\ a_{m,1}x_{100m-99} & +\ldots+ & a_{m,100}x_{100m} & \leq & b_m \end{array} \tag{5.2}$$

They may be linked together by some other constraint. With help of slack variables $s_1, \ldots, s_m$ for the inequalities in (5.2) a valid cut may be $\alpha_1 s_1 + \ldots + \alpha_m s_m \leq \beta$. The latter may be considered sparse because fewer than 1% of the variables have a nonzero coefficient. In an implementation, however, the slack variables will be substituted and the resulting cut will be very dense.

Another transformation for polyhedra is the replacement of some unconstrained variable x by the difference of two nonnegative variables x^+, x^-. This also has an effect on the (relative) number of nonzeros. Hence, we should also try to avoid assumptions like $x \in \mathbb{R}^n_+$. Row transformations like copying or appending of unit rows does not influence sparsity. This allows us to use inequalities (replacing equations by two inequalities) and handle bounds explicitly, giving rise to the following definition.

Definition 5.1 (Primal Sparsity/Density). *Let $P = P^{\leq}(A, b) \subseteq \mathbb{R}^n$ be a polyhedron. The* absolute (primal) density *(resp.* sparsity*) of an inequality $A_{j,*}{}^{\intercal} x \leq b_j$ is the value $d := |\mathrm{supp}\,(A_{j,*})|$ (resp. $n - d$). Its* relative (primal) density *(resp.* sparsity*) is defined as $\delta := d/n$ (resp. $1 - d/n$).*

Often, cutting planes are derived from a linear row combination $\lambda^\intercal Ax \leq \lambda^\intercal b$ for $\lambda \in \mathbb{R}^m_+$. In order to cut off something, the latter is strengthened which often does not change the sparsity. For example, Chvátal Gomory (CG) cut generators simply round down the right-hand side. This consideration leads to another definition.

Definition 5.2 (Dual Sparsity/Density)**.** *Let $P = P^{\leq}(A, b) \subseteq \mathbb{R}^n$ be a polyhedron. The* absolute dual density *(resp.* sparsity*) for a cut derived from row multipliers $\lambda \in \mathbb{R}^m_+$ is the value $d^* := |\operatorname{supp}(\lambda)|$ (resp. $m - d^*$). Its* relative dual density *(resp.* sparsity*) is defined as $\delta^* := d^*/m$ (resp. $1 - d^*/m$).*

The motivation for the notion of dual sparsity is based on the assumption that the original problem is sparse. In this case, a row combination with only few nonzero multipliers implies a sparse cutting plane as well.

Remark 5.1. *When modeling $a^\intercal x = b$ in $P^{\leq}(A, b)$, the two inequalities $a^\intercal x \leq b$ and $-a^\intercal x \leq -b$ are used. Fortunately, this does not influence the absolute dual sparsity because for cut derivation only one of the two associated multipliers has to be used.*

5.2 Effects of Sparsity

So far we only *claimed* the importance of sparsity or cited other authors. It is now time to prove that sparsity is indeed a relevant property. The major claim is that dense cuts slow down the underlying LP solver. As the latter is called once for each B&B node it only makes sense to measure this slowdown for a complete (or at least large) exploration of the search tree. Hence, a potential experiment would be to solve an instance multiple times with different sparsity. Of course, making a row sparser is relatively hard but making it dense is not a tough problem assuming the presence of several equations. Unfortunately, a MIP solver is not very robust with respect to changes in the LP. Even just adding a valid equation to one of the inequalities may result in a different optimal relaxation solution, different generated cutting planes, a different branching decision, or almost any other decision which is made during branch & bound. Therefore we carefully devised the following experiment to measure the claimed slowdown effect.

We ran CPLEX with default settings. After the presolve phase we tried to create a dense equation $\alpha^\intercal x = \beta$ by combining all present equations of the presolved problem in a certain way. We discarded this problem instance for the experiment if $\alpha^\intercal x = \beta$ had a relative primal density of less than 25 %. Note that this is also true if no equations are present.

We proceeded with the regular solving process except that we hooked Algorithm 5.1 into the branching decision algorithm of CPLEX.

Algorithm 5.1 Pseudocode for Densification Experiment

```
1. Get the current optimal basis $B$ from the node LP.
2. For $d = 0, \dots, 9$, carry out Steps 3, ..., 6.
3. Copy LP to LP′ and apply the CPLEX branching steps (bound tightening) to LP′.
4. Add $\alpha^\intercal x = \beta$ to the first $d$ rows of LP′.
5. Feed $B$ as a warm-start basis into LP′.
6. Solve LP′ with the dual simplex method. Measure the number of simplex iterations
   (pivot steps) and the solving time.
```

Remark 5.2. *The set B in Algorithm 5.1 need not index a basis for* LP′ *because adding of rows may result in a zero-row, although it is highly unlikely. This did not not happen in our experiments, i.e., all bases B were primal infeasible and dual feasible in Step 6.*

We now present the results of this experiment. In Figure 5.1 we depicted the measured relationship between the number of "densified" rows and the resulting speed of the dual simplex implementation. A correlation is not only visible but also strong. On the one hand adding 10 cuts of a certain family is very typical. On the other hand a case where adding 10 cuts implies a reduction of the branch & bound tree by 20 % is harder to find. The latter would be necessary to compensate the slowdown imposed by 10 very dense cuts.

A second diagram shows almost the same relationship, except that we replaced the number of "densified" rows by the average number of nonzeros in the LPs. In order to make the results comparable we took relative values. For every instance we used the values for the unmodified LP as the point of reference.

We close this section by concluding that sparsity is indeed a very important property of cutting planes. After defining sparsity and showing its importance we can then turn to the main topic of this work.

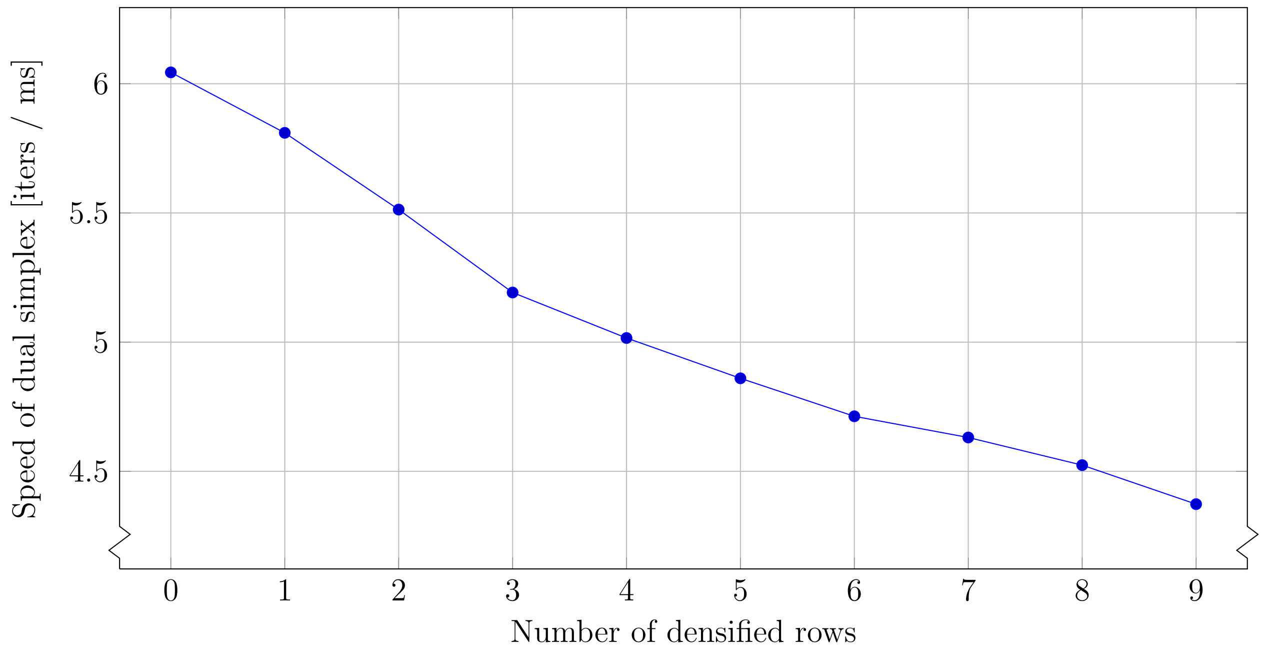


Figure 5.1: Simplex speed for densified MIPs.

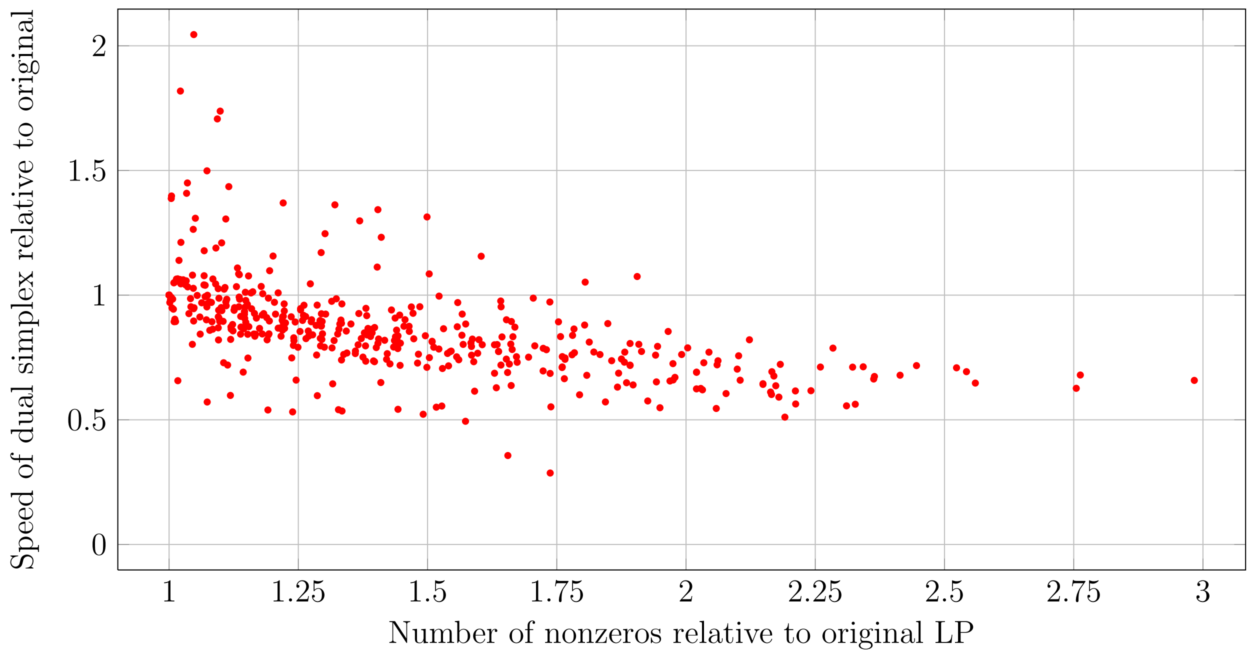


Figure 5.2: Relative simplex speed depending on the number of nonzeros.

5.3 Measuring Sparsity of Lift-and-Project Cuts

In this section we present the main sparsity results for lift-and-project cutting planes. For this we carried out two experiments with all mentioned normalization constraints. As the notion of cut rank is very important in theory and practice we also carried out all experiments for rank 1 cuts and for cuts with arbitrary rank. The latter means that in a cut round all previously generated cuts are considered as regular inequalities and may be taken into account for new cuts. In the first experiment we attempted to generate lift-and-project cuts in 10 rounds, generating 100 cuts every time. We then measured basic sparsity-properties. For a subset of small problems we analyzed the sparsity in the CGLP with a lot more computational power. The goal was to find out how sparse the cuts *can* become compared to the first (optimal) solution.

5.3.1 Measuring the Actual Sparsity

To measure the sparsity of lift-and-project cutting planes we carried out the following experiment. For every problem instance from our testset we ran SCIP with default settings, except that we disabled other cut generators and branch & bound. We allowed 10 rounds of 100 cuts, i.e., 10 times the separator was allowed to generate 100 cuts before SCIP solved the enhanced LP relaxation to produce a new $\hat{x}$. The following data was gathered for every cut in every instance of our testbed.

- Cut round
- Absolute primal density
- Absolute dual density (average of $\text{supp}(w)$ and $\text{supp}(v)$)
- Number of cancellations (number of zero coefficients with nonzero dual support)

From the absolute primal/dual density we obtained the corresponding relative values. For every round we only used the arithmetic mean to compute average values. This makes sense because then the average density corresponds to the sum of all generated nonzeros, divided by the number of generated cuts. Furthermore, the overall number of nonzeros is a measure for the additional effort that the LP solver has to spend (see Figure 5.2 on page 51). Eventually, all instances were considered to compute averaged (arithmetic and geometric mean) values. The experiments were repeated for all normalizations and rank conditions (rank 1 or arbitrary rank). The details for every instance can be found in the appendix on page 72.

Type	Rank	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
Abs. primal density	$= 1$	1,361.2	27.4	101.6	184.3	255.6	209.1	285.2
(arithmetic mean)	≤ 10	1,373.3	39.1	189.5	196.6	252.9	238.9	440.3
Abs. primal density	$= 1$	460.8	10.4	37.0	29.1	30.7	30.5	48.0
(geometric mean)	≤ 10	487.1	19.7	65.0	60.2	78.7	64.7	130.4
Abs. dual density	$= 1$	1,194.2	155.2	105.8	92.5	94.6	97.0	242.2
(arithmetic mean)	≤ 10	1,212.3	212.7	388.1	95.0	114.1	112.3	355.1
Abs. dual density	$= 1$	386.9	29.9	30.0	19.2	19.8	20.2	46.6
(geometric mean)	≤ 10	410.8	60.1	77.8	32.5	39.6	35.3	103.2

Table 5.1: Statistics of absolute density of lift-and-project cuts.

Type	Rank	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
Rel. primal density	$= 1$	86.6	6.4	16.1	17.1	18.7	17.6	21.9
(arithmetic mean)	≤ 10	89.7	9.2	30.3	24.0	30.7	25.3	36.3
Rel. primal density	$= 1$	82.2	1.9	6.6	5.2	5.5	5.4	8.6
(geometric mean)	≤ 10	86.9	3.5	11.6	10.7	14.0	11.5	23.3
Rel. dual density	$= 1$	27.7	6.2	5.8	3.7	3.8	3.8	7.3
(arithmetic mean)	≤ 10	27.4	9.0	22.0	4.6	5.2	4.8	10.3
Rel. dual density	$= 1$	23.2	1.8	1.8	1.2	1.2	1.2	2.8
(geometric mean)	≤ 10	23.3	3.3	4.6	1.8	2.2	1.9	5.7

Table 5.2: Statistics of relative density of lift-and-project cuts in %.

Type	Rank	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
Number of cancelations	$= 1$	6.7	67.4	12.0	1.7	9.8	3.2	25.9
(arithmetic mean)	≤ 10	5.8	76.2	9.7	1.2	15.7	2.8	43.0
Number of cuts	$= 1$	239.9	367.2	82.2	404.5	323.4	380.0	360.5
(arithmetic mean)	≤ 10	301.6	363.0	71.4	410.9	329.6	387.3	400.4

Table 5.3: Sparsity-related statistics of lift-and-project cuts.

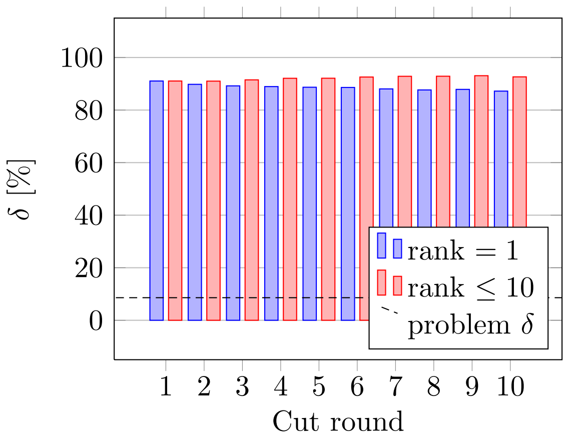
We start with statistical values which are averaged over the complete instance set. In Table 5.1 we present the absolute values, in Table 5.2 the relative values, while Table 5.3 lists some additional statistics.

A first observation is that the average number of generated cuts is roughly the same for all normalizations except for β-NC. With the latter constraint our separator is only able to generate 20 % of this amount.

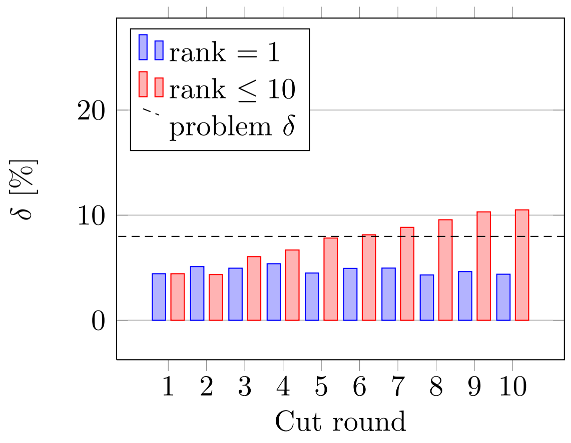
Looking at the primal and dual sparsity values we see that in general the α_∞-NC normalization generates very dense cuts while the α_1-NC normalization is the winner when it comes to sparsity. At least for the former effect we can present an explanation. Let $\alpha_i^\intercal x \leq \beta_i$ for $i = 1, 2$ be two sparse cuts which are feasible to the CGLP with α_∞-NC and assume that their coefficient vectors α_i have distinct support. Then the cut $(\alpha_1^\intercal + \alpha_2^\intercal)x \leq \beta_1 + \beta_2$ is a valid cut and the objective value (cut violation) is the sum of the two original objective values. Hence, one can think of α_∞-NC as a normalization which tries to mix different cuts such that their sum "fits" into the $[-1, 1]$-bounds. Such a mix is typically very dense. An interesting question arises because we don't know how the α_p-NC for $p = 2$ performs as the associated CGLP truncation is nonlinear.

The other four constraints give almost the same results whereas TNC generates relatively dense cuts, too. We like to mention that our proposed constraint DNC yields the sparsest cuts when we restrict ourselves to the multiplier-based normalizations. When comparing rank-1 cuts with cuts of arbitrary rank we can conclude that cuts of higher rank are denser than their rank 1 counterparts. The number of cancellations is very interesting as the α_1-NC which produces the sparsest cuts also showed the largest number of cancellations. The reverse observation is possible for α_∞-NC. The fewest cancellations can be observed in case of the multiplier-based normalization constraints. This implies that if such a cut is sparse then the sparsity must come from the sparsity of the incorporated inequalities. Hence, we expect a strong correlation between primal and dual sparsity for these normalizations.

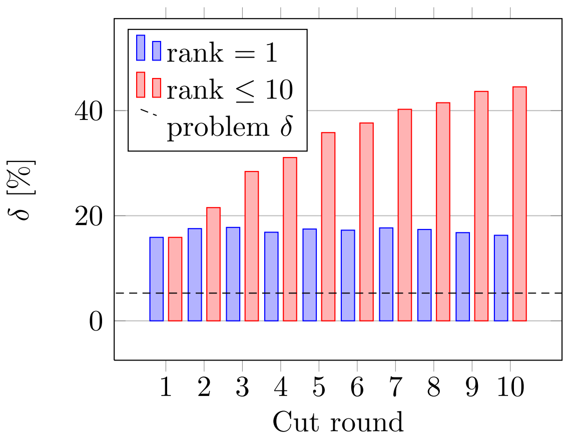
After measuring sparsity one may want to know which cuts are sparse and which are dense. Using our experiment data we investigated the question how sparsity is affected by the cut round. This question is especially interesting with respect to our claim that higher rank cuts tend to be denser in general. The 6 diagrams in Figure 5.3 show the results answering exactly this question. We have no results for the β-NC as there was not a single instance where 10 cut rounds could be carried out. The reason is that not enough cuts (if any) were able to cut off the current relaxation solution much and the MIP solver decided to stop generating more cuts. This corresponds to our first observation about the number of generated cuts.



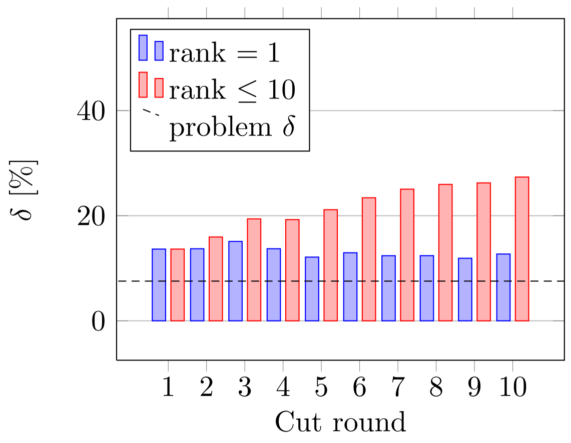
(a) α_∞-NC



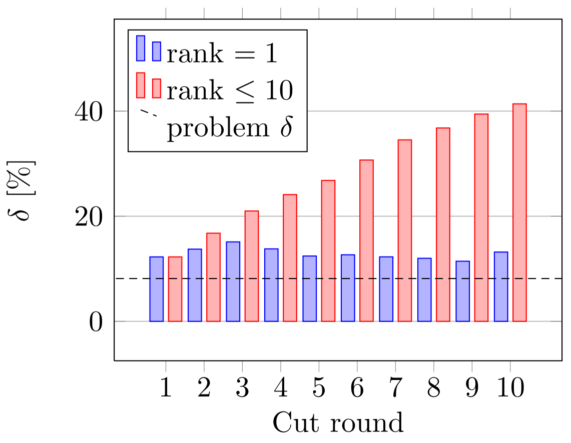
(b) α_1-NC



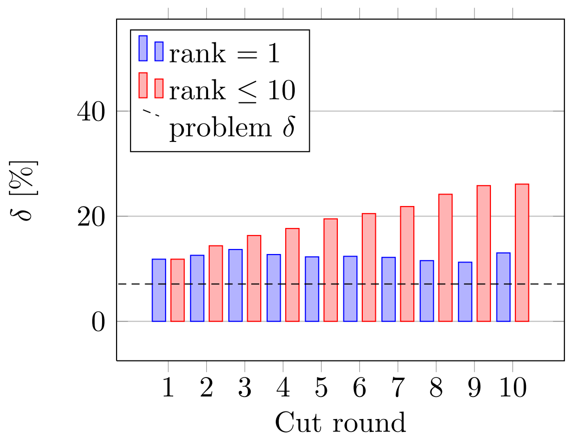
(c) TNC



(d) SNC



(e) ENC



(f) DNC

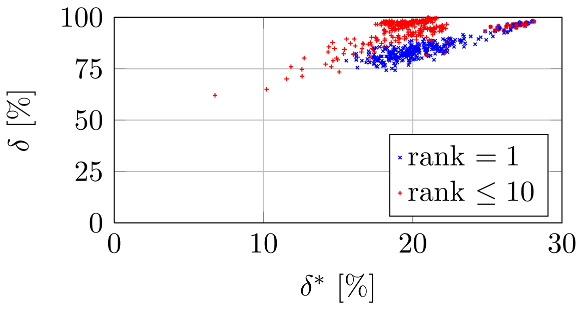
Figure 5.3: Arithmetic mean of relative primal density by rounds.

In order to take into account as many instances as possible we included those for which the current normalization generated at least one cut in round 10. In other words, the depicted values come from slightly different instance sets although they overlap at about 80 %. Hence, their specific values are not comparable. But every single diagram tells us a lot about the relationship between sparsity and the cut round. First, we verify that the bars for round 1 have the same height which is reasonable as the generation in round 1 coincides for rank 1 and arbitrary-rank experiments. Second, the density for cuts with arbitrary rank increases with the cut round. For rank 1 cuts, the average primal cut density does not change significantly for all normalizations except for α_∞-NC. Interestingly, it decreases for this normalization but we don't have any idea for a reason.

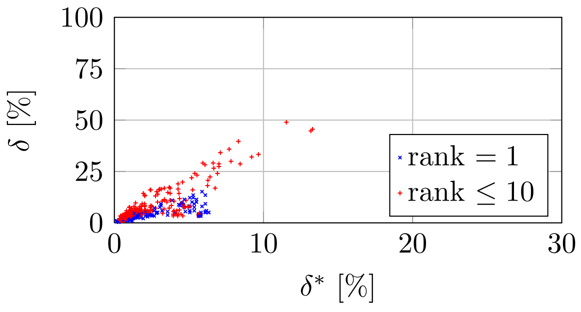
In Section 5.1 we introduced the concept of dual sparsity. The idea comes from arguments in literature about the sparsity property of the SNC which is claimed to stem from few selected multipliers. Our hope was to find out for which (if any) normalization constraint there is a correlation between primal and dual sparsity.

Figure 5.4 shows results for this correlation. We looked at the problem `pp08aCUTS` because for this instance our separator was able to generate a decent amount of cuts regardless of the normalization constraint (see Tables A.11 and A.12 for details). In the diagram we painted a dot for every generated cut. The coordinates of this dot come from the relative dual and primal sparsity of this cut, respectively. The number of cuts having the same primal *and* dual sparsity is not depicted.

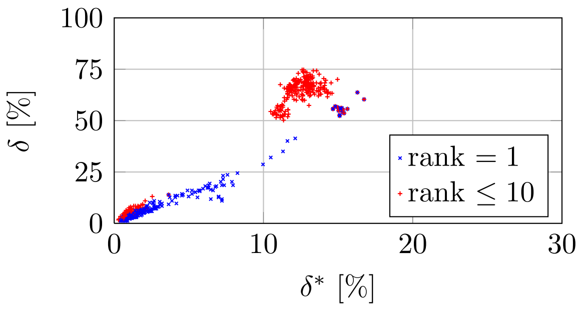
As expected, α_∞-NC looks horrible when the goal is sparsity. It does not need many multipliers, though. The centers of the "clouds" are at 20 % dual density but the cuts have a density of more than 75 %. The diagrams for α_1-NC, SNC, ENC and DNC look very similar. We observe a linear correlation where the slope for rank 1 cuts is smaller than the slope for cuts of arbitrary rank. The conclusion is clear: Having the number of multipliers fixed, higher rank cuts are denser than rank 1 cuts. This seems to be true for rank 1 cuts from TNC while it looks different for the higher rank cuts. Here we see two "clouds" where a small amount is very sparse ($\approx 10\,\%$) and a larger portion is dense ($50 - 75\,\%$). This time we were able to produce a figure for β-NC as well. Again, the higher rank cuts show an interesting effect. Almost all "hide" at the coordinates $(1.0, 1.0)$ which means that such a cut incorporated *every* constraint which in turn resulted in a fully dense inequality.



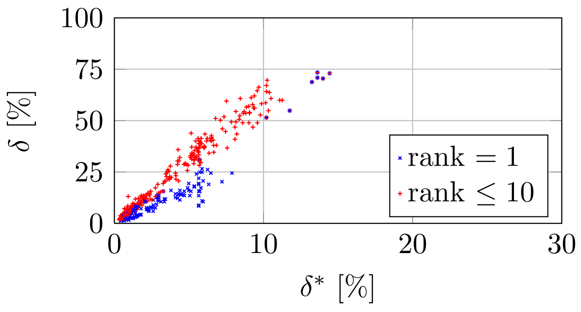
(a) α_∞-NC



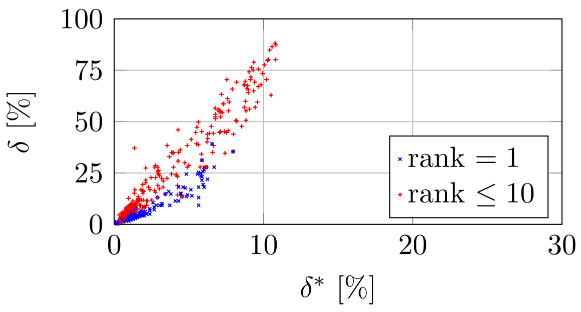
(b) α_1-NC



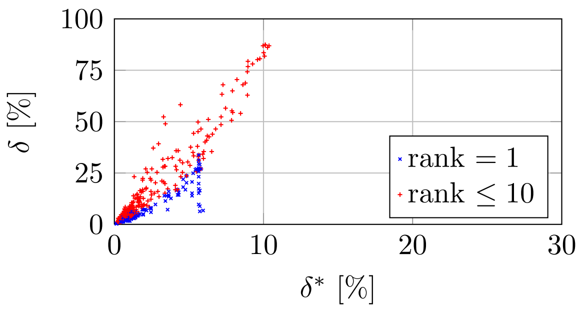
(c) TNC



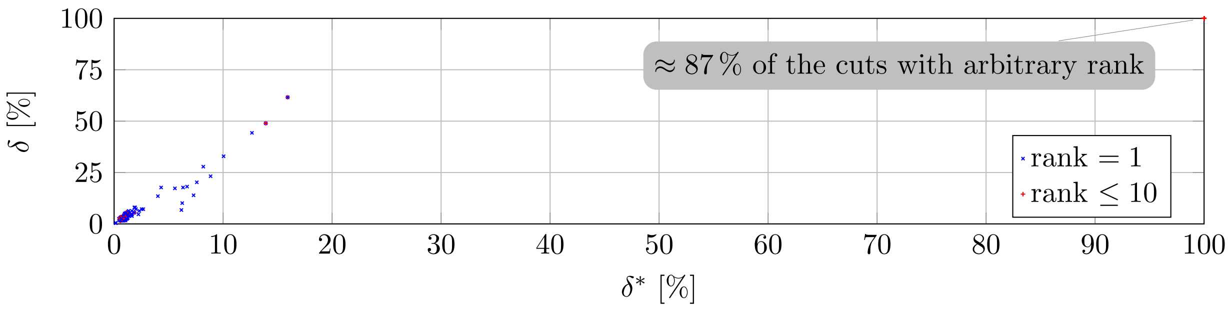
(d) SNC



(e) ENC



(f) DNC



(g) β-NC

Figure 5.4: Primal vs. dual density in instance `pp08aCUTS`.

We finish the discussion with a summary of the conclusions we made.

- There is a correlation between dual and primal sparsity. For several normalizations it is almost linear.
- Rank 1 cuts are preferable when the goal is sparsity.
- β-NC does not produce many cuts which are dense once higher rank cuts are permitted.
- α_∞-NC produces dense cuts already in the first round.
- α_1-NC produces very sparse cuts.
- We slowly loose control of sparsity in case of higher rank cuts derived with the TNC.
- The remaining normalizations produce cuts with a decent sparsity.

5.3.2 Measuring the Possible Sparsity

This section is dedicated to the question how sparse lift-and-project cutting planes *could* be in theory. For this we solved a series of MIPs for every cut. As those problems are extensions to the CGLP this experiment is very time-consuming. Although we allowed every instance to be processed a complete week (on a single CPU) we were only able to carry out the experiment for the 12 small instances `pp08aCUTS`, `mas74`, `p0201`, `stein45`, `pk1`, `rout`, `misc03`, `p0282`, `stein27`, `misc07`, `bell5`, and `timtab1`. Nevertheless, we obtained several interesting results.

The purpose of our MIP was to investigate the structure of the CGLP with respect to sparsity of the generated cuts. Building on top of the CGLP we added binary variables x_j for $j \in [n]$ which are supposed to be 1 if the cut coefficient α_j is nonzero. The absolute density is then just the sum of all x variables.

$$-Mx_j \leq \alpha_j \leq Mx_j \tag{5.3}$$

Equation 5.3 shows how the variables are linked together. We call the CGLP together with all these equations the *augmented CGLP*. This model is correct as long as M is a sufficiently large constant. In general, these so-called *big-M* formulations often introduce numerical difficulties. In fact we cannot choose M to be extremely large as our MIP solver uses floating point arithmetic. It “thinks” that a number i is zero if $|i| < \varepsilon$ for some small $\varepsilon > 0$. In our specific case it could be that $|x_j| < \varepsilon$ even though $|\alpha_j| > \varepsilon$, i.e., the solver claims that α is a zero coefficient even though it is not.

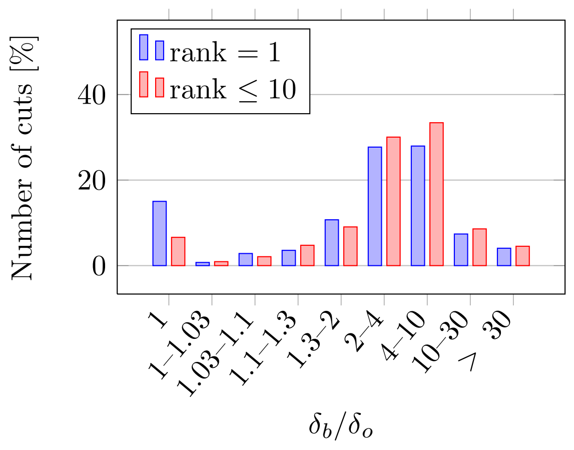
In order to reduce the number of mistakes to an acceptable level we did two things. First, we decreased ε from its default value of 10^{-9} to a value of 10^{-12}. Second, we solved the LP part a second time after fixing α_j to zero for all j with $|x_j| < \varepsilon$ in the MIP solution. Of course this way there is still no guarantee for correctness, but we reduced the number of cuts to a minimum for which the solver incorrectly pretended sparsity.

The just described MIP was used to find the minimal CGLP objective value (best cut violation) under an absolute sparsity constraint. Algorithm 5.2 provides the details of this experiment for every cut.

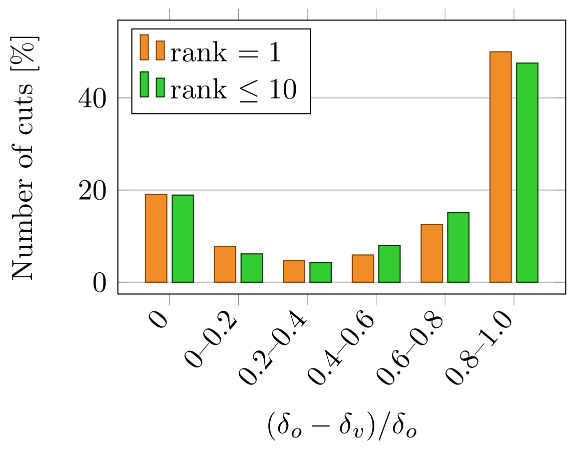
Algorithm 5.2 Pseudocode for Measuring the Lift-and-Project Sparsity

```
1. Let $d$ be the *base* density of the cut obtained by solving the CGLP with violation $v_d$. Construct a map $\varphi : [d]_0 \mapsto \mathbb{R}_+$ with $\varphi(0) := 0$ and $\varphi(d) := 1$.
2. For $m = 1, \ldots, d-1$, carry out Steps 3, ..., 4.
3. Bound the absolute density of the cut by $m$ (from above) via the $x$ variables. Solve the augmented CGLP with a time limit of $60\,s$.
4. Solve the CGLP with those $\alpha_j$ fixed to zero for which $x_j = 0$ in the optimal solution of the augmented CGLP. and obtain the cut violation $v_m$. Let $\varphi(m) := v_m / v_d$ be the relative cut violation.
5. By $\delta_b := d/n$ we denote the relative density of the cut obtained from the CGLP. By $\delta_o := \min\{m : \varphi(m) = \varphi(d)\}/n$ we denote the optimal relative density. Finally, by $\delta_v := \min\{m : \varphi(m) > 0\}/n$ we denote the first valid density, i.e., the smallest density for which we obtain an inequality which cuts off $\hat{x}$.
```

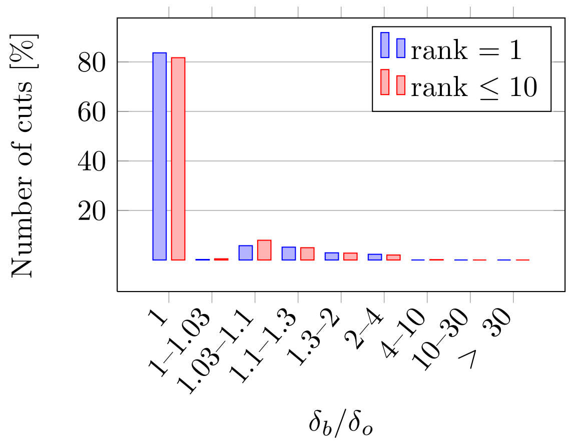
The computational effort for this procedure was especially large for the α_∞-NC due to the high base density. Although we restricted the solving time to 60 seconds per MIP, a base cut with several thousand nonzeros implied a huge number of MIPs to solve. We tried to reduce it by warm-starting each run with the previously found solution. This helped a lot, especially when the optimal density had been found because then all subsequent runs had no integrality gap to close.



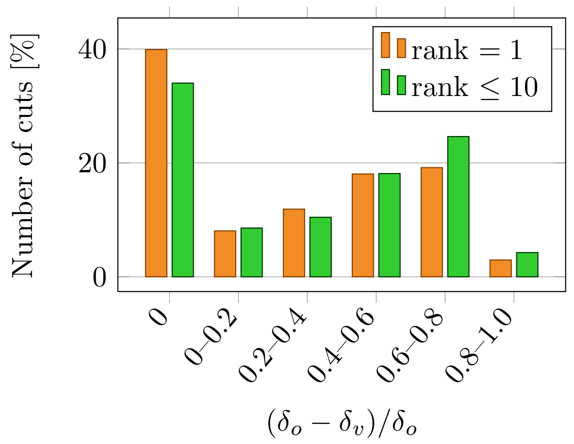
(a) Potential optimization for α_∞-NC



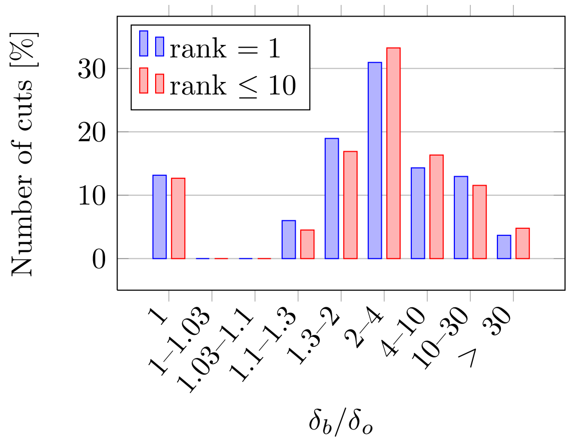
(b) Nonoptimal density improvement for α_∞-NC



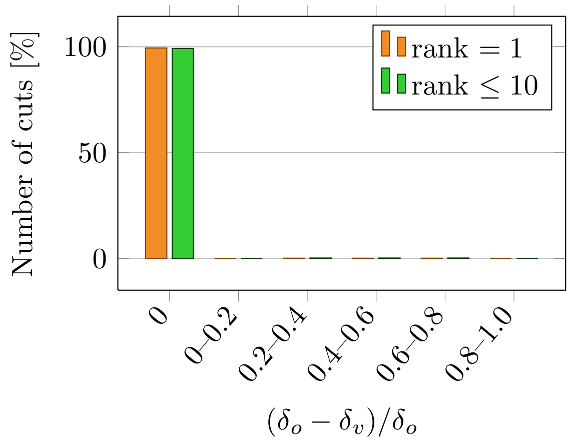
(c) Potential optimization for α_1-NC



(d) Nonoptimal density improvement for α_1-NC



(e) Potential optimization for β-NC



(f) Nonoptimal density improvement for β-NC

Figure 5.5: Distribution of δ_b and δ_v with respect to δ_o for the α- and β-normalizations.

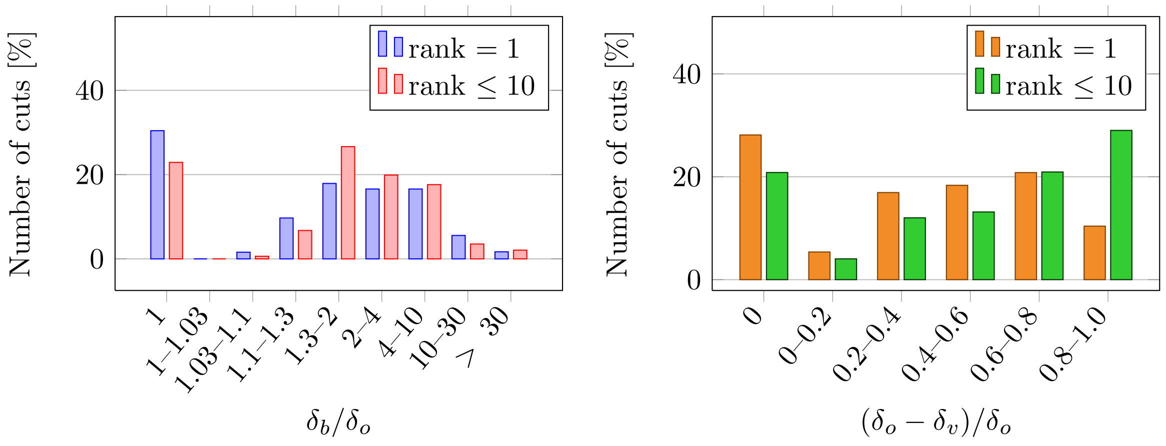
We now turn to the results of this experiment. Figures 5.5 – 5.7 consist of pairs of diagrams. For each normalization we depicted the relationship between δ_b and δ_o (blue and red) as well as that between δ_v and δ_o (orange and green). They display the relative amount of generated cuts with a certain property.

We focus on the first and observe that all 7 pictures have almost the same shape. There is a more or less large bar for $\delta_b/\delta_o = 1$. It means that the cut from the CGLP was already optimally sparse. The remaining ranges always show a distribution similar to the Gaussian distribution. Here we are interested in the height and width of this "hill".

We already know that α_1-NC and the SNC-related normalizations generate cuts of decent sparsity. It is thus no surprise that the majority of the generated cuts has already optimal sparsity. For all these normalizations the rank 1 cuts perform better than cuts of arbitrary rank. This is very clear for α_∞-NC and TNC.

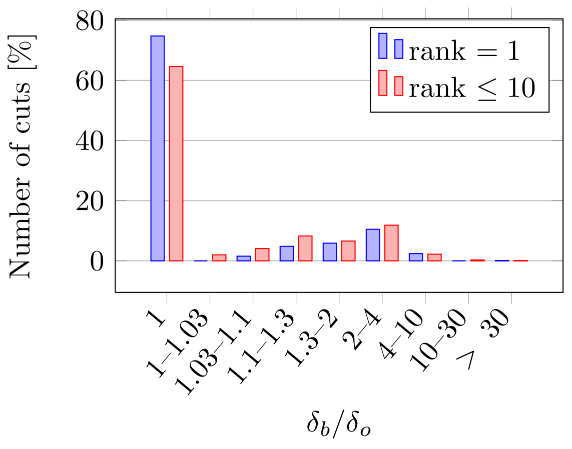
Except for our previous winner, the α_1-NC, all distributions have their maximum for the nonoptimal cuts in the range of $1.3 \cdot \delta_o \leq \delta_b \leq 10 \cdot \delta_o$. For example, α_∞-NC has several cuts which are 500 % too dense. β-NC is slightly and the TNC already much better. Note that most of the nonoptimal cuts from α_1-NC are only at most 20 % too dense.

The pictures on the right-hand side show the case where we have cuts which are sparser than δ_o by allowing a cut violation which is smaller than that of the optimal CGLP solution. Note that we still require cuts, i.e., the violation must be strictly greater than zero.

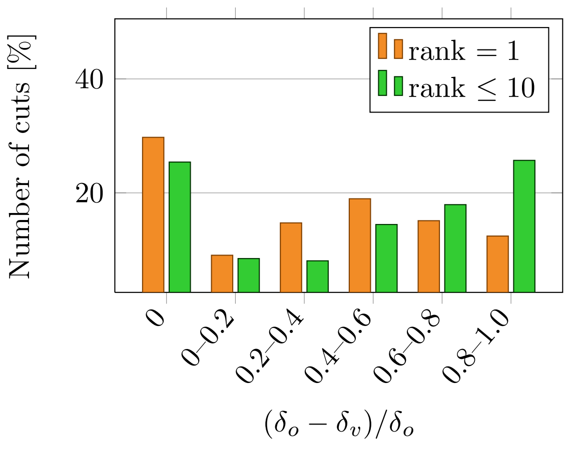


(a) Potential optimization for TNC (b) Nonoptimal density improvement for TNC

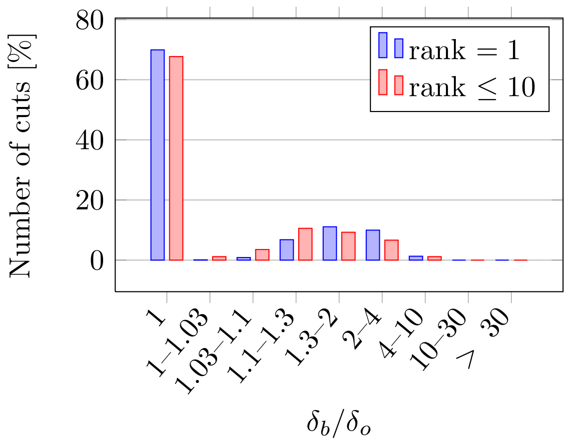
Figure 5.6: Distribution of δ_b and δ_v with respect to δ_o for the trivial normalization.



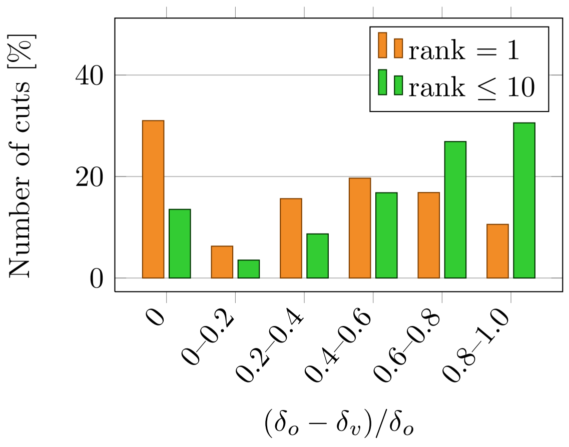
(a) Potential optimization for SNC



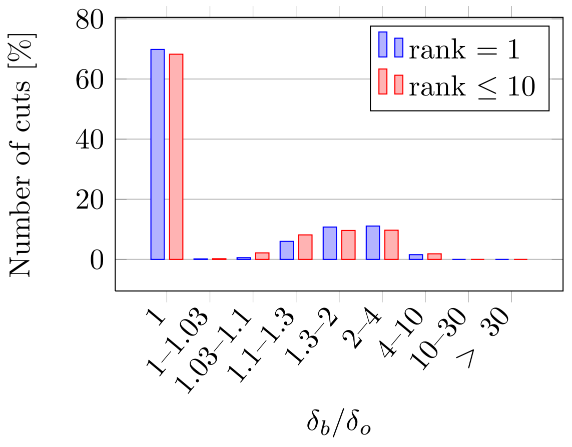
(b) Nonoptimal density improvement for SNC



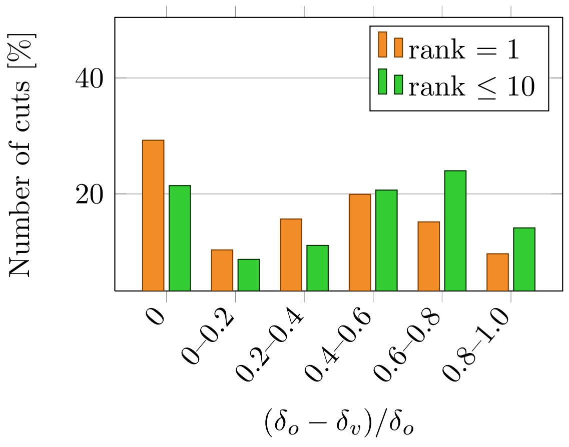
(c) Potential optimization for ENC



(d) Nonoptimal density improvement for ENC



(e) Potential optimization for DNC



(f) Nonoptimal density improvement for DNC

Figure 5.7: Distribution of δ_b and δ_v with respect to δ_o for the multiplier-based normalizations.

Again, the 0-bar is very special. Here it shows the relative amount of cuts for which we cannot gain sparsity by allowing a nonoptimal violation. On the contrary, the rightmost bar displays the cuts with almost arbitrary small density after sacrificing optimality.

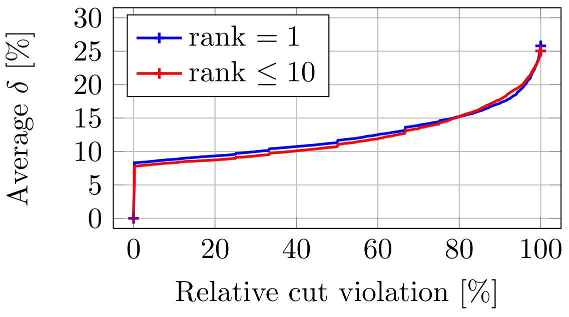
Except for β-NC, all normalizations allow to generate the same set of cuts in the first round. Of course, they have different objective values due to different scaling. But as we allow any negative objective values now, the sparse cuts must appear even for normalization constraints which tend to yield dense cuts. This observation manifests itself in the diagrams, too:

On the one hand, the optimal cuts from the α_1-NC are known to be sparse already. Hence, sacrificing optimality does not give sparser cuts. This implies a very tiny rightmost bar. On the other hand, the optimal cuts from the α_∞-NC are typically dense and the rightmost bar is very huge, collecting all the sparse cuts.

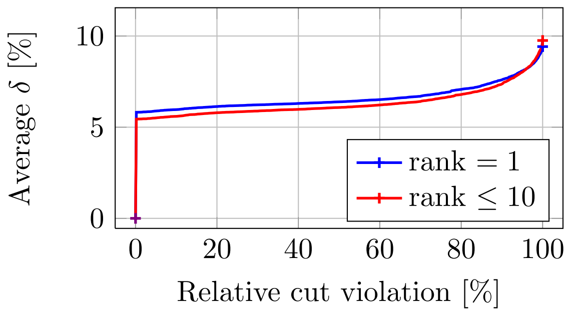
A very interesting artifact can be seen in Figure 5.5 (f). Here the violation seems to be completely unrelated to sparsity. Looking for the source of this anomaly we found the following effect: In Step 3 of Algorithm 5.2 the augmented CGLP often pretends that it has found a cut of specified density with optimal violation. Then in Step 4 the verification via the CGLP without the x variables fails, returning a violation of zero. We have not found a good reason why this happens *every* time, except for general numerical trouble with the β-NC. Of course, the diagram itself is simply wrong but we decided to keep it because of the interesting anomaly.

As we observed in Section 5.3.1 already, the TNC shows different results depending on the rank restrictions. Again, the fact that rank 1 TNC cuts are sparser than their counterparts with arbitrary rank is clearly visible. The other multiplier-based normalizations behave in the same way except that the DNC, being only a bit sparser than the other two, has very few cuts with a small δ_v.

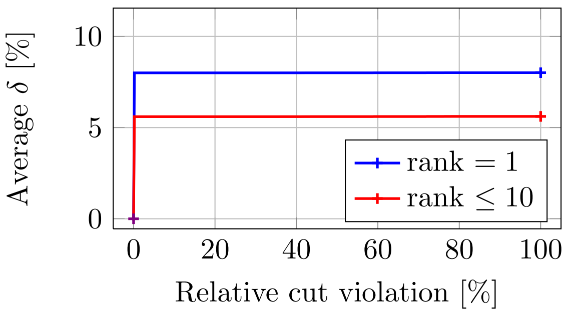
We now turn to Figure 5.8 which is the last in this section. The diagrams give the average cut density for every fixed violation (relative to the CGLP violation). It shows an "average graph" for φ^{-1}, i.e., the inverse map of φ (see Algorithm 5.2), averaged over all cuts. All graphs are monotone increasing because a higher allowed density will never result in a smaller optimal cut violation. The interesting point is that the graphs are convex for a nonzero violation. From this we can conclude that if one desires very sparse cuts it may be helpful to relax the requirement of an optimal violation to a value of 90 % or 95 %. On the contrary, relaxing the violation too much doesn't help a lot.



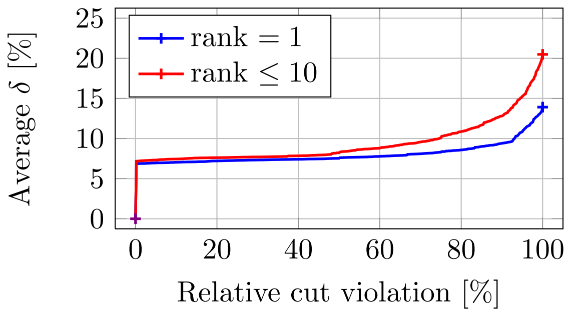
(a) α_∞-NC



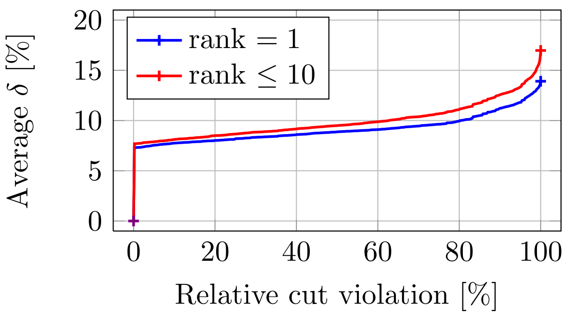
(b) α_1-NC



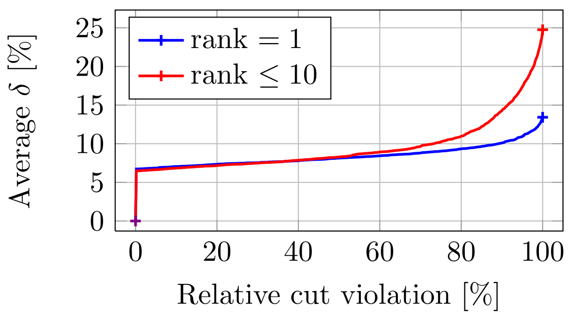
(c) β-NC



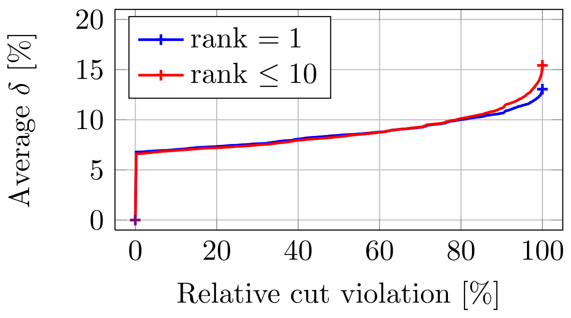
(d) TNC



(e) SNC



(f) ENC



(g) DNC

Figure 5.8: Possible sparsity when permitting nonoptimal cut violations.

Finally, we present our conclusions about the topic of this section.

- The *big-M* model can be used to measure or control sparsity in many cases. It does not work for the β-NC though.
- The α_1-NC, DNC, SNC, ENC natively produce cuts which are often optimally sparse. This is even true if we allow nonoptimal cut violations.
- The converse holds for the horrible normalization α_∞-NC. Here, the optimal density is a fraction of the native density.
- The situation for the TNC is moderate. At least a third of the cuts is already optimal, but the others are quite too dense.

5.4 Improving Sparsity of Lift-and-Project Cuts

The next step after measuring the current state and the possibilities is of course the work on practical improvements. The augmented CGLP we used in Section 5.3.2 is not a practical method as solving it often takes too long. A second problem is that a potential improvement method must also be usable in one of the setups with a reduced size CGLP. For example, the cut coefficients are present as variables in the implicit algorithm by Balas and Perregaard. This is also true for the MLP.

A first approach is an indirect control via constraints. In the previous sections of this chapter we proved that the normalization constraints have a heavy impact on sparsity. Choosing the right one is nontrivial because sparsity is not the only goal. Further ones are the strength of the cuts (the ability to reduce the size of the B&B tree significantly), diversity (cutting in different directions), and other numerical properties like a small range of the coefficients.

Another idea complements the above one. The CGLP is a very large and degenerate LP. Hence requiring optimality reduces its size only marginally. We suggest to perform local improvements by pivoting in the optimal face. The problem of finding a sparse vertex of a polyhedron is $\mathcal{NP}$-hard which will become clear in the next section. Nevertheless, a greedy improvement strategy for CGLP can always be applied.

With our experiments we showed that for the most used normalization (SNC) the cuts are often optimal when we demand optimal violation. Our conjecture is that the second idea won't help too much. Unfortunately, an implementation of such a local improvement heuristic was not possible due to time limitations for this work.

5.5 Improving Sparsity of any LP Row

Up to now we specifically investigated lift-and-project cutting planes and aspects how to generate sparse cutting planes. This section is dedicated to the more general problem of improving sparsity of any LP row.

5.5.1 The Row Sparsification Problem

For a given linear inequality system $Ax \leq b$ and another inequality $\alpha^\intercal x \leq \beta$ we don't have much choice in how to change the single inequality. Any nonnegative linear combination $\lambda^\intercal Ax \leq \lambda^\intercal b$ can be added to $\alpha^\intercal x \leq \beta$. But as we do not want to relax the single inequality (or in the context of cut separation generate a weaker cut) we can only use those inequalities that are *always tight*. That is, not only $\lambda^\intercal Ax \leq \lambda^\intercal b$ must be true for every $x \in P^{\leq}(A, b)$, but also $\lambda^\intercal Ax = \lambda^\intercal b$. In other words, we are allowed to use equations for row sparsification. To explore the problem we state it in a formal fashion.

Problem 5.3 (SPARSELINEARROW). *Given a matrix $A \in \mathbb{Q}^{m\times n}$, a vector $a \in \mathbb{Q}^n$ and an integer $k \in [n]$, the task of the problem* SPARSELINEARROW *is to decide whether there exists $\lambda \in \mathbb{Q}^m$ such that*

$$|\mathrm{supp}\,(a + \lambda^\intercal A)| \leq k.$$

The associated minimization problem attempts to minimize k.

5.5.2 Complexity of Row Sparsification

We now investigate the computational complexity of Problem 5.3. We state another related problem of which we know that it is $\mathcal{NP}$-complete (see Problem MP5 in [29]).

Problem 5.4 (SPARSESOLUTIONLINEARSYSTEM). *Given a matrix $A \in \mathbb{Z}^{m\times n}$, a vector $b \in \mathbb{Z}^m$ and an integer $k \in [n]$, the task of the problem* SPARSESOLUTIONLINEARSYSTEM *is to decide whether there exists $x \in \mathbb{Q}^n$ such that*

$$|\mathrm{supp}(x)| \leq k \text{ and } Ax = b.$$

In order to show that Problem 5.3 is $\mathcal{NP}$-complete as well we need to give a polynomial time reduction which reduces Problem 5.4 to Problem 5.3.

Theorem 5.3. *The problem* SparseLinearRow *is $\mathcal{NP}$-complete.*

Proof. First we describe a polynomial-time algorithm which accepts any input for Problem SparseSolutionLinearSystem and outputs input for SparseLinearRow.

Let $A \in \mathbb{Z}^{m \times n}$, $b \in \mathbb{Z}^m$ and k be given. With a carefully implemented Gaussian elimination (see [25]) or any polynomial-time linear programming algorithm, find an inhomogeneous solution x^* and a basis $\{y_1, \dots, y_d\}$ for $\ker A = \{x \in \mathbb{R}^n : Ax = 0\}$ (with $d \leq n$). Let $Y \in \mathbb{Q}^{d \times n}$ be the matrix consisting of rows y_j. Give (Y, x^*, k) as input to SparseLinearRow.

We finish the proof by showing the following equivalence:

$$\begin{aligned}
& (A, b, k) \text{ is a ``yes''-instance for SparseSolutionLinearSystem} \\
\Leftrightarrow \quad & \exists x \in \mathbb{Q}^n : Ax = b \ \wedge \ |\text{supp}(x)| \leq k \\
\Leftrightarrow \quad & \exists x^* \in \mathbb{Q}^n, \exists \lambda \in \mathbb{Q}^m : Ax^* = b \ \wedge \ x = x^* + \lambda^\intercal Y \ \wedge \ |\text{supp}(x)| \leq k \\
\Leftrightarrow \quad & \exists x^* \in \mathbb{Q}^n, \exists \lambda \in \mathbb{Q}^m : Ax^* = b \ \wedge \ |\text{supp}\,(x^* + \lambda^\intercal Y)| \leq k \\
\Leftrightarrow \quad & (Y, x^*, k) \text{ is a ``yes''-instance for SparseLinearRow}
\end{aligned}$$

□

Instead of deciding for a certain k, we are interested in the problem which minimizes k (the support of $a + \lambda^\intercal A$). Knowing that SparseLinearRow is $\mathcal{NP}$-complete, we can deduce Corollary 5.4.

Corollary 5.4. *The minimization version of* SparseLinearRow *is $\mathcal{NP}$-hard.*

For a practical implementation we can adopt our previous MIP model which used binary variables to indicate a nonzero coefficient. Of course, as a big-M method this imposes the same numerical problems. Instead of such a scheme which attempts to solve the problem optimally, the state-of-the-art solvers mostly approximate with a very small effort. An example is to test every *single* equation whether it can be combined with a cut row to obtain a cut with better numerical properties.

5.6 Sparsity of Other Classes of Cutting Planes

In this section we present our knowledge about sparsity properties of other cutting plane classes and related work.

We start with simple intersection cuts which we introduced in Section 3.2. By a quick look at the formula it is clear that these cuts simply inherit their sparsity from the tableau row. The situation is even better for GMI cuts. In principle, they also inherit from the tableau. Additionally, a coefficient α_j is zero if j is an integer variable and the corresponding row entry is an integral. The state-of-the-art solvers make heavy use of this property. An important part of the cut generation module is the so-called *aggregator*. Its task is to combine the present LP rows of the tableau in order to find a row with certain properties. Among these properties is sparsity, integrality of coefficients and the necessary condition of a fractional right-hand side. This aggregated row is then fed into several different cut generators, the GMI separator being only one of them. Note that the linear combination does not necessarily yield *basic* GMI cuts.

The only other class of cuts for which we found a remark about sparsity is the class of Chvátal Gomory cuts. It was thoroughly analyzed by Fischetti and Lodi in [26]. They implemented a MIP to generate CG cuts. Compared to lift-and-project this MIP is smaller than the CGLP as it only consists of one set of multipliers instead of two. But the objective function is in fact the same. They enhance their objective function by the sum of the dual multipliers scaled down by some small value. In our context this corresponds to a penalization of dual density. As they reported to generate sparser cuts this way, we can assume that the correspondence observed in Section 5.3.1 is not specific to lift-and-project.

Appendix A

Computational Results per Instance

Densification

Tables A.1 and A.2 show the instance-wise results for our first experiment. It is described in Section 5.2 which also contains interpretation and figures.

Both tables display the speed of the dual simplex implementation after making some rows dense. An additional column "Density" displays the relative density of the equation which was used to make the above rows dense.

Sparsity Measurements

The remaining tables on pages 72 – 83 show the details of the experiment about the actual sparsity. We provide details for rank 1 cuts as well as for cuts of arbitrary rank and state absolute and relative sparsity. All excluded problems were not taken into account because at least one configuration was unable to generate a cut. The specific reasons are twofold:

- The optimal cut violation was not negative.
- The β-NC does not guarantee to bound the polyhedron in every direction. Hence, in some cases the LP was unbounded.

The statistical analysis and interpretation for this experiment can be found in Section 5.3.1.

We do not provide details for the last experiment as there is a lot of information for every generated cut. Hence, the reader has to trust the presented figures in Section 5.3.2.

Instance	Density [%]	0 rows	1 rows	2 rows	3 rows	4 rows	5 rows	6 rows	7 rows	8 rows	9 rows
10teams	100.0	11,693.4	11,327.2	10,952.9	10,403.5	10,149.6	9,787.8	9,484.6	9,040.2	8,813.3	8,465.5
a1c1s1	61.7	2,811.1	2,654.5	2,182.4	2,060.2	1,913.8	1,750.4	1,624.8	1,530.3	1,442.4	1,346.6
aflow30a	33.5	9,893.0	9,948.1	9,556.7	9,455.8	9,267.6	9,148.8	9,027.5	8,918.9	8,831.7	8,759.3
aflow40b	26.7	4,594.6	4,591.9	4,326.8	4,247.7	4,142.7	4,102.4	4,024.9	3,962.0	3,931.1	3,878.9
air04	90.2	4,222.4	4,002.3	3,746.1	3,528.3	3,231.7	3,319.3	3,121.1	2,844.8	2,866.2	2,668.6
air05	89.1	5,278.1	4,847.8	4,557.2	4,225.9	4,038.4	3,900.3	3,752.8	3,545.0	3,431.8	3,372.6
arki001	98.0	6,252.2	6,148.6	5,753.2	5,515.6	5,285.0	5,005.2	4,911.3	4,827.6	4,664.2	4,610.2
atlanta-ip	50.5	441.6	578.6	445.1	270.3	232.3	206.9	194.3	205.3	311.3	310.3
blend2	56.9	19,170.0	18,135.8	17,882.3	17,954.9	17,682.2	17,399.9	17,176.9	16,938.8	16,541.1	15,617.1
dano3mip	94.4	555.0	528.0	437.6	419.8	413.5	387.3	332.9	333.4	300.6	270.9
danoint	88.2	8,295.4	7,903.6	7,600.6	7,352.9	7,161.9	6,971.9	6,805.1	6,620.3	6,461.2	6,303.9
dcmulti	75.1	16,111.0	18,039.3	14,554.1	13,872.9	13,386.7	12,994.3	12,989.1	11,725.7	12,073.5	10,968.2
ds	96.5	398.3	269.7	187.6	122.3	90.0	117.5	104.5	121.1	122.9	95.1
fiber	34.3	15,817.3	10,743.9	15,285.5	15,123.9	12,392.2	14,860.0	14,880.9	14,316.0	14,494.4	11,525.8
fixnet6	26.1	10,806.6	11,296.4	10,734.5	10,064.9	9,619.3	8,210.8	9,066.7	9,415.8	9,272.7	9,291.6
flugpl	64.3	38,674.0	39,248.7	39,659.1	40,381.6	37,635.4	37,636.4	38,034.1	37,210.9	38,059.0	37,975.5
gesa2_o	60.4	4,951.4	4,525.0	4,497.4	4,111.1	3,937.9	3,746.2	3,593.2	3,454.2	3,318.5	3,095.9
gesa2-o	60.4	4,934.1	4,558.0	4,600.9	4,106.7	3,948.8	3,787.0	3,587.8	3,484.0	3,339.2	3,111.7
gesa3_o	41.9	5,274.1	5,150.3	5,073.8	4,494.1	4,400.6	4,073.3	3,932.0	3,937.8	3,809.6	3,732.8
glass4	94.0	15,986.2	15,297.7	14,766.3	14,189.2	13,988.9	13,732.3	13,483.0	13,303.5	12,995.7	12,822.2
harp2	100.0	17,492.7	16,642.8	15,956.8	14,956.4	14,365.3	13,679.9	13,177.9	12,724.6	12,292.9	11,879.6
khb05250	43.9	19,428.7	16,675.2	18,823.4	18,001.3	17,990.0	16,618.8	15,850.4	14,679.8	13,483.8	13,664.8
l152lav	78.4	19,392.0	19,435.6	18,617.6	17,720.3	17,253.4	16,032.6	15,227.8	14,563.1	14,163.4	13,776.6
markshare1	100.0	39,602.2	39,848.8	42,151.8	42,567.0	42,408.2	42,405.6	42,466.9	41,837.3	41,993.9	40,916.7
markshare2	100.0	39,510.2	41,277.1	42,034.9	42,286.5	42,260.1	42,214.8	42,086.3	42,093.8	41,907.9	41,458.7
misc03	80.4	26,430.8	26,426.3	26,438.8	25,336.2	25,094.5	24,679.1	24,987.1	24,824.5	24,579.2	24,417.6
misc06	60.6	9,297.4	9,712.3	9,134.1	8,770.9	8,371.9	8,228.0	7,700.3	7,471.7	7,133.5	7,062.3
misc07	85.4	20,642.7	20,245.1	19,542.4	19,021.3	18,961.9	18,746.2	18,047.3	18,179.8	18,080.4	17,982.0
mod010	78.5	3,346.0	3,390.7	3,226.6	3,001.9	3,300.8	3,141.4	2,925.1	5,012.6	4,289.5	3,322.2
mod011	63.6	1,526.6	1,454.7	1,389.3	1,364.4	1,272.8	1,253.2	1,229.8	1,148.2	1,062.8	1,062.8
modglob	74.5	16,076.2	15,555.9	12,797.0	12,461.7	11,902.8	11,328.9	11,063.8	10,631.6	10,189.5	10,006.1

Table A.1: Simplex speed for densified MIPs (part 1).

Instance	Density [%]	0 rows	1 rows	2 rows	3 rows	4 rows	5 rows	6 rows	7 rows	8 rows	9 rows
momentum2	98.0	355.5	302.0	337.6	299.2	367.4	300.7	299.9	253.2	319.0	294.7
momentum3	99.5	92.9	84.1	83.1	79.2	80.8	80.1	80.8	83.9	78.3	85.7
msc98-ip	54.9	599.9	577.6	488.7	458.8	472.8	452.7	469.4	454.7	509.6	518.0
mzzv11	57.8	1,379.8	1,345.9	1,308.8	1,291.8	1,254.4	1,249.2	1,203.7	1,238.8	1,150.1	1,175.0
mzzv42z	56.5	1,433.3	1,400.1	1,283.5	1,323.6	1,281.1	1,286.3	1,287.4	1,181.6	1,192.6	1,193.6
nw04	92.3	3,374.3	3,287.5	3,253.5	2,814.3	3,258.2	3,443.7	3,368.3	3,176.4	2,900.7	2,858.6
p0033	25.0	48,825.6	55,004.0	59,524.9	57,590.2	60,611.3	58,107.5	58,107.5	56,864.8	59,121.0	56,052.9
p0201	91.8	31,918.9	31,733.6	31,373.3	31,025.2	30,082.3	28,563.4	27,594.8	26,276.4	26,203.4	25,440.9
pk1	98.8	54,476.2	55,267.2	52,589.7	51,680.6	51,625.8	50,205.4	49,000.4	48,354.6	47,325.6	46,604.5
pp08aCUTS	57.0	17,225.0	16,621.7	15,969.0	15,470.5	14,918.1	14,558.4	14,263.0	13,768.9	13,459.9	13,153.3
pp08a	58.1	18,917.8	17,767.2	16,968.3	16,542.6	16,205.0	15,553.9	14,966.4	14,666.7	14,267.0	13,861.4
protfold	93.5	2,861.7	2,752.4	2,627.0	2,511.2	2,423.2	2,374.9	2,332.9	2,293.0	2,227.6	2,158.0
qiu	62.9	7,034.3	6,881.1	6,287.8	5,966.1	5,738.1	5,477.7	5,290.9	5,115.3	4,950.3	4,746.8
qnet1	46.3	13,450.0	13,111.7	12,810.6	12,551.7	11,963.4	11,294.4	11,610.5	11,042.7	11,150.0	10,318.6
qnet1_o	49.3	13,560.8	13,584.0	13,458.0	12,994.5	12,644.3	12,380.8	12,252.3	11,805.9	11,690.5	11,537.0
rd-rplusc-21	91.3	202.9	204.0	205.6	205.8	203.5	203.7	201.0	197.1	199.9	196.0
rentacar	87.0	6,426.9	6,303.5	5,844.8	5,306.7	4,989.8	4,520.0	4,488.1	4,066.8	3,917.1	3,669.3
rgn	77.8	37,525.7	35,193.9	32,552.5	30,739.8	28,286.8	28,029.0	27,724.7	26,648.7	26,067.8	25,070.0
roll3000	78.8	5,482.2	5,331.1	4,955.0	4,564.0	4,363.1	4,302.0	4,012.3	3,961.3	3,845.6	3,803.5
rout	88.6	18,946.1	17,976.5	17,293.2	16,568.8	16,114.2	15,629.7	15,195.4	14,742.3	14,305.5	14,035.9
set1ch	50.1	8,127.4	7,359.8	7,378.2	6,193.7	5,769.1	5,425.1	5,193.8	4,926.9	4,659.9	4,486.1
stp3d	49.6	87.7	99.0	80.8	90.9	85.9	78.0	72.7	88.1	57.2	75.9
swath	98.8	5,380.1	4,532.8	4,159.5	3,848.6	3,465.4	3,239.1	3,091.6	2,870.3	2,723.1	2,621.5
t1717	86.3	840.1	802.2	700.8	638.4	609.9	595.4	523.0	487.3	451.5	436.4
timtab1	98.4	16,352.4	15,459.5	12,994.7	12,045.7	11,401.8	10,836.2	10,306.6	9,971.3	9,269.2	8,890.9
timtab2	96.9	9,036.7	8,476.8	7,011.5	6,401.9	5,981.0	5,594.5	5,288.8	5,029.8	4,696.2	4,446.8
tr12-30	56.1	4,854.1	4,113.2	3,991.9	3,742.9	3,456.6	3,253.8	3,072.7	2,904.5	2,746.5	2,592.1
vpm2	52.1	26,279.0	23,710.6	24,409.8	23,878.4	23,419.8	22,542.4	21,561.7	21,265.7	20,564.2	19,854.3

Table A.2: Simplex speed for densified MIPs (part 2).

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	1,273.3	1.4	162.5	3.7	6.7	6.0	14.1
lseu	80.0	12.7	48.9	10.9	11.3	6.0	31.9
pk1	86.0	18.9	59.7	66.2	71.0	70.3	58.9
aflow40b	2,710.7	15.7	76.7	53.2	52.9	54.5	158.3
aflow30a	826.8	8.8	209.9	17.6	18.7	25.7	55.8
bell3a	47.2	1.9	11.7	8.8	8.2	7.6	8.3
rgn	173.2	5.8	54.9	16.6	14.1	16.0	17.7
gesa3_o	1,058.7	15.7	42.1	26.7	26.9	28.4	34.6
p2756	2,190.1	44.1	37.6	55.6	53.7	51.4	71.2
glass4	143.6	2.1	8.7	5.0	7.3	4.5	3.9
stein27	13.8	7.5	8.1	7.4	8.4	7.9	16.8
misc07	228.3	7.9	148.1	58.6	64.5	66.9	110.4
tr12-30	575.4	3.6	14.3	4.0	4.1	4.4	9.2
gesa2_o	1,148.2	6.4	18.6	10.1	13.0	11.5	17.7
khb05250	1,247.3	3.2	46.4	50.4	45.7	42.9	11.6
p0033	28.2	3.7	7.5	5.9	6.4	5.9	9.4
misc03	136.3	6.6	67.6	38.6	38.7	40.5	52.2
egout	34.7	2.2	8.3	3.5	3.0	3.4	6.1
mas74	138.1	10.3	53.2	146.5	144.5	136.7	113.0
mitre	4,941.0	4.1	12.4	41.0	38.8	41.8	23.3
momentum1	2,711.6	62.8	11.5	36.7	5.0	45.7	515.1
gen	592.8	9.4	9.3	13.8	17.4	16.5	19.0
liu	64.0	4.8	6.6	4.4	4.8	4.5	7.6
fixnet6	818.7	10.0	62.0	46.6	74.2	57.2	62.5
gesa2	1,067.9	6.6	13.4	10.3	11.9	7.8	13.1
air03	10,753.6	4.7	589.2	1,822.3	6,341.0	2,933.1	5,173.2
fiber	1,030.4	8.9	61.8	11.5	14.2	12.3	24.3
p0201	190.9	28.5	70.4	71.4	85.7	81.6	90.8
mas76	140.3	6.7	22.8	143.1	134.8	130.6	112.7
modglob	315.4	8.0	45.3	14.5	15.8	14.7	15.2
vpm1	151.8	3.5	7.6	5.4	5.9	5.7	6.9
timtab2	165.1	3.6	8.4	4.1	3.4	3.6	7.9
pp08aCUTS	201.7	5.8	10.4	10.3	11.5	17.8	20.4
gesa3	967.0	20.1	21.0	26.7	28.0	22.9	36.9
markshare2	60.0	29.4	29.9	55.5	53.6	53.6	32.8
gesa2-o	1,149.6	6.5	14.9	9.9	15.6	13.2	17.0
dcmulti	493.5	12.7	39.2	44.3	52.1	52.1	51.3
rd-rplusc-21	453.7	4.7	29.9	13.7	22.9	20.5	29.2
swath	6,296.7	7.8	42.4	275.7	144.0	269.9	931.0
manna81	617.7	3.0	3.0	3.0	3.0	3.0	3.0
opt1217	767.3	40.3	343.8	146.4	162.6	167.5	196.7
flugpl	13.1	1.7	4.1	4.0	4.5	2.6	6.2
p0548	320.8	6.7	69.8	7.2	7.0	7.3	13.3
l152lav	1,988.4	30.3	266.8	1,213.7	1,374.1	1,358.0	1,038.3
mkc	3,239.6	6.8	190.8	81.7	71.0	76.1	105.4
protfold	1,784.2	85.3	75.3	284.8	917.0	502.5	853.9
roll3000	798.9	21.0	56.9	35.2	70.0	48.5	98.1
arki001	932.4	39.9	124.9	55.0	50.4	54.0	111.8
misc06	1,133.4	24.7	58.2	34.6	28.4	29.8	63.6
nsrand-ipx	6,557.3	627.4	601.1	1,023.1	968.7	960.1	2,587.0
a1c1s1	2,066.6	6.6	24.6	11.5	12.5	15.0	21.8
stein45	24.1	12.6	17.8	12.9	13.5	14.3	24.7
rout	541.8	25.6	88.5	74.0	85.2	79.8	154.7
pp08a	189.6	3.2	7.3	5.6	7.7	7.8	8.2
blend2	262.8	9.9	39.2	34.2	37.5	26.6	48.5
bell5	52.3	2.3	5.0	3.5	2.9	3.2	6.5
qiu	728.8	69.9	204.8	92.2	96.2	83.8	162.9
set1ch	531.7	2.0	22.0	3.1	2.9	3.5	6.0
mod011	6,376.4	5.5	7.0	251.1	257.3	436.8	91.9
mod010	2,461.7	14.5	125.8	308.8	426.1	385.0	700.0
cap6000	5,566.4	53.6	1,916.5	5,847.5	5,862.0	5,862.0	4,789.0
p0282	163.3	9.8	19.5	13.8	13.9	13.9	28.5
noswot	74.4	4.6	8.9	8.3	7.8	7.1	7.7
harp2	1,286.3	30.1	139.9	45.0	53.4	33.4	99.3
qnet1	1,362.7	8.5	250.9	16.4	21.2	18.1	50.3
seymour	920.1	27.8	10.9	34.6	41.5	38.9	585.5
vpm2	145.0	3.0	9.4	5.3	4.0	5.0	6.0
timtab1	94.2	3.1	8.4	3.9	3.6	3.9	6.9
net12	12,261.4	3.8	7.1	5.7	3.6	5.1	22.9
gt2	153.4	5.9	40.1	14.3	5.6	5.6	34.3
mod008	318.4	280.3	184.1	242.4	243.0	245.5	310.5
markshare1	50.0	25.5	27.9	47.8	47.8	47.8	28.0
danoint	509.5	80.0	210.8	50.1	70.7	76.9	266.6
momentum2	2,753.2	23.5	27.4	21.9	24.8	28.3	91.5
enigma	98.3	16.0	76.6	84.8	85.1	85.7	73.7
10teams	1,599.4	39.6	250.8	627.5	659.2	756.9	978.2

Table A.3: Arithmetic mean of absolute primal density of cuts with rank 1.

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	1,308.7	34.7	162.5	104.7	148.0	114.3	468.2
lseu	82.1	40.5	48.9	62.5	61.6	62.2	66.8
pk1	86.0	19.0	73.4	78.5	80.9	70.9	63.5
aflow40b	2,711.4	181.3	76.7	405.4	451.7	403.4	1,830.1
aflow30a	837.2	101.7	209.9	249.0	447.2	216.8	590.7
bell3a	67.2	8.5	62.9	13.4	15.6	13.3	10.9
rgn	171.1	13.5	119.5	41.5	55.8	51.2	60.4
gesa3_o	1,055.9	24.5	434.7	49.2	59.2	40.7	240.0
p2756	2,218.7	68.1	37.6	128.6	144.8	121.8	152.1
glass4	226.2	4.4	7.1	10.2	11.2	10.0	14.8
stein27	19.1	8.1	14.2	9.4	17.6	7.6	21.0
misc07	229.1	6.4	148.1	73.3	122.3	99.8	117.7
tr12-30	774.3	59.5	750.2	76.8	97.5	59.9	236.5
gesa2_o	1,153.2	19.1	837.5	35.2	48.5	32.1	57.9
khb05250	1,234.8	5.7	882.3	59.0	85.6	143.1	85.6
p0033	26.3	8.2	12.0	9.5	11.0	12.9	13.5
misc03	135.5	5.3	67.6	60.2	97.8	85.0	77.0
egout	37.7	4.2	33.5	6.4	7.0	11.3	13.2
mas74	137.6	2.5	118.8	150.0	150.0	150.0	112.2
mitre	4,910.3	4.6	12.4	26.6	46.0	38.8	40.5
momentum1	2,711.3	59.1	11.5	124.5	5.0	44.6	617.4
gen	589.6	10.0	4.7	14.1	15.7	16.3	16.4
liu	148.4	11.2	11.2	12.9	22.3	29.4	39.3
fixnet6	851.9	68.2	566.6	203.4	414.6	205.1	504.9
gesa2	1,082.0	15.8	936.9	33.0	71.4	26.4	94.1
air03	10,753.6	4.7	501.3	2,150.9	1,023.7	2,732.2	5,173.2
fiber	1,036.4	32.7	61.8	66.0	113.2	82.3	180.3
p0201	192.8	36.5	70.4	88.9	124.2	84.5	138.8
mas76	141.5	2.0	22.8	150.0	150.0	150.0	112.6
modglob	336.2	15.2	45.3	35.2	106.2	28.9	100.3
vpm1	145.2	8.8	7.6	7.9	16.8	15.2	14.6
timtab2	213.4	21.6	8.4	41.2	96.7	77.4	120.3
pp08aCUTS	222.7	13.3	206.8	31.4	46.7	46.1	109.5
gesa3	1,026.8	22.5	21.0	43.1	55.0	27.4	131.6
markshare2	60.0	33.2	33.2	60.0	60.0	60.0	32.8
gesa2-o	1,151.8	19.9	14.9	37.1	50.6	34.7	137.9
dcmulti	505.3	20.8	269.4	82.0	159.5	64.2	279.2
rd-rplusc-21	464.2	6.3	29.1	23.8	22.9	26.5	156.7
swath	6,298.3	8.1	41.0	288.8	394.0	498.3	1,528.0
manna81	583.2	3.0	3.0	3.0	3.0	3.0	3.0
opt1217	767.9	23.7	589.1	74.8	75.0	74.5	199.8
flugpl	13.1	5.6	13.3	8.9	9.2	9.2	8.8
p0548	329.7	11.5	276.9	15.0	14.4	14.1	19.0
l152lav	1,988.1	36.5	266.8	1,655.8	1,683.6	1,386.7	1,416.3
mkc	3,231.9	8.7	162.7	77.4	107.8	107.3	534.4
protfold	1,784.0	75.9	37.0	515.8	912.1	404.7	662.1
roll3000	799.5	31.6	57.3	46.4	92.4	52.8	192.4
arki001	924.7	62.8	125.6	139.7	133.9	111.8	319.8
misc06	1,072.1	29.5	58.2	41.2	40.0	40.4	110.5
nsrand-ipx	6,570.8	627.4	531.4	1,131.6	960.4	960.1	3,973.8
a1c1s1	2,066.6	22.6	24.6	76.9	346.3	189.9	667.5
stein45	34.1	12.9	17.8	17.1	29.2	13.8	31.5
rout	545.8	104.6	88.5	173.3	328.0	127.2	309.5
pp08a	215.9	13.0	42.4	21.9	33.9	25.0	76.1
blend2	271.0	35.5	103.9	82.3	142.9	91.2	73.2
bell5	58.5	10.6	5.0	13.9	17.8	12.4	16.3
qiu	787.7	107.3	378.0	178.0	219.4	98.5	161.2
set1ch	565.9	7.9	22.0	8.6	11.2	12.1	31.4
mod011	6,376.4	5.5	375.2	246.1	336.8	436.8	896.0
mod010	2,461.9	14.1	129.2	358.5	676.6	399.7	1,017.9
cap6000	5,556.7	108.4	1,916.5	3,381.9	5,862.0	5,862.0	4,789.0
p0282	165.9	20.9	19.5	35.0	44.4	36.1	46.7
noswot	79.3	6.1	41.8	8.8	18.9	10.7	14.7
harp2	1,314.2	58.5	170.0	81.8	78.2	93.2	621.8
qnet1	1,385.4	45.3	1,012.9	74.3	136.8	79.3	763.8
seymour	934.6	29.0	11.0	56.0	189.1	38.8	747.4
vpm2	164.5	20.5	9.4	24.4	42.8	22.9	98.7
timtab1	125.4	14.4	8.4	20.8	39.3	37.2	55.8
net12	12,347.2	3.5	7.5	4.7	3.7	4.8	121.7
gt2	166.3	71.6	40.1	65.5	62.5	86.9	108.9
mod008	317.5	113.5	294.0	312.9	317.2	317.1	133.0
markshare1	50.0	27.9	27.9	50.0	50.0	50.0	27.9
danoint	511.2	79.1	210.8	149.1	321.0	132.3	304.7
momentum2	2,752.7	23.4	26.8	34.6	21.7	31.6	65.4
enigma	99.7	17.4	79.8	93.1	95.5	91.2	83.7
10teams	1,599.4	50.0	243.4	518.7	659.0	695.3	995.6

Table A.4: Arithmetic mean of absolute primal density of cuts with arbitrary rank.

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	95.7	0.1	12.2	0.3	0.5	0.4	1.1
lseu	91.9	14.6	56.2	12.6	13.0	6.9	36.6
pk1	100.0	21.9	69.5	77.0	82.5	81.8	68.5
aflow40b	99.4	0.6	2.8	1.9	1.9	2.0	5.8
aflow30a	98.2	1.0	24.9	2.1	2.2	3.0	6.6
bell3a	51.3	2.0	12.7	9.6	8.9	8.3	9.0
rgn	99.0	3.3	31.4	9.5	8.0	9.1	10.1
gesa3_o	98.0	1.4	3.9	2.5	2.5	2.6	3.2
p2756	92.4	1.9	1.6	2.3	2.3	2.2	3.0
glass4	45.3	0.7	2.7	1.6	2.3	1.4	1.2
stein27	50.9	27.9	29.9	27.5	31.0	29.3	62.1
misc07	98.4	3.4	63.8	25.3	27.8	28.9	47.6
tr12-30	55.3	0.3	1.4	0.4	0.4	0.4	0.9
gesa2_o	97.6	0.5	1.6	0.9	1.1	1.0	1.5
khb05250	96.0	0.2	3.6	3.9	3.5	3.3	0.9
p0033	93.9	12.3	25.1	19.8	21.2	19.7	31.3
misc03	98.7	4.8	49.0	28.0	28.0	29.3	37.8
egout	70.8	4.4	17.0	7.2	6.1	7.0	12.4
mas74	92.0	6.9	35.5	97.7	96.3	91.1	75.4
mitre	100.0	0.1	0.3	0.8	0.8	0.8	0.5
momentum1	98.6	2.3	0.4	1.3	0.2	1.7	18.7
gen	92.9	1.5	1.5	2.2	2.7	2.6	3.0
liu	5.5	0.4	0.6	0.4	0.4	0.4	0.7
fixnet6	93.4	1.1	7.1	5.3	8.5	6.5	7.1
gesa2	90.8	0.6	1.1	0.9	1.0	0.7	1.1
air03	100.0	0.0	5.5	16.9	59.0	27.3	48.1
fiber	98.5	0.9	5.9	1.1	1.4	1.2	2.3
p0201	97.9	14.6	36.1	36.6	44.0	41.8	46.5
mas76	93.6	4.5	15.2	95.4	89.9	87.0	75.2
modglob	82.1	2.1	11.8	3.8	4.1	3.8	4.0
vpm1	83.4	1.9	4.2	3.0	3.2	3.1	3.8
timtab2	48.4	1.0	2.5	1.2	1.0	1.1	2.3
pp08aCUTS	85.1	2.4	4.4	4.3	4.8	7.5	8.6
gesa3	89.5	1.9	1.9	2.5	2.6	2.1	3.4
markshare2	100.0	49.1	49.9	92.6	89.3	89.3	54.6
gesa2-o	97.8	0.6	1.3	0.8	1.3	1.1	1.4
dcmulti	93.3	2.4	7.4	8.4	9.9	9.8	9.7
rd-rplusc-21	87.2	0.9	5.7	2.6	4.4	3.9	5.6
swath	99.6	0.1	0.7	4.4	2.3	4.3	14.7
manna81	18.6	0.1	0.1	0.1	0.1	0.1	0.1
opt1217	99.8	5.2	44.7	19.0	21.1	21.8	25.6
flugpl	93.3	12.4	29.5	28.4	32.2	18.6	44.5
p0548	78.4	1.6	17.1	1.8	1.7	1.8	3.2
l152lav	100.0	1.5	13.4	61.0	69.1	68.3	52.2
mkc	99.0	0.2	5.8	2.5	2.2	2.3	3.2
protfold	97.2	4.7	4.1	15.5	50.0	27.4	46.5
roll3000	92.7	2.4	6.6	4.1	8.1	5.6	11.4
arki001	97.2	4.2	13.0	5.7	5.3	5.6	11.7
misc06	90.0	2.0	4.6	2.7	2.3	2.4	5.0
nsrand-ipx	99.4	9.5	9.1	15.5	14.7	14.5	39.2
a1c1s1	82.9	0.3	1.0	0.5	0.5	0.6	0.9
stein45	53.6	28.0	39.6	28.7	30.1	31.7	54.9
rout	97.6	4.6	15.9	13.3	15.4	14.4	27.9
pp08a	81.0	1.4	3.1	2.4	3.3	3.3	3.5
blend2	85.9	3.2	12.8	11.2	12.3	8.7	15.8
bell5	65.4	2.9	6.3	4.4	3.6	4.1	8.1
qiu	86.8	8.3	24.4	11.0	11.5	10.0	19.4
set1ch	82.3	0.3	3.4	0.5	0.5	0.5	0.9
mod011	98.1	0.1	0.1	3.9	4.0	6.7	1.4
mod010	99.9	0.6	5.1	12.5	17.3	15.6	28.4
cap6000	94.9	0.9	32.7	99.7	99.9	99.9	81.7
p0282	81.7	4.9	9.7	6.9	6.9	6.9	14.3
noswot	62.0	3.8	7.4	6.9	6.5	5.9	6.4
harp2	93.7	2.2	10.2	3.3	3.9	2.4	7.2
qnet1	96.2	0.6	17.7	1.2	1.5	1.3	3.6
seymour	78.2	2.4	0.9	2.9	3.5	3.3	49.8
vpm2	80.1	1.6	5.2	2.9	2.2	2.8	3.3
timtab1	46.9	1.5	4.2	2.0	1.8	2.0	3.5
net12	97.9	0.0	0.1	0.0	0.0	0.0	0.2
gt2	88.7	3.4	23.2	8.2	3.2	3.2	19.9
mod008	99.8	87.9	57.7	76.0	76.2	77.0	97.3
markshare1	100.0	50.9	55.9	95.5	95.5	95.5	56.0
danoint	99.3	15.6	41.1	9.8	13.8	15.0	52.0
momentum2	99.0	0.8	1.0	0.8	0.9	1.0	3.3
enigma	98.3	16.0	76.6	84.8	85.1	85.7	73.7
10teams	100.0	2.5	15.7	39.2	41.2	47.3	61.1

Table A.5: Arithmetic mean of relative primal density of cuts with rank 1 in %.

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	98.4	2.6	12.2	7.9	11.1	8.6	35.2
lseu	94.3	46.5	56.2	71.9	70.8	71.5	76.8
pk1	100.0	22.1	85.4	91.2	94.1	82.4	73.9
aflow40b	99.4	6.6	2.8	14.9	16.6	14.8	67.1
aflow30a	99.4	12.1	24.9	29.6	53.1	25.7	70.1
bell3a	73.0	9.2	68.4	14.6	16.9	14.5	11.9
rgn	97.7	7.7	68.3	23.7	31.9	29.3	34.5
gesa3_o	97.8	2.3	40.2	4.6	5.5	3.8	22.2
p2756	93.6	2.9	1.6	5.4	6.1	5.1	6.4
glass4	71.4	1.4	2.2	3.2	3.5	3.1	4.7
stein27	70.7	30.1	52.7	34.8	65.3	28.1	77.7
misc07	98.8	2.7	63.8	31.6	52.7	43.0	50.7
tr12-30	74.5	5.7	72.1	7.4	9.4	5.8	22.7
gesa2_o	98.1	1.6	71.2	3.0	4.1	2.7	4.9
khb05250	95.1	0.4	67.9	4.5	6.6	11.0	6.6
p0033	87.7	27.4	40.0	31.7	36.8	42.9	45.1
misc03	98.2	3.8	49.0	43.6	70.9	61.6	55.8
egout	77.0	8.6	68.4	13.0	14.3	23.1	27.0
mas74	91.7	1.7	79.2	100.0	100.0	100.0	74.8
mitre	99.4	0.1	0.3	0.5	0.9	0.8	0.8
momentum1	98.6	2.1	0.4	4.5	0.2	1.6	22.4
gen	92.4	1.6	0.7	2.2	2.5	2.6	2.6
liu	12.9	1.0	1.0	1.1	1.9	2.5	3.4
fixnet6	97.1	7.8	64.6	23.2	47.3	23.4	57.6
gesa2	92.0	1.3	79.7	2.8	6.1	2.2	8.0
air03	100.0	0.0	4.7	20.0	9.5	25.4	48.1
fiber	99.1	3.1	5.9	6.3	10.8	7.9	17.2
p0201	98.9	18.7	36.1	45.6	63.7	43.4	71.2
mas76	94.3	1.3	15.2	100.0	100.0	100.0	75.0
modglob	87.6	4.0	11.8	9.2	27.7	7.5	26.1
vpm1	79.8	4.8	4.2	4.3	9.2	8.4	8.0
timtab2	62.6	6.3	2.5	12.1	28.3	22.7	35.3
pp08aCUTS	94.0	5.6	87.2	13.2	19.7	19.5	46.2
gesa3	95.1	2.1	1.9	4.0	5.1	2.5	12.2
markshare2	100.0	55.4	55.4	100.0	100.0	100.0	54.7
gesa2-o	97.9	1.7	1.3	3.2	4.3	2.9	11.7
dcmulti	95.5	3.9	50.9	15.5	30.1	12.1	52.8
rd-rplusc-21	89.3	1.2	5.6	4.6	4.4	5.1	30.1
swath	99.7	0.1	0.6	4.6	6.2	7.9	24.2
manna81	17.6	0.1	0.1	0.1	0.1	0.1	0.1
opt1217	99.9	3.1	76.6	9.7	9.8	9.7	26.0
flugpl	93.5	40.1	94.9	63.9	65.6	65.8	62.9
p0548	80.6	2.8	67.7	3.7	3.5	3.4	4.6
l152lav	100.0	1.8	13.4	83.2	84.6	69.7	71.2
mkc	98.7	0.3	5.0	2.4	3.3	3.3	16.3
protfold	97.2	4.1	2.0	28.1	49.7	22.1	36.1
roll3000	92.8	3.7	6.6	5.4	10.7	6.1	22.3
arki001	96.4	6.6	13.1	14.6	14.0	11.7	33.3
misc06	85.1	2.3	4.6	3.3	3.2	3.2	8.8
nsrand-ipx	99.6	9.5	8.1	17.1	14.6	14.5	60.2
a1c1s1	82.9	0.9	1.0	3.1	13.9	7.6	26.8
stein45	75.8	28.7	39.6	38.0	64.8	30.6	70.0
rout	98.3	18.8	15.9	31.2	59.1	22.9	55.8
pp08a	92.2	5.6	18.1	9.4	14.5	10.7	32.5
blend2	88.6	11.6	33.9	26.9	46.7	29.8	23.9
bell5	73.1	13.3	6.3	17.4	22.2	15.5	20.3
qiu	93.8	12.8	45.0	21.2	26.1	11.7	19.2
set1ch	87.6	1.2	3.4	1.3	1.7	1.9	4.9
mod011	98.1	0.1	5.8	3.8	5.2	6.7	13.8
mod010	100.0	0.6	5.2	14.6	27.5	16.2	41.3
cap6000	94.7	1.8	32.7	57.7	99.9	99.9	81.7
p0282	82.9	10.4	9.7	17.5	22.2	18.1	23.3
noswot	66.1	5.1	34.8	7.4	15.7	8.9	12.3
harp2	95.7	4.3	12.4	6.0	5.7	6.8	45.3
qnet1	97.8	3.2	71.5	5.2	9.7	5.6	53.9
seymour	79.5	2.5	0.9	4.8	16.1	3.3	63.6
vpm2	90.9	11.3	5.2	13.5	23.6	12.6	54.5
timtab1	62.4	7.2	4.2	10.4	19.6	18.5	27.7
net12	98.6	0.0	0.1	0.0	0.0	0.0	1.0
gt2	96.1	41.4	23.2	37.9	36.1	50.2	63.0
mod008	99.5	35.6	92.2	98.1	99.4	99.4	41.7
markshare1	100.0	55.8	55.8	100.0	100.0	100.0	55.8
danoint	99.6	15.4	41.1	29.1	62.6	25.8	59.4
momentum2	99.0	0.8	1.0	1.2	0.8	1.1	2.4
enigma	99.7	17.4	79.8	93.1	95.5	91.2	83.7
10teams	100.0	3.1	15.2	32.4	41.2	43.5	62.2

Table A.6: Arithmetic mean of relative primal density of cuts with arbitrary rank in %.

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	1,103.7	4.4	192.2	4.4	5.5	8.2	48.9
lseu	72.4	12.2	25.4	7.1	6.6	4.1	17.5
pk1	76.9	66.6	48.8	40.1	42.6	42.8	51.6
aflow40b	1,841.9	57.6	81.1	36.5	39.6	38.4	158.0
aflow30a	517.0	25.3	182.2	14.6	16.5	21.2	56.3
bell3a	57.1	6.7	11.1	7.2	7.3	7.3	12.4
rgn	134.5	23.2	33.3	16.4	15.1	15.7	25.1
gesa3_o	865.7	24.7	37.2	21.8	20.9	21.8	34.8
p2756	2,168.6	30.8	23.8	32.1	32.1	29.6	43.1
glass4	137.2	2.0	6.1	4.2	5.5	3.3	3.8
stein27	7.3	2.7	3.6	2.7	3.1	2.9	8.8
misc07	203.1	40.8	68.7	22.3	23.8	23.2	79.2
tr12-30	407.4	4.7	10.3	4.7	5.1	5.3	11.2
gesa2_o	977.8	10.5	15.8	9.2	11.2	9.0	18.7
khb05250	1,140.0	41.7	52.4	48.3	43.0	42.6	72.5
p0033	21.1	6.2	5.7	3.6	3.7	3.5	6.4
misc03	120.8	15.7	42.1	15.5	16.0	15.5	43.3
egout	30.7	3.2	4.5	2.7	2.7	2.8	5.8
mas74	145.6	108.2	123.5	67.5	64.9	62.4	91.3
mitre	4,494.4	33.7	13.1	27.5	26.8	26.9	27.7
momentum1	2,526.5	1,087.2	13.4	30.8	5.2	34.6	503.3
gen	583.4	29.4	9.0	10.2	12.2	11.6	12.9
liu	55.4	3.3	4.7	2.9	3.1	2.9	4.8
fixnet6	778.2	27.0	72.5	32.6	45.4	39.3	77.8
gesa2	979.5	11.2	14.6	9.1	9.6	9.3	17.1
air03	10,561.5	1,598.3	872.8	459.1	515.0	636.7	4,799.1
fiber	940.1	14.6	52.0	11.9	13.8	13.0	28.7
p0201	165.0	58.5	53.7	20.9	21.0	20.3	55.8
mas76	146.0	92.5	139.1	65.7	57.4	56.1	93.6
modglob	282.8	11.4	42.1	12.2	13.2	12.0	15.5
vpm1	143.6	6.5	7.3	5.0	5.5	5.1	7.5
timtab2	121.9	2.8	7.6	2.5	2.5	2.6	5.7
pp08aCUTS	140.8	8.8	9.4	7.8	8.9	13.1	18.3
gesa3	870.3	28.3	21.3	22.1	22.9	23.7	40.1
markshare2	48.8	38.6	38.6	27.8	26.8	26.8	43.6
gesa2-o	1,001.0	10.4	12.5	9.0	13.4	10.0	18.6
dcmulti	345.7	51.4	37.1	27.7	31.0	30.6	50.8
rd-rplusc-21	349.4	19.4	30.5	8.5	12.9	18.6	30.0
swath	4,944.6	466.5	27.4	244.3	125.5	230.7	914.4
manna81	455.9	1.5	1.5	1.5	1.5	1.5	1.5
opt1217	728.3	460.3	236.4	84.5	95.2	98.8	136.3
flugpl	10.3	2.1	2.3	2.1	2.6	2.1	3.9
p0548	308.0	10.9	31.0	8.1	8.4	7.9	15.2
l152lav	1,891.6	1,280.6	291.8	638.6	653.4	661.1	1,160.1
mkc	3,116.9	129.1	91.5	64.5	53.3	58.7	148.1
protfold	469.1	376.7	49.1	151.2	343.3	102.6	1,012.6
roll3000	702.1	66.7	22.6	11.3	12.7	11.9	57.6
arki001	614.9	138.4	103.0	31.2	28.2	29.1	105.4
misc06	1,019.0	52.0	42.0	22.4	19.0	20.2	57.5
nsrand-ipx	6,419.4	256.4	246.5	335.8	293.6	292.5	929.4
a1c1s1	1,444.8	9.6	6.7	11.1	12.2	14.6	24.9
stein45	13.9	5.5	9.2	5.7	6.0	6.4	12.3
rout	493.3	120.2	99.3	49.0	58.8	56.3	216.9
pp08a	134.9	5.6	7.8	5.1	6.5	7.2	9.4
blend2	256.8	58.6	35.9	21.5	25.3	16.8	47.1
bell5	48.3	2.3	4.8	3.1	2.9	3.0	6.1
qiu	535.5	167.6	185.9	80.6	83.7	76.2	147.9
set1ch	463.6	3.6	4.1	3.0	3.1	3.2	6.5
mod011	4,248.6	83.3	8.3	187.7	191.7	329.3	278.8
mod010	1,949.6	509.6	159.1	252.9	271.5	270.4	717.8
cap6000	5,662.0	2,526.4	2,965.3	2,939.8	2,932.5	2,932.5	3,469.0
p0282	155.6	8.6	17.3	6.0	5.7	5.7	13.2
noswot	71.4	5.9	6.4	5.2	5.0	5.3	9.7
harp2	1,024.3	185.3	77.5	60.3	64.2	52.8	189.4
qnet1	1,213.1	44.7	195.5	26.3	29.3	25.7	88.9
seymour	655.3	10.8	5.4	13.4	15.1	14.5	253.8
vpm2	127.3	5.9	10.0	5.1	4.7	5.5	6.8
timtab1	70.2	2.3	7.1	2.4	2.6	2.7	5.0
net12	11,966.7	6.4	5.7	5.4	3.0	3.9	34.5
gt2	154.3	8.2	27.5	5.1	4.4	4.3	27.7
mod008	311.3	116.8	92.8	121.5	121.8	123.1	155.4
markshare1	40.7	33.4	34.8	23.9	23.9	23.9	36.0
danoint	420.2	395.6	182.4	26.0	36.4	36.3	266.5
momentum2	2,452.9	203.4	29.4	18.6	21.9	25.2	108.0
enigma	92.5	77.7	55.8	47.5	48.0	49.3	51.0
10teams	1,505.6	333.0	241.2	352.1	359.2	406.7	1,045.8

Table A.7: Arithmetic mean of absolute dual density of cuts with rank 1.

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	1,186.1	174.4	192.2	94.8	121.0	101.7	648.8
lseu	70.7	42.1	25.4	31.4	28.8	29.9	37.8
pk1	76.3	68.7	143.2	42.0	42.6	42.8	50.2
aflow40b	2,522.4	443.0	81.1	189.6	185.6	223.3	1,511.9
aflow30a	757.5	173.8	182.2	124.5	221.2	121.6	429.9
bell3a	63.8	48.9	159.5	11.8	13.7	11.9	15.8
rgn	144.3	57.9	233.3	18.5	17.9	17.5	50.1
gesa3_o	860.1	30.5	1,370.8	23.9	27.5	23.3	168.6
p2756	2,194.2	253.9	23.8	60.2	55.8	44.9	104.5
glass4	209.3	7.9	5.2	7.0	8.2	7.6	16.5
stein27	11.5	3.1	66.6	3.7	8.3	2.7	11.1
misc07	204.4	31.5	68.7	23.6	29.6	37.5	81.0
tr12-30	643.5	99.5	2,773.6	53.7	64.4	50.1	208.9
gesa2_o	992.6	24.1	2,681.8	19.3	24.1	19.7	41.4
khb05250	1,096.5	58.0	1,900.6	53.1	69.4	127.3	282.7
p0033	20.7	14.1	9.1	6.1	6.0	6.7	8.2
misc03	119.9	16.4	42.1	21.3	38.6	33.4	50.2
egout	32.7	5.1	90.7	4.0	4.5	7.4	9.0
mas74	147.1	107.2	251.8	71.4	72.2	69.3	93.9
mitre	3,445.5	35.7	13.1	16.8	26.6	23.6	38.3
momentum1	2,526.3	1,018.0	13.4	85.4	5.2	32.1	569.9
gen	577.2	45.4	3.8	9.1	8.9	9.8	11.4
liu	91.5	5.8	6.8	5.6	8.8	12.3	15.5
fixnet6	820.4	96.5	1,457.1	91.3	180.6	90.5	357.2
gesa2	977.1	76.7	3,078.8	20.9	37.2	20.3	72.9
air03	10,561.5	1,598.3	780.3	623.5	503.0	630.8	4,799.1
fiber	940.0	106.1	52.0	37.9	57.0	46.5	177.8
p0201	163.6	70.0	53.7	28.3	43.1	22.4	85.5
mas76	147.6	120.8	139.1	73.7	73.2	72.7	93.7
modglob	298.9	18.0	42.1	18.0	54.4	18.2	73.4
vpm1	137.0	17.6	7.3	5.7	9.1	8.3	12.1
timtab2	153.0	14.3	7.6	23.4	52.1	42.9	70.7
pp08aCUTS	172.9	15.4	780.1	17.4	25.8	28.4	80.3
gesa3	904.1	103.4	21.3	25.9	30.2	21.3	108.0
markshare2	48.7	43.8	43.4	30.0	30.0	30.0	43.6
gesa2-o	995.3	25.1	12.5	19.7	25.3	21.1	96.1
dcmulti	380.7	67.4	704.6	33.4	65.0	31.7	178.9
rd-rplusc-21	343.1	27.6	31.6	12.2	12.9	19.0	104.4
swath	5,326.2	318.7	26.4	146.3	145.9	256.4	1,262.6
manna81	431.8	1.5	1.5	1.5	1.5	1.5	1.5
opt1217	727.3	444.8	1,003.9	33.5	30.9	30.8	140.2
flugpl	9.8	4.4	49.3	4.5	4.5	4.3	4.4
p0548	307.2	13.3	672.7	9.5	9.0	9.4	14.8
l152lav	1,893.7	1,337.2	291.8	831.4	323.5	657.8	1,189.7
mkc	3,042.2	132.1	198.0	48.7	52.6	55.0	271.3
protfold	465.8	293.6	38.0	254.6	342.6	86.4	498.1
roll3000	704.9	127.1	22.9	17.1	15.9	13.0	112.0
arki001	629.8	222.8	103.4	69.6	62.5	57.1	257.8
misc06	1,004.0	58.1	42.0	21.6	19.4	23.4	85.0
nsrand-ipx	6,438.6	256.4	209.9	379.4	292.6	292.5	1,548.6
a1c1s1	1,444.8	30.4	6.7	50.3	194.5	136.3	484.6
stein45	21.6	5.7	9.2	7.9	14.3	6.1	16.8
rout	491.5	328.6	99.3	90.5	160.4	75.6	291.9
pp08a	171.2	14.4	105.0	13.8	19.9	17.0	56.9
blend2	257.1	143.0	250.5	43.4	57.9	49.1	52.1
bell5	47.7	9.8	4.8	8.1	9.9	7.5	9.7
qiu	578.3	201.7	1,183.4	113.7	127.0	87.1	147.9
set1ch	487.2	9.6	4.1	6.2	7.6	9.0	25.4
mod011	4,248.6	83.3	903.3	184.5	189.2	329.3	1,558.4
mod010	1,963.5	430.7	159.5	191.8	263.1	266.1	830.4
cap6000	5,744.8	4,161.1	2,965.3	1,796.2	2,932.5	2,932.5	3,469.0
p0282	160.0	28.4	17.3	17.9	22.1	19.3	28.8
noswot	75.0	7.8	142.6	5.1	11.2	6.7	14.6
harp2	1,075.2	466.7	98.9	61.1	58.1	68.6	536.4
qnet1	1,273.2	191.2	2,182.0	67.9	101.6	79.3	786.2
seymour	696.7	11.6	5.4	23.0	82.7	14.7	325.4
vpm2	143.9	29.0	10.0	12.9	21.3	12.7	67.0
timtab1	91.2	9.0	7.1	11.3	21.1	19.8	33.5
net12	12,109.7	5.8	6.2	3.3	3.3	4.4	234.4
gt2	155.4	69.8	27.5	39.4	35.3	46.4	63.8
mod008	311.9	427.9	556.2	157.2	158.9	158.9	247.7
markshare1	39.7	36.1	35.8	24.7	25.0	25.0	35.8
danoint	439.5	407.3	182.4	76.6	155.5	70.2	311.4
momentum2	2,453.3	198.6	28.6	24.2	18.4	23.5	74.8
enigma	92.5	88.1	59.9	47.4	48.9	48.3	52.1
10teams	1,539.3	396.6	234.0	283.9	345.9	372.3	1,035.7

Table A.8: Arithmetic mean of absolute dual density of cuts with arbitrary rank.

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	35.5	0.1	6.2	0.1	0.2	0.3	1.6
lseu	35.9	6.1	12.6	3.5	3.3	2.0	8.7
pk1	33.1	28.7	21.0	17.3	18.4	18.4	22.2
aflow40b	26.4	0.8	1.2	0.5	0.6	0.6	2.3
aflow30a	23.3	1.1	8.2	0.7	0.7	1.0	2.5
bell3a	22.7	2.7	4.4	2.9	2.9	2.9	4.9
rgn	34.1	5.9	8.5	4.2	3.8	4.0	6.4
gesa3_o	25.6	0.7	1.1	0.6	0.6	0.6	1.0
p2756	34.8	0.5	0.4	0.5	0.5	0.5	0.7
glass4	13.0	0.2	0.6	0.4	0.5	0.3	0.4
stein27	4.2	1.6	2.1	1.6	1.8	1.7	5.1
misc07	28.6	5.8	9.7	3.1	3.4	3.3	11.2
tr12-30	13.0	0.1	0.3	0.2	0.2	0.2	0.4
gesa2_o	27.0	0.3	0.4	0.3	0.3	0.2	0.5
khb05250	41.1	1.5	1.9	1.7	1.6	1.5	2.6
p0033	28.6	8.4	7.7	4.9	5.0	4.7	8.7
misc03	30.4	4.0	10.6	3.9	4.0	3.9	10.9
egout	21.3	2.2	3.1	1.8	1.9	2.0	4.1
mas74	46.5	34.6	39.5	21.6	20.7	19.9	29.2
mitre	38.5	0.3	0.1	0.2	0.2	0.2	0.2
momentum1	11.5	5.0	0.1	0.1	0.0	0.2	2.3
gen	33.5	1.7	0.5	0.6	0.7	0.7	0.7
liu	1.2	0.1	0.1	0.1	0.1	0.1	0.1
fixnet6	33.4	1.2	3.1	1.4	2.0	1.7	3.3
gesa2	26.5	0.3	0.4	0.2	0.3	0.3	0.5
air03	48.5	7.3	4.0	2.1	2.4	2.9	22.1
fiber	35.7	0.6	2.0	0.5	0.5	0.5	1.1
p0201	31.9	11.3	10.4	4.0	4.1	3.9	10.8
mas76	46.8	29.6	44.6	21.1	18.4	18.0	30.0
modglob	25.4	1.0	3.8	1.1	1.2	1.1	1.4
vpm1	27.9	1.3	1.4	1.0	1.1	1.0	1.5
timtab2	9.6	0.2	0.6	0.2	0.2	0.2	0.5
pp08aCUTS	21.0	1.3	1.4	1.2	1.3	2.0	2.7
gesa3	25.2	0.8	0.6	0.6	0.7	0.7	1.2
markshare2	38.4	30.4	30.4	21.9	21.1	21.1	34.3
gesa2-o	27.6	0.3	0.3	0.2	0.4	0.3	0.5
dcmulti	24.6	3.7	2.6	2.0	2.2	2.2	3.6
rd-rplusc-21	1.3	0.1	0.1	0.0	0.0	0.1	0.1
swath	37.6	3.5	0.2	1.9	1.0	1.8	7.0
manna81	3.5	0.0	0.0	0.0	0.0	0.0	0.0
opt1217	44.1	27.9	14.3	5.1	5.8	6.0	8.3
flugpl	23.0	4.7	5.1	4.6	5.8	4.6	8.6
p0548	29.1	1.0	2.9	0.8	0.8	0.7	1.4
l152lav	45.4	30.7	7.0	15.3	15.7	15.9	27.8
mkc	39.8	1.6	1.2	0.8	0.7	0.7	1.9
protfold	8.1	6.5	0.8	2.6	5.9	1.8	17.4
roll3000	21.9	2.1	0.7	0.4	0.4	0.4	1.8
arki001	22.7	5.1	3.8	1.2	1.0	1.1	3.9
misc06	35.3	1.8	1.5	0.8	0.7	0.7	2.0
nsrand-ipx	46.7	1.9	1.8	2.4	2.1	2.1	6.8
a1c1s1	17.4	0.1	0.1	0.1	0.1	0.2	0.3
stein45	3.3	1.3	2.2	1.4	1.4	1.5	2.9
rout	34.5	8.4	6.9	3.4	4.1	3.9	15.2
pp08a	24.3	1.0	1.4	0.9	1.2	1.3	1.7
blend2	30.6	7.0	4.3	2.6	3.0	2.0	5.6
bell5	22.9	1.1	2.3	1.5	1.4	1.4	2.9
qiu	17.8	5.6	6.2	2.7	2.8	2.5	4.9
set1ch	24.4	0.2	0.2	0.2	0.2	0.2	0.3
mod011	27.1	0.5	0.1	1.2	1.2	2.1	1.8
mod010	37.4	9.8	3.1	4.9	5.2	5.2	13.8
cap6000	41.2	18.4	21.6	21.4	21.4	21.4	25.3
p0282	27.7	1.5	3.1	1.1	1.0	1.0	2.4
noswot	17.3	1.4	1.6	1.3	1.2	1.3	2.4
harp2	35.1	6.3	2.7	2.1	2.2	1.8	6.5
qnet1	35.6	1.3	5.7	0.8	0.9	0.8	2.6
seymour	9.3	0.2	0.1	0.2	0.2	0.2	3.6
vpm2	24.9	1.1	1.9	1.0	0.9	1.1	1.3
timtab1	9.5	0.3	1.0	0.3	0.3	0.4	0.7
net12	31.3	0.0	0.0	0.0	0.0	0.0	0.1
gt2	41.3	2.2	7.3	1.4	1.2	1.1	7.4
mod008	48.3	18.1	14.4	18.9	18.9	19.1	24.1
markshare1	38.3	31.5	32.9	22.6	22.6	22.6	34.0
danoint	23.1	21.8	10.0	1.4	2.0	2.0	14.7
momentum2	11.2	0.9	0.1	0.1	0.1	0.1	0.5
enigma	38.2	32.1	23.1	19.6	19.8	20.4	21.1
10teams	42.7	9.4	6.8	10.0	10.2	11.5	29.6

Table A.9: Arithmetic mean of relative dual density of cuts with rank 1 in %.

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	37.1	5.4	6.2	2.9	3.7	3.2	20.0
lseu	31.5	18.4	12.6	13.8	12.7	13.5	16.7
pk1	32.4	28.1	60.2	17.0	17.3	17.2	20.0
aflow40b	35.4	5.9	1.2	2.5	2.5	3.0	20.7
aflow30a	30.8	6.7	8.2	4.8	8.5	4.8	17.1
bell3a	25.7	19.6	65.2	4.8	5.5	4.8	6.4
rgn	30.9	11.1	54.7	3.9	3.8	3.8	10.5
gesa3_o	24.7	0.8	39.4	0.7	0.8	0.6	4.7
p2756	37.1	4.6	0.4	1.1	1.0	0.8	1.9
glass4	17.6	0.6	0.5	0.5	0.7	0.6	1.2
stein27	6.4	1.6	37.5	2.0	4.4	1.4	6.3
misc07	28.7	3.9	9.7	2.9	3.8	4.6	11.2
tr12-30	16.7	2.5	71.8	1.4	1.6	1.3	5.2
gesa2_o	26.3	0.6	71.1	0.5	0.6	0.5	1.1
khb05250	38.8	2.0	67.6	1.9	2.4	4.5	9.9
p0033	22.6	14.3	11.9	6.6	6.6	7.1	8.8
misc03	29.6	3.1	10.6	4.4	8.3	6.9	12.0
egout	20.0	3.0	56.8	2.4	2.7	4.1	5.1
mas74	46.0	30.5	79.3	22.5	22.7	21.8	29.9
mitre	29.4	0.3	0.1	0.1	0.2	0.2	0.3
momentum1	11.5	4.6	0.1	0.4	0.0	0.1	2.6
gen	32.9	2.6	0.2	0.5	0.5	0.6	0.7
liu	1.8	0.1	0.1	0.1	0.2	0.2	0.3
fixnet6	33.7	3.9	61.9	3.7	7.3	3.7	14.3
gesa2	25.5	1.9	79.6	0.5	0.9	0.5	1.8
air03	48.5	7.3	3.6	2.9	2.3	2.9	22.1
fiber	34.6	3.9	2.0	1.4	2.1	1.7	6.5
p0201	30.2	12.3	10.4	5.2	7.8	4.1	15.6
mas76	46.6	35.8	44.6	23.2	23.1	22.9	29.3
modglob	24.5	1.4	3.8	1.4	4.2	1.4	5.8
vpm1	24.8	3.0	1.4	1.0	1.5	1.4	2.1
timtab2	8.7	0.7	0.6	1.2	2.5	2.1	3.5
pp08aCUTS	20.0	1.7	86.7	1.9	2.8	3.2	9.1
gesa3	25.6	2.8	0.6	0.7	0.8	0.6	3.0
markshare2	37.0	33.7	33.5	23.1	23.1	23.1	33.6
gesa2-o	26.4	0.6	0.3	0.5	0.7	0.5	2.4
dcmulti	25.0	4.0	48.7	2.1	4.1	2.0	11.4
rd-rplusc-21	1.3	0.1	0.1	0.0	0.0	0.1	0.4
swath	40.3	2.4	0.2	1.1	1.1	1.9	9.5
manna81	3.3	0.0	0.0	0.0	0.0	0.0	0.0
opt1217	42.9	26.2	60.3	2.0	1.8	1.8	8.3
flugpl	17.0	7.3	91.8	7.4	7.9	7.5	7.7
p0548	27.4	1.2	60.6	0.8	0.8	0.8	1.3
l152lav	45.3	31.7	7.0	19.5	7.8	15.4	28.3
mkc	38.0	1.6	2.5	0.6	0.7	0.7	3.4
protfold	8.0	4.8	0.7	4.2	5.9	1.4	8.3
roll3000	21.2	3.5	0.7	0.5	0.5	0.4	3.3
arki001	22.9	7.6	3.8	2.4	2.2	2.0	9.0
misc06	32.8	1.9	1.5	0.7	0.6	0.8	2.7
nsrand-ipx	46.6	1.9	1.5	2.7	2.1	2.1	11.2
a1c1s1	17.4	0.3	0.1	0.6	2.1	1.5	5.4
stein45	5.0	1.3	2.2	1.7	3.2	1.4	3.9
rout	32.1	19.3	6.9	5.6	9.8	4.7	18.3
pp08a	22.6	1.7	14.7	1.6	2.4	2.0	6.9
blend2	29.8	16.3	29.6	5.0	6.6	5.6	6.0
bell5	19.5	3.7	2.3	3.2	3.9	3.0	3.8
qiu	18.9	6.5	39.4	3.7	4.2	2.9	4.9
set1ch	23.0	0.4	0.2	0.3	0.3	0.4	1.2
mod011	27.1	0.5	5.7	1.2	1.2	2.1	9.8
mod010	37.7	8.2	3.1	3.6	5.0	5.0	15.7
cap6000	47.5	34.4	21.6	14.8	21.4	21.4	25.3
p0282	26.8	4.6	3.1	3.0	3.5	3.2	4.8
noswot	16.6	1.7	31.6	1.1	2.3	1.4	3.2
harp2	36.6	15.4	3.4	2.0	1.9	2.3	17.8
qnet1	36.9	5.5	63.7	1.9	2.9	2.3	22.4
seymour	9.9	0.2	0.1	0.3	1.1	0.2	4.6
vpm2	22.9	4.4	1.9	2.0	3.2	2.0	10.2
timtab1	9.1	0.8	1.0	1.0	1.8	1.8	3.0
net12	31.6	0.0	0.0	0.0	0.0	0.0	0.6
gt2	38.3	17.0	7.3	9.8	8.8	11.3	15.6
mod008	47.7	65.0	85.5	24.2	24.5	24.5	37.9
markshare1	36.2	33.3	33.0	22.8	23.0	23.0	33.0
danoint	22.8	21.5	10.0	3.8	7.8	3.5	15.9
momentum2	11.2	0.9	0.1	0.1	0.1	0.1	0.3
enigma	37.1	34.3	24.8	18.5	19.1	18.5	19.7
10teams	43.2	10.8	6.6	7.6	9.4	9.7	28.6

Table A.10: Arithmetic mean of relative dual density of cuts with arbitrary rank in %.

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	101.0	623.0	11.0	554.0	301.0	702.0	437.0
lseu	68.0	177.0	8.0	168.0	152.0	183.0	64.0
pk1	111.0	117.0	33.0	128.0	127.0	115.0	120.0
aflow40b	483.0	844.0	38.0	841.0	710.0	731.0	509.0
aflow30a	344.0	680.0	31.0	688.0	621.0	589.0	373.0
bell3a	13.0	17.0	12.0	15.0	15.0	15.0	15.0
rgn	167.0	163.0	34.0	169.0	170.0	169.0	148.0
gesa3_o	690.0	994.0	994.0	1,000.0	995.0	997.0	844.0
p2756	207.0	310.0	100.0	276.0	278.0	284.0	212.0
glass4	436.0	631.0	145.0	414.0	632.0	485.0	423.0
stein27	137.0	145.0	101.0	144.0	137.0	144.0	136.0
misc07	169.0	372.0	16.0	419.0	439.0	429.0	161.0
tr12-30	437.0	993.0	500.0	983.0	917.0	900.0	415.0
gesa2_o	409.0	714.0	134.0	661.0	642.0	692.0	535.0
khb05250	97.0	109.0	111.0	96.0	107.0	110.0	95.0
p0033	36.0	77.0	49.0	85.0	82.0	73.0	48.0
misc03	128.0	543.0	12.0	442.0	434.0	425.0	245.0
egout	25.0	42.0	19.0	44.0	37.0	52.0	34.0
mas74	148.0	199.0	54.0	157.0	153.0	157.0	53.0
mitre	84.0	96.0	16.0	88.0	90.0	90.0	83.0
momentum1	96.0	98.0	100.0	1,000.0	98.0	1,000.0	557.0
gen	46.0	45.0	28.0	53.0	35.0	35.0	34.0
liu	502.0	600.0	500.0	596.0	584.0	598.0	472.0
fixnet6	147.0	325.0	113.0	305.0	236.0	255.0	155.0
gesa2	338.0	483.0	119.0	706.0	545.0	543.0	442.0
air03	35.0	35.0	6.0	4.0	1.0	21.0	35.0
fiber	355.0	640.0	46.0	635.0	594.0	632.0	495.0
p0201	520.0	637.0	22.0	680.0	651.0	647.0	582.0
mas76	142.0	171.0	11.0	147.0	148.0	148.0	126.0
modglob	307.0	475.0	30.0	430.0	401.0	329.0	352.0
vpm1	40.0	113.0	15.0	97.0	92.0	95.0	57.0
timtab2	695.0	1,000.0	100.0	999.0	992.0	992.0	773.0
pp08aCUTS	326.0	388.0	203.0	366.0	349.0	300.0	354.0
gesa3	620.0	974.0	80.0	980.0	980.0	978.0	740.0
markshare2	46.0	66.0	66.0	53.0	55.0	55.0	51.0
gesa2-o	401.0	732.0	73.0	754.0	566.0	690.0	531.0
dcmulti	497.0	511.0	97.0	479.0	434.0	443.0	528.0
rd-rplusc-21	73.0	332.0	17.0	393.0	20.0	409.0	431.0
swath	449.0	685.0	29.0	530.0	45.0	551.0	538.0
manna81	200.0	100.0	200.0	200.0	200.0	200.0	200.0
opt1217	158.0	249.0	65.0	235.0	234.0	224.0	190.0
flugpl	50.0	50.0	65.0	51.0	68.0	51.0	49.0
p0548	267.0	385.0	238.0	412.0	378.0	407.0	259.0
l152lav	397.0	670.0	36.0	812.0	772.0	810.0	727.0
mkc	705.0	975.0	175.0	975.0	960.0	972.0	762.0
protfold	77.0	1,000.0	9.0	1,000.0	92.0	1,000.0	1,000.0
roll3000	689.0	1,000.0	96.0	992.0	969.0	995.0	967.0
arki001	315.0	431.0	61.0	372.0	446.0	424.0	344.0
misc06	63.0	69.0	9.0	64.0	73.0	76.0	66.0
nsrand-ipx	716.0	68.0	63.0	909.0	65.0	68.0	773.0
a1c1s1	100.0	1,000.0	98.0	984.0	982.0	992.0	869.0
stein45	228.0	244.0	35.0	242.0	241.0	235.0	228.0
rout	240.0	573.0	35.0	490.0	500.0	531.0	252.0
pp08a	275.0	421.0	387.0	409.0	358.0	294.0	325.0
blend2	64.0	271.0	63.0	165.0	134.0	179.0	142.0
bell5	50.0	99.0	15.0	88.0	98.0	97.0	75.0
qiu	471.0	471.0	135.0	467.0	455.0	437.0	78.0
set1ch	374.0	739.0	100.0	722.0	667.0	609.0	349.0
mod011	16.0	16.0	38.0	467.0	460.0	16.0	468.0
mod010	285.0	510.0	29.0	509.0	500.0	500.0	574.0
cap6000	12.0	30.0	2.0	2.0	2.0	2.0	2.0
p0282	160.0	315.0	23.0	339.0	340.0	342.0	173.0
noswot	127.0	145.0	68.0	119.0	158.0	144.0	126.0
harp2	251.0	392.0	168.0	564.0	622.0	766.0	322.0
qnet1	339.0	854.0	121.0	750.0	702.0	862.0	532.0
seymour	1,000.0	1,000.0	100.0	1,000.0	1,000.0	1,000.0	1,000.0
vpm2	148.0	330.0	21.0	277.0	265.0	245.0	231.0
timtab1	693.0	1,000.0	100.0	996.0	1,000.0	989.0	736.0
net12	1,000.0	993.0	23.0	1,000.0	98.0	922.0	1,000.0
gt2	45.0	246.0	15.0	239.0	238.0	237.0	81.0
mod008	72.0	95.0	95.0	95.0	95.0	94.0	74.0
markshare1	50.0	45.0	45.0	44.0	44.0	44.0	47.0
danoint	532.0	249.0	40.0	527.0	517.0	520.0	529.0
momentum2	99.0	100.0	83.0	1,000.0	69.0	1,000.0	75.0
enigma	82.0	75.0	51.0	105.0	94.0	91.0	90.0
10teams	700.0	700.0	56.0	900.0	700.0	900.0	700.0

Table A.11: Number of generated cuts with rank 1.

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	732.0	664.0	11.0	601.0	734.0	608.0	730.0
lseu	183.0	173.0	8.0	196.0	171.0	181.0	185.0
pk1	114.0	112.0	30.0	113.0	115.0	117.0	111.0
aflow40b	348.0	842.0	38.0	857.0	598.0	860.0	844.0
aflow30a	667.0	717.0	31.0	697.0	691.0	712.0	658.0
bell3a	25.0	19.0	22.0	15.0	16.0	15.0	17.0
rgn	167.0	163.0	34.0	165.0	167.0	166.0	158.0
gesa3_o	971.0	994.0	192.0	965.0	1,000.0	924.0	937.0
p2756	291.0	258.0	100.0	281.0	282.0	284.0	281.0
glass4	471.0	523.0	92.0	446.0	206.0	597.0	497.0
stein27	158.0	144.0	47.0	144.0	140.0	144.0	136.0
misc07	150.0	364.0	16.0	436.0	381.0	512.0	148.0
tr12-30	877.0	987.0	684.0	1,000.0	1,000.0	1,000.0	998.0
gesa2_o	612.0	765.0	417.0	762.0	778.0	739.0	713.0
khb05250	101.0	109.0	61.0	102.0	105.0	100.0	105.0
p0033	93.0	99.0	19.0	75.0	61.0	76.0	92.0
misc03	187.0	497.0	12.0	433.0	365.0	452.0	203.0
egout	40.0	47.0	18.0	40.0	42.0	79.0	60.0
mas74	143.0	188.0	93.0	154.0	152.0	154.0	53.0
mitre	120.0	74.0	16.0	52.0	70.0	70.0	100.0
momentum1	93.0	89.0	100.0	1,000.0	100.0	1,000.0	403.0
gen	46.0	45.0	13.0	53.0	35.0	35.0	33.0
liu	600.0	600.0	200.0	600.0	600.0	600.0	600.0
fixnet6	280.0	316.0	30.0	359.0	302.0	276.0	359.0
gesa2	588.0	523.0	484.0	624.0	721.0	677.0	690.0
air03	35.0	35.0	4.0	15.0	3.0	20.0	35.0
fiber	652.0	564.0	46.0	611.0	635.0	638.0	618.0
p0201	547.0	642.0	22.0	624.0	641.0	638.0	557.0
mas76	128.0	170.0	11.0	150.0	149.0	149.0	150.0
modglob	382.0	515.0	30.0	471.0	464.0	469.0	438.0
vpm1	66.0	98.0	15.0	74.0	109.0	113.0	70.0
timtab2	1,000.0	1,000.0	100.0	1,000.0	1,000.0	1,000.0	1,000.0
pp08aCUTS	371.0	416.0	342.0	413.0	378.0	368.0	373.0
gesa3	699.0	916.0	80.0	980.0	980.0	980.0	872.0
markshare2	53.0	55.0	55.0	53.0	53.0	55.0	53.0
gesa2-o	635.0	773.0	73.0	743.0	821.0	805.0	876.0
dcmulti	519.0	530.0	91.0	460.0	428.0	481.0	550.0
rd-rplusc-21	197.0	288.0	18.0	627.0	20.0	430.0	471.0
swath	453.0	565.0	32.0	520.0	422.0	592.0	543.0
manna81	200.0	100.0	200.0	200.0	200.0	200.0	100.0
opt1217	242.0	259.0	63.0	248.0	248.0	248.0	193.0
flugpl	92.0	105.0	86.0	104.0	95.0	100.0	95.0
p0548	384.0	445.0	196.0	411.0	407.0	420.0	391.0
l152lav	580.0	698.0	36.0	743.0	116.0	793.0	671.0
mkc	975.0	975.0	170.0	975.0	975.0	975.0	975.0
protfold	85.0	1,000.0	1.0	1,000.0	98.0	1,000.0	995.0
roll3000	699.0	1,000.0	94.0	1,000.0	1,000.0	1,000.0	1,000.0
arki001	130.0	400.0	61.0	382.0	371.0	388.0	378.0
misc06	63.0	84.0	9.0	71.0	58.0	71.0	85.0
nsrand-ipx	856.0	68.0	49.0	941.0	68.0	68.0	911.0
a1c1s1	100.0	993.0	98.0	1,000.0	997.0	1,000.0	962.0
stein45	225.0	243.0	35.0	238.0	238.0	236.0	224.0
rout	332.0	561.0	35.0	543.0	577.0	500.0	316.0
pp08a	369.0	426.0	95.0	401.0	430.0	417.0	411.0
blend2	221.0	276.0	23.0	222.0	269.0	281.0	216.0
bell5	130.0	154.0	15.0	126.0	120.0	135.0	130.0
qiu	471.0	471.0	126.0	431.0	418.0	440.0	78.0
set1ch	559.0	678.0	100.0	647.0	674.0	642.0	728.0
mod011	16.0	16.0	35.0	473.0	454.0	16.0	311.0
mod010	275.0	346.0	30.0	460.0	440.0	505.0	484.0
cap6000	27.0	33.0	2.0	27.0	2.0	2.0	2.0
p0282	320.0	354.0	23.0	336.0	298.0	323.0	304.0
noswot	146.0	162.0	91.0	124.0	151.0	142.0	127.0
harp2	139.0	701.0	30.0	736.0	745.0	787.0	342.0
qnet1	710.0	852.0	117.0	885.0	803.0	811.0	854.0
seymour	1,000.0	1,000.0	99.0	1,000.0	849.0	1,000.0	1,000.0
vpm2	312.0	315.0	21.0	305.0	307.0	307.0	302.0
timtab1	1,000.0	1,000.0	100.0	1,000.0	1,000.0	1,000.0	1,000.0
net12	156.0	564.0	39.0	1,000.0	99.0	100.0	261.0
gt2	242.0	252.0	15.0	156.0	168.0	262.0	243.0
mod008	95.0	101.0	64.0	95.0	95.0	95.0	104.0
markshare1	54.0	45.0	45.0	44.0	44.0	44.0	45.0
danoint	531.0	247.0	40.0	517.0	512.0	515.0	531.0
momentum2	100.0	99.0	91.0	1,000.0	97.0	1,000.0	86.0
enigma	80.0	66.0	8.0	96.0	95.0	97.0	112.0
10teams	700.0	700.0	60.0	800.0	544.0	900.0	700.0

Table A.12: Number of generated cuts with arbitrary rank.

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	6.4	0.0	21.9	2.2	4.7	4.3	1.6
lseu	0.1	4.5	4.6	0.1	0.4	0.1	0.4
pk1	0.0	27.1	6.4	0.5	0.4	0.4	7.4
aflow40b	11.6	15.9	1.6	0.0	0.3	2.0	14.7
aflow30a	7.7	5.9	105.4	0.6	0.3	0.6	5.2
bell3a	14.5	4.9	2.4	0.9	1.1	0.3	1.1
rgn	6.7	7.4	17.6	0.3	0.4	0.2	5.2
gesa3_o	0.7	5.2	8.1	0.0	0.0	0.9	3.6
p2756	4.4	1.5	0.1	0.1	1.2	0.7	0.7
glass4	0.1	0.0	0.5	0.1	0.2	0.2	0.1
stein27	0.0	0.0	0.0	0.0	0.0	0.0	0.0
misc07	2.9	42.2	40.7	1.3	1.8	1.4	9.8
tr12-30	45.9	1.2	6.0	0.2	0.5	0.1	0.5
gesa2_o	0.3	1.6	2.7	0.1	0.1	0.2	2.6
khb05250	5.3	0.1	6.9	0.0	0.3	0.0	0.6
p0033	0.1	3.0	0.7	0.0	0.0	0.1	0.2
misc03	1.1	10.9	25.8	1.2	1.3	1.2	10.1
egout	0.4	0.3	0.5	0.6	0.0	0.1	0.6
mas74	0.8	11.5	6.1	1.5	0.4	0.0	13.0
mitre	13.7	14.9	1.1	0.0	0.1	0.1	0.2
momentum1	27.4	922.9	0.3	2.6	0.0	1.4	69.2
gen	1.6	8.6	0.1	0.3	0.1	0.0	0.2
liu	0.3	0.2	0.2	0.0	0.0	0.0	0.0
fixnet6	3.4	0.8	23.9	0.0	0.1	0.2	4.9
gesa2	1.3	3.5	0.8	0.3	0.3	0.0	0.6
air03	1.1	333.7	3.8	29.5	591.0	162.9	633.5
fiber	6.1	1.6	13.3	1.1	2.0	0.6	1.1
p0201	0.6	50.3	24.0	0.2	0.8	0.2	5.3
mas76	0.8	9.3	3.7	5.7	2.5	0.3	11.7
modglob	2.3	0.4	16.7	0.0	0.0	0.0	0.4
vpm1	2.3	2.2	0.3	0.3	0.1	0.4	0.1
timtab2	2.5	0.3	2.0	0.6	0.0	0.0	0.5
pp08aCUTS	4.8	1.3	0.7	0.0	0.4	0.0	2.6
gesa3	2.4	9.6	1.4	0.0	0.0	1.3	2.0
markshare2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
gesa2-o	0.4	1.8	1.8	0.1	0.1	0.0	2.0
dcmulti	7.2	81.9	4.3	2.6	0.0	0.0	5.5
rd-rplusc-21	4.4	11.1	0.1	0.1	0.0	0.0	1.3
swath	2.0	189.8	2.6	1.1	0.5	1.4	49.1
manna81	0.0	0.0	0.0	0.0	0.0	0.0	0.0
opt1217	0.3	87.9	5.6	3.8	0.0	1.3	0.5
flugpl	0.0	0.8	2.1	1.9	0.9	0.0	0.0
p0548	0.4	1.3	17.7	0.3	0.2	0.1	0.3
l152lav	0.6	561.7	2.2	30.6	32.3	24.9	177.5
mkc	2.7	957.2	178.0	0.1	0.0	0.8	176.0
protfold	2.0	407.2	78.4	2.5	62.8	1.3	173.2
roll3000	21.4	304.1	4.5	0.7	1.2	0.7	14.1
arki001	6.5	52.8	41.2	1.0	0.0	1.0	9.9
misc06	34.0	13.4	7.3	1.4	3.0	2.8	6.5
nsrand-ipx	0.0	12.3	1.7	2.6	0.2	0.2	0.4
a1c1s1	64.7	0.9	0.0	0.0	0.0	0.1	1.0
stein45	0.0	0.0	0.0	0.0	0.0	0.0	0.0
rout	10.8	54.1	30.4	0.7	1.6	1.4	27.8
pp08a	4.9	0.0	1.5	0.0	0.1	0.0	0.5
blend2	1.8	21.0	1.2	1.0	2.5	9.1	2.0
bell5	0.8	0.1	0.0	0.3	0.0	0.1	0.4
qiu	33.5	22.9	22.0	0.0	0.1	0.1	21.7
set1ch	12.5	0.4	0.2	0.0	0.5	0.0	0.4
mod011	11.4	33.3	0.0	0.0	0.0	0.0	3.2
mod010	1.2	361.5	1.4	3.8	11.2	10.4	245.6
cap6000	17.5	22.0	0.0	0.0	0.0	0.0	43.0
p0282	0.1	0.4	1.5	0.1	0.1	0.0	0.1
noswot	0.2	1.0	0.6	0.2	0.3	0.2	0.4
harp2	7.2	18.6	13.7	4.8	2.7	3.1	1.2
qnet1	30.3	11.4	41.5	1.9	6.6	2.6	8.8
seymour	0.0	0.0	0.0	0.0	0.0	0.0	0.0
vpm2	2.1	1.9	0.1	1.5	0.1	0.5	0.1
timtab1	1.8	0.1	2.2	0.4	0.0	0.0	0.3
net12	22.7	0.5	0.2	0.2	0.3	0.2	4.6
gt2	0.0	0.6	2.5	7.4	0.0	0.0	0.3
mod008	0.0	21.9	0.0	0.0	0.0	0.0	0.0
markshare1	0.0	0.0	0.6	0.0	0.0	0.0	0.0
danoint	3.1	209.1	72.6	0.6	1.1	0.8	81.1
momentum2	16.3	79.8	0.8	0.3	0.2	0.2	7.8
enigma	0.9	24.0	12.1	0.3	0.3	0.4	1.5
10teams	0.6	48.6	10.5	3.4	3.9	2.5	91.8

Table A.13: Number of cancellations (arithmetic mean) for cuts with rank 1.

Instance	α_∞-NC	α_1-NC	β-NC	DNC	ENC	SNC	TNC
qnet1_o	8.7	38.5	21.9	0.5	0.6	0.4	69.2
lseu	0.5	7.9	4.6	0.1	0.2	0.2	1.2
pk1	0.0	28.8	3.0	0.2	0.2	0.3	6.2
aflow40b	7.9	215.9	1.6	0.7	2.4	2.0	107.0
aflow30a	3.0	77.1	105.4	0.4	1.8	1.5	109.2
bell3a	3.3	7.8	0.3	0.0	0.0	0.0	0.6
rgn	1.4	34.2	10.8	0.2	0.5	0.9	12.1
gesa3_o	6.0	10.3	3.4	0.1	0.1	0.1	34.0
p2756	1.2	4.7	0.1	0.1	0.3	0.2	2.2
glass4	0.4	0.9	0.5	0.1	0.3	0.0	4.1
stein27	0.0	0.0	0.1	0.0	0.0	0.0	0.0
misc07	2.2	19.7	40.7	2.7	2.4	1.6	9.1
tr12-30	55.7	58.0	0.0	0.1	0.3	0.5	40.8
gesa2_o	1.8	8.1	1.2	0.5	0.1	0.1	6.7
khb05250	5.7	3.1	0.4	0.0	0.2	0.1	27.4
p0033	0.2	5.9	1.1	0.1	0.1	0.1	0.4
misc03	1.6	1.5	25.8	0.7	0.9	0.5	7.0
egout	0.2	0.9	0.2	0.0	0.0	0.0	0.4
mas74	2.4	11.9	0.6	0.0	0.0	0.0	13.5
mitre	14.0	15.1	1.1	0.0	0.0	0.0	4.8
momentum1	27.5	861.2	0.3	28.2	0.0	2.1	115.2
gen	1.6	24.9	0.7	0.0	0.0	0.0	0.2
liu	1.2	0.1	0.4	0.1	0.2	0.1	0.3
fixnet6	1.4	24.8	19.4	0.4	0.9	1.0	64.0
gesa2	1.7	9.9	0.9	0.0	1.0	0.0	12.3
air03	1.1	333.7	3.5	26.2	19.3	146.3	633.5
fiber	2.9	16.5	13.3	0.3	0.2	0.1	17.9
p0201	0.9	72.2	24.0	0.2	0.2	0.3	18.3
mas76	1.4	13.7	3.7	0.0	0.0	0.0	13.7
modglob	2.9	2.2	16.7	3.4	2.6	0.2	14.4
vpm1	2.4	4.1	0.3	0.0	0.1	0.0	0.1
timtab2	1.4	1.0	2.0	0.2	1.6	0.2	1.7
pp08aCUTS	1.7	3.4	0.1	0.2	0.3	0.1	10.6
gesa3	2.5	17.0	1.4	0.0	0.1	0.0	12.8
markshare2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
gesa2-o	2.0	8.3	1.8	0.0	0.1	0.1	17.0
dcmulti	5.8	109.7	1.9	0.1	0.1	0.2	54.0
rd-rplusc-21	5.9	8.8	0.1	0.6	0.0	0.4	30.2
swath	2.2	99.6	2.6	2.0	3.5	6.1	80.2
manna81	0.0	0.0	0.0	0.0	0.0	0.0	0.0
opt1217	0.2	396.0	1.2	0.0	0.0	0.0	6.9
flugpl	0.2	0.8	0.1	0.1	0.0	0.0	0.3
p0548	0.2	2.5	8.3	0.0	0.0	0.0	0.5
l152lav	0.6	479.4	2.2	2.0	1,035.4	22.6	73.7
mkc	11.1	995.9	120.7	0.0	0.0	0.0	156.7
protfold	1.8	378.6	17.0	2.1	61.4	1.5	217.8
roll3000	25.4	326.7	4.5	0.2	0.8	0.4	59.1
arki001	10.7	83.0	41.4	2.7	2.8	0.7	47.2
misc06	4.3	5.7	7.3	0.0	0.0	0.0	3.5
nsrand-ipx	0.0	12.3	1.4	0.1	0.2	0.2	2.3
a1c1s1	64.7	10.5	0.0	2.7	32.5	1.7	129.1
stein45	0.0	0.0	0.0	0.0	0.0	0.0	0.0
rout	8.4	154.3	30.4	0.3	1.6	1.1	82.1
pp08a	1.7	2.6	1.3	0.1	0.2	0.1	8.0
blend2	3.2	34.0	12.6	0.3	0.1	0.2	1.6
bell5	0.3	0.5	0.0	0.0	0.0	0.0	0.3
qiu	18.5	43.5	9.5	0.1	0.4	0.1	20.6
set1ch	8.6	1.6	0.2	0.0	0.1	0.1	3.2
mod011	11.4	33.3	0.0	0.0	0.4	0.0	307.5
mod010	0.8	184.6	1.4	4.7	8.1	11.1	232.4
cap6000	20.1	24.4	0.0	0.1	0.0	0.0	43.0
p0282	0.1	1.6	1.5	0.0	0.0	0.0	0.2
noswot	0.0	2.1	0.5	0.1	0.1	0.2	0.3
harp2	8.3	25.8	28.4	0.2	0.1	0.8	43.8
qnet1	6.3	29.2	25.7	0.2	0.7	0.1	103.2
seymour	0.0	0.0	0.0	0.1	0.4	0.0	0.0
vpm2	3.0	10.7	0.1	0.0	0.0	0.1	5.9
timtab1	0.9	0.5	2.2	0.0	0.2	0.1	2.2
net12	25.0	0.6	0.2	0.2	0.3	0.2	18.8
gt2	0.0	5.8	2.5	0.0	0.0	0.0	1.7
mod008	0.0	0.0	0.0	0.0	0.0	0.0	0.0
markshare1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
danoint	1.7	216.1	72.6	0.4	1.7	0.5	67.5
momentum2	16.5	75.4	0.8	0.8	0.2	0.3	5.4
enigma	0.3	29.4	15.0	0.2	0.3	0.4	4.3
10teams	0.8	60.0	10.1	3.4	3.7	2.8	69.5

Table A.14: Number of cancellations (arithmetic mean) for cuts with arbitrary rank.

Bibliography

[1] The SCIP developers: Papers and talks dealing with SCIP, 2011. URL http://scip.zib.de/further.shtml.

[2] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[3] T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):361–372, 2006.

[4] T. Achterberg, T. Berthold, T. Koch, and K. Wolter. Constraint integer programming: a new approach to integrate CP and MIP. In *Proceedings of the 5th international conference on Integration of AI and OR techniques in constraint programming for combinatorial optimization problems*, CPAIOR'08, pages 6–20, Berlin, Heidelberg, 2008. Springer-Verlag.

[5] K. Andersen and G. Cornuéjols. Reduce-and-split cuts: Improving the performance of mixed-integer gomory cuts. *Management Science*, 51:2005, 2005.

[6] K. Andersen and R. Weismantel. Zero-coefficient cuts. In F. Eisenbrand and F. B. 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 57–70. Springer, 2010.

[7] E. Balas. Intersection cuts - a new type of cutting planes for integer programming. In *Operations Research*, 1971.

[8] E. Balas. Disjunctive programming: Properties of the convex hull of feasible points. MSRR 348, Carnegie Mellon University, 1974.

[9] E. Balas. Disjunctive programming. In *Annals of Discrete Optimization*, volume II, pages 3–51, Amsterdam, Netherlands, 1979.

[10] E. Balas. A modified lift-and-project procedure. *Math. Program.*, 79:19–31, 1997.

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

[12] E. Balas and M. Perregaard. Lift-and-project for mixed 0-1 programming: recent progress. *Discrete Appl. Math.*, 123:129–154, November 2002.

[13] E. Balas and M. Perregaard. A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts for 0-1 programming. *Math. Program.*, 94(2-3):221–245, 2003.

[14] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Program.*, 58:295–324, 1993.

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

[16] T. Berthold. Primal heuristics for mixed integer programs. Master's thesis, 2006.

[17] P. Bonami. On optimizing over lift-and-project closures. 2010.

[18] A. Caprara and A. N. Letchford. On the separation of split cuts and related inequalities. *Mathematical Programming*, 94:279–294, 2001.

[19] S. Ceria. A brief history of lift-and-project. *Annals OR*, 149(1):57–61, 2007.

[20] S. Ceria and G. Pataki. Solving integer and disjunctive programs by lift and project. In R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, editors, *IPCO*, volume 1412 of *Lecture Notes in Computer Science*, pages 271–283. Springer, 1998.

[21] S. Ceria and J. Soares. Disjunctive cut generation for mixed 0-1 programs: Duality and lifting, 1997.

[22] A. Charnes and W. W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3-4):181–186, 1962.

[23] G. Cornuéjols. Valid inequalities for mixed integer linear programs. *Mathematical Programming B*, 112:2008, 2006.

[24] G. Cornuéjols and Y. Li. Elementary closures for integer programs. *Operations Research Letters*, 28(1):1 – 8, 2001.

[25] J. Edmonds. Systems of distinct representatives and linear algebra. 1967.

[26] M. Fischetti and A. Lodi. Optimizing over the first chvátal closure. *Math. Program.*, 110(1):3–20, 2007.

[27] M. Fischetti, A. Lodi, and A. Tramontani. On the separation of disjunctive cuts. *Math. Program.*, 128:205–230, June 2011.

[28] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Duxbury Press, Nov. 2002.

[29] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[30] P. V. Hentenryck. The OPL optimization programming language, 1999.

[31] L. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk*, 244, 1979.

[32] T. Koch. ZIMPL user guide. Technical report, Konrad-Zuse-Zentrum für Informationstechnik, 2001.

[33] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991.

[34] C. M. Mczeal, M. W. P. Savelsbergh, and R. E. Bixby. An updated mixed integer programming library: MIPLIB 3.0. 1996.

[35] H. D. Mittelmann. Benchmarks for optimization software, 2011. URL `http://plato.asu.edu/bench.html`.

[36] M. Perregaard. Lift-and-project cuts: Properties of the cut LP, 1998.

[37] M. Perregaard. Generating disjunctive cuts for mixed integer programs. In *Mellon University*, 2003.

[38] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, Inc., New York, NY, USA, 1986.

[39] H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. 3(3):411–430, 1990.

[40] H. D. Sherali and W. P. Adams. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero–one programming problems. *Discrete Applied Mathematics*, 52(1):83 – 106, 1994.

[41] K. Wolter. Implementation of cutting plane separators for mixed integer programs. Master's thesis, 2006.

Erklärung

Hiermit erkläre ich, Matthias Walter, dass die vorliegende Diplomarbeit von mir selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht sind. Die vorliegende Diplomarbeit wurde nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt.

Ort, Datum *Unterschrift*