Non-Recursive Cut Generation

Aleksandr M. Kazachkov

Spring 2018

Tepper School of Business
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Egon Balas (Chair)
Daniel Bienstock
Gérard Cornuéjols
François Margot

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Algorithms, Combinatorics, and Optimization.*

This work was supported by the National Science Foundation under grants CMMI-1263239 and CMMI-1560828, and by the Office of Naval Research under contracts N00014-12-1-0032 and N00014-15-1-2082.

Abstract

There has recently been reinvigorated interest in finding new general-purpose cutting planes (cuts) for mixed-integer programs, particularly ones that can be derived from valid general disjunctions. This dissertation focuses on the theoretical and computational development of new disjunctive cuts. The motivation is primarily practical: cuts are a critical component of modern integer programming solvers, but the families of cuts that are currently implemented are relatively simple and suffer from numerical instability and the "tailing off" effect when used recursively. We introduce three techniques for efficiently generating a large number of strong cuts without recursion, prove useful theoretical properties of the cuts, and perform computational testing of their performance through implementations in the open source COIN-OR framework. Our evaluation criteria are strength (compared to Gomory cuts) and time (via branch-and-bound tests).

In our first contribution, we investigate *generalized intersection cuts*, a relatively recent approach with attractive theoretical properties. We first observe that a key hyperplane activation procedure embedded in this previously computationally unexplored paradigm is not computationally viable. We overcome this issue by a novel technique called *partial hyperplane activation* (PHA), introduce a variant of PHA based on a notion of hyperplane tilting, and prove the validity of both algorithms. We propose several implementation strategies and parameter choices for our PHA algorithms and provide supporting theoretical results. The accompanying computational findings shed light on the strengths of the PHA approach and identify properties related to strong cuts that we subsequently use.

The next chapter examines a notion of tilting that can be used to produce cutting planes from split disjunctions starting from any given valid inequality. Geometrically, this tilting involves changing the angle of the given inequality until it becomes supporting for both sides of the disjunction. We provide computational experience with the strength and practicality of this procedure and make connections with existing literature for which tilting offers a unifying perspective.

We then introduce *$\mathcal{V}$-polyhedral cuts* for generating valid inequalities from general disjunctions. We show how to efficiently obtain points and rays from which we build a linear program whose feasible solutions correspond to valid disjunctive cuts. This linear program is much smaller than the one from the alternative approach, lift-and-project, enabling us to test larger disjunctions that arise from the leaf nodes of a partial branch-and-bound tree. We show that the cuts from one strong disjunction significantly improve the gap closed compared to cuts produced from a union of many shallow disjunctive sets. Our cuts also decrease Gurobi solving time. However, this hinges on choosing the best partial tree size per instance, which remains an open problem and motivates future work on better understanding the interaction between branch-and-bound and cuts.

Finally, the last chapter builds a correspondence between lift-and-project and $\mathcal{V}$-polyhedral cuts. The motivation is finding the necessary ingredients to apply standard cut strengthening techniques such as modularization to the more-efficiently generated $\mathcal{V}$-polyhedral cuts.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

Notation	Description
CGLP	cut-generating linear program
GIC	generalized intersection cut
GMIC	Gomory mixed-integer cut
L&PC	lift-and-project cut
MLP	membership linear program
PHA	partial hyperplane activation
PRLP	point-ray linear program
SIC	standard intersection cut
VPC	$\mathcal{V}$-polyhedral cut

Chapter 1

Introduction & Background

Mixed-integer linear programs are optimization models for settings that involve discrete decisions and are prevalent in a wide range of real-world applications (see, e.g., the references in [120]). Cutting planes, or cuts, are one of the most important components enabling the fast performance of modern integer programming solvers [6]. There has been significant research effort in the past decade devoted to finding new classes of cuts, prompted by a combination of their practical significance and reinvigorated theoretical interest spurred by Andersen et al. [9]. The dissertation focuses on the theoretical development and computational investigation of new general-purpose cuts for mixed-integer linear programs, particularly approaches through which a large number of "good" cuts can be generated quickly and non-recursively. The motivation for avoiding recursive applications of cuts is empirical evidence that such recursion leads to numerical issues (such as due to compounding inaccuracies) and a "tailing off" of the strength of the cuts in later rounds [27, 60, 118, 139].

Despite an abundance of recent work on cuts providing illuminating theory, the computational experience on benefits over existing techniques has been mixed. In this dissertation, we identify some of the computational challenges associated with designing new cuts, introduce techniques aimed at resolving these challenges, present promising computational evidence for the practical significance of the methods, and support the experimental evidence with new theoretical results.

The cuts developed in the earlier chapters lead to the dissertation's primary contribution in Chapter 4, which is a $\mathcal{V}$-polyhedral framework for generating cuts from very strong disjunctions cuts in an efficient way, without resorting to recursion; the accompanying computational results indicate that the cuts already have strong promise for practical settings, though their effectiveness in deployed solvers would drastically increase with a better understanding of the interaction between cuts and branch-and-bound.

We first introduce, in Section 1.1, the mixed-integer linear programming setting more formally, give the notation used throughout the dissertation, and briefly survey the existing literature on cutting planes. In the subsequent sections, we discuss how we evaluate cuts, provide basic concepts for some common cuts, and review strengthening techniques. We then summarize the contributions of this dissertation in Section 1.6. Finally, in Section 1.7, we provide a table referencing notation that will be used frequently throughout.

1.1 Preliminaries

We assume some familiarity with polyhedra and linear and integer programming; good references for more details are the books *Integer Programming* by Conforti et al. [58] and *Lectures on Polytopes* by Ziegler [140]. The set of real numbers will be denoted by $\mathbb{R}$ and the set of integers by $\mathbb{Z}$. As shorthand, for any positive integer n, we use $[n]$ to denote the set of integers $\{1, \ldots, n\}$.

Let P denote a *polyhedron* described by a set of inequalities:[1]

$$P := \{x \in \mathbb{R}^n : A_{i\cdot} x \geq b_i,\ i \in [m]\},$$

where $A_{i\cdot}$ refers to row i of the constraint matrix A, and let

$$P_I := \{x \in P : x_j \in \mathbb{Z} \text{ for all } j \in \mathcal{I}\},$$

where $\mathcal{I} \subseteq [n]$ is the index set of the integer-restricted variables. We assume that P is full dimensional and pointed,[2] all data is rational, and all variable bounds are subsumed by $Ax \geq b$. For a given $c \in \mathbb{Q}^n$, our goal is to solve the *mixed-integer program*

$$\min_{x \in P_I} c^{\mathsf{T}} x. \tag{IP}$$

The decision version of integer programming is one of the classical $\mathcal{NP}$-complete questions [105]. Consequently, it is believed that, in general, there is no efficient algorithm (with running time that is polynomial in the size of the input) to optimize over P_I directly. On the other hand, one can optimize over P efficiently. As a result, integer programs are solved by starting with the *linear programming relaxation* of (IP), obtained by removing the integrality restrictions on the variables:

$$\min_{x \in P} c^{\mathsf{T}} x. \tag{LP}$$

Let $\bar{x}$ be an optimal solution to (LP). If $\bar{x}$ is integer feasible, then it is also an optimal solution to (IP). Otherwise, when $\bar{x} \notin P_I$, the next step in an integer programming solver is to add inequalities to P to produce a tighter relaxation of P_I. Inequalities are called *valid* (for P_I) if they are satisfied by all points in P_I. *Cutting planes*, or cuts, are valid inequalities for P_I that remove some part of P. Cuts can either be *problem specific* or *general purpose.* The former refers to cuts that only apply to specific classes of integer programming problems, whereas the latter cuts apply generically and will be the focus of this dissertation. All general-purpose cuts can be generated via the use of *disjunctions* that reintroduce some of the integrality information discarded when creating P from P_I; we elaborate on this below.

Suppose we are given the disjunction, indexed by a finite set $\mathcal{T}$,

$$\bigvee_{t \in \mathcal{T}} \{x \in \mathbb{R}^n : D^t x \geq D_0^t\}. \tag{1.1}$$

[1]We consider only convex polyhedra, so we drop the qualifier "convex".

[2]That is, $\operatorname{rank}(A) = n$ and P does not contain a line, or, equivalently, its smallest nonempty *face* has dimension 0, which is called a *vertex* or *extreme point.* A face of P of dimension $k-1$ is a hyperplane that contains exactly k affinely independent points from P on it.

Incorporating the constraints of P, we denote *disjunctive term* $t \in \mathcal{T}$ by

$$P^t := \{x \in P : D^t x \geq D_0^t\}.$$

Let $P_D := \operatorname{cl}\operatorname{conv}(\cup_{t\in\mathcal{T}} P^t)$ be the *disjunctive hull*, the *closed convex hull* of the points of P satisfying the disjunction. We assume the disjunction satisfies $P_I \subseteq P_D$ and $\bar{x} \notin P_D$. The theory of optimizing over disjunctions was developed by Balas [17] as part of *disjunctive programming*. The strongest cuts that are derived from a valid disjunction are *facet-defining* inequalities for P_D.[3]

The majority of the cuts currently implemented in mixed-integer programming solvers are generated from *split disjunctions*, which take the form $(\pi^\mathsf{T} x \leq \lfloor \pi^\mathsf{T}\bar{x} \rfloor) \vee (\pi^\mathsf{T} x \geq \lceil \pi^\mathsf{T}\bar{x} \rceil)$, where $\pi \in \mathbb{Z}^n$. Such cuts are referred to as *split cuts* [61] and include *Gomory mixed-integer cuts* (GMICs) [88] (equivalent to mixed-integer rounding cuts [120]) and *lift-and-project cuts* (L&PCs) [24]. The most commonly used split disjunction is of the type $(x_j \leq \lfloor \bar{x}_j \rfloor) \vee (x_j \geq \lceil \bar{x}_j \rceil)$, called a disjunction from an *elementary split* (sometimes called a *simple split*).

There are two notable properties of split disjunctions when considering generalizations. The first property is that split disjunctions arise from a single equation of the type $\pi^\mathsf{T} x = \pi^\mathsf{T} \bar{x}$, and in the case of an elementary split disjunction, this equation only utilizes the integrality of one integer variable to produce the disjunction. The second property is that each term of an elementary split disjunction is defined by exactly one inequality. Disjunctions of this type are called *simple*, and they are in one-to-one correspondence with cuts from *P_I-free convex sets*, which are convex sets, obtained by reversing the inequalities defining the terms of the simple disjunction, that contain no feasible integer points in their relative interior.

A natural generalization of the first property is to look at P_I-free convex sets arising from information from two equations, such as from two rows of the optimal simplex tableau to (LP) for which the basic variables are integer but take fractional values in $\bar{x}$. Such convex sets are triangles and quadrilaterals (in two dimensions), which are known to be all the maximal *lattice-free* (containing no integer points) convex sets in the plane [15].[4] The computational results are mixed on practical instances [34, 37, 72, 79] despite theoretical promise [16, 38]. As a partial explanation of this, some probabilistic results show that, on average, triangle cuts and quadrilateral cuts may not be much stronger than split cuts [39, 71, 97]. This is in contrast to the theory, which shows that splits cuts can be, in the worst-case, arbitrarily worse than triangle cuts [38]. Additionally, all split cuts are in fact equivalent to *intersection cuts* [21, 32], but this is no longer true for other disjunctive sets, for which there may exist *irregular cuts* that may dominate all the possible intersection cuts [8, 30]

There has been work relaxing the second property as well, in which cuts are generated from nonconvex sets, i.e., general (not necessarily simple) disjunctions [9, 49, 54, 66, 68, 69, 70, 114]. Every cut from a general disjunction can, in fact, be obtained from a simple one, corresponding to a P_I-free convex set that is entirely contained in $\mathbb{R}^n \setminus \cup_{t\in\mathcal{T}} \{x : D^t x \geq D_0^t\}$, the complement of the disjunction (1.1). However, the facet-defining inequalities for P_D may each be a consequence of a different P_I-free convex set, implying that using the general

[3]A *facet* is a *proper* face of maximal dimension (i.e., not equal to the entire polyhedron). A *facet-defining* inequality for P_D is one that is valid and produces a facet when set to equality and intersected with P_D. Keeping with convention, we often refer to facet-defining inequalities as facets themselves.

[4]The notion of *maximality* is that the lattice-free set is not properly contained in any other lattice-free set. Maximal lattice-free sets are known to be polyhedra [115].

disjunction avoids the complicated step of choosing a convex set derived from the disjunction. We discuss existing computational results with general disjunctions in Section 4.1.

When considering ways to generalize split disjunctions, the problem of too many choices arises. There are many options of selecting the disjunctions used to generate cuts, a variety of approaches to obtaining cuts from the disjunctions, and a much larger pool of potential cuts, which can make it more difficult to efficiently find the "good" cuts (supposing they exist). For example, when computing with two-row cuts, if there are n possible rows to choose from, there are $O(n^2)$ possible pairs of rows that can be selected, which is already computationally impractical. This is reflected in computational experiments; for example, Dash et al. [70] find strong *cross cuts* (a special case of *t-branch split cuts* [110]) but conclude that their separation procedure is too computationally demanding to be practical. The hurdles faced when designing more complex cut generators can also, by contrast, partially explain the practical effectiveness of split cuts: split cuts have the benefits of being simple, which allows them to be separated quickly, yet simultaneously quite strong, which is documented both theoretically [39] and empirically [35].

Cuts have repeatedly been shown to be critical in enabling the capabilities of modern solvers [6, 45, 46, 64]. This is one motivation for pursuing new cuts. Another is the pursuit of improved numerical stability. The success of cuts hinges on their application in rounds, by applying a set of cuts, reoptimizing, then recursing [26]. However, Gomory cuts are known to become increasingly parallel to each other as the number of rounds increases, leading to increasing dual degeneracy (the existence of multiple optimal solutions) and constraint matrices with large determinants (slowing down computation time and decreasing accuracy) [60, 139]. On the other hand, there is evidence that stronger inequalities do not suffer from this "tailing-off" effect [24, 59]. This is why one of the goals of the cuts introduced in this dissertation is to produce a multitude of strong cuts without resorting to recursion.

The challenge undertaken by the recent research on cuts, and this dissertation, is efficiently finding still stronger cuts that interact favorably with the existing ones and produce marginal benefit in solving time that offsets the extra effort to generate the cuts.

1.2 Cut evaluation

A new cut family can be evaluated, computationally, along different dimensions. The primary goal is improved empirical solver performance. However, it can be difficult to set up a fair test environment for this metric. As a result, as proxies, other criteria are often employed. Most commonly, measures of the strength of the generated cuts are reported. Another relevant quantity is the numerical stability of the cuts. Along with these metrics for the existence of "good" cuts within the family being tested, the computational expense of generating the new cuts needs to be considered. We assess all four of these aspects (to varying degrees) for each family of new cuts we introduce in this thesis, i.e., whether the new cuts

(1) are different from (and stronger than) existing cuts,

(2) result in better solution times,

(3) have desirable numerical properties, and

(4) can be found efficiently (the cuts that are strong, helpful, and numerically stable).

We often report the *percent integrality gap closed* to assess strength. Let x_I denote an optimal solution to (IP), and let $\widetilde{x}$ be an optimal solution to (LP) after a set of cuts have been added. We measure the quantity

$$\% \text{ integrality gap closed} := 100 \times \frac{c^\mathsf{T}\widetilde{x} - c^\mathsf{T}\bar{x}}{c^\mathsf{T}x_I - c^\mathsf{T}\bar{x}}.$$

Our baseline, as is common in the literature, is typically the percent of the integrality gap that is closed by either *standard intersection cuts* (SICs) or GMICs.

1.3 Standard intersection cuts, Gomory mixed-integer cuts, and the nonbasic space

We first prove the formula for the SIC from any valid simple disjunction. This is done in the *nonbasic space*, consisting of the variables that are nonbasic[5] at $\bar{x}$, a concept we will again utilize in Chapters 2 and 4 as it confers computational advantages over the *structural space*, referring to the x variables (not including slack variables). We then derive GMICs, usually simply referred to as Gomory cuts, from a geometric perspective, by showing how they arise via a strengthening of SICs. In the process, we show how, in some sense, Gomory cuts optimally take advantage of nonbasic integer variables when taking a split disjunction.

For this section, we work with a modified polyhedron $\widetilde{P}$ in which nonnegative slack variables $s_{n+1}, \dots, s_{n+m}$ have been added, one per constraint of P, and append to the indices $\mathcal{I}$ (of integer variables) the indices corresponding to slacks on rows that are bounds on integer-restricted variables. Concretely, with $\mathbb{R}_{\geq 0}$ denoting the set of nonnegative real numbers,

$$\widetilde{P} := \left\{ (x,s) \in \mathbb{R}^n \times \mathbb{R}^m_{\geq 0} : \sum_{j=1}^{n} A_{ij}x_j - s_{n+i} = b_i \text{ for all } i \in [m] \right\}.$$

We assume, without loss of generality, that variables $x_1, \dots, x_n$ are all basic and that $\bar{s}_j = 0$ for all $j \in \mathcal{N}$, where $\bar{s} = A\bar{x} - b$. Let $\mathcal{B}$ be the index set of the basic variables for a vertex $\bar{x}$ of $\widetilde{P}$ and $\mathcal{N}$ denote the *cobasis*, the associated index set of the nonbasic variables. Let the index set of the rows with nonbasic slacks be $N := \{i \in [m] : n + i \in \mathcal{N}\}$. Define A_N as the $n \times n$ nonsingular submatrix of A with rows indexed by N. These rows define the *LP basis cone* $C(\mathcal{N}) := \{x \in \mathbb{R}^n : A_N x \geq b_N\}$. This cone has n affinely independent rays, one for each $j \in \mathcal{N}$, whose intersections with the disjunctive terms define the intersection cut.

Although the cuts in this section are derived in the nonbasic space, converting to the structural space is an easy operation, as we show in Lemma 1.1.

Lemma 1.1. *Consider a cut in the full (structural and slack) space with coefficients γ_j, $j \in [n+m]$ and right-hand side γ_0. The equivalent cut in the structural space is $\alpha^\mathsf{T}x \geq \beta$, where $\alpha_k = (\gamma_k + \sum_{i=1}^{m} \gamma_{n+i}A_{ik})$ for all $k \in [n]$, $\alpha_k = 0$ for all $k \in \{n+1, \dots, n+m\}$, and $\beta = \gamma_0 + \sum_{i=1}^{m} \gamma_{n+i}b_i$.*

[5]The set of indices of the nonbasic variables, also called the *cobasis*, corresponds to a set of n linearly independent inequalities of P, which define a vertex of P.

Proof. Observe that, for any $(x, s) \in \mathbb{R}^{n+m}$,

$$\gamma^\mathsf{T} \left[\begin{smallmatrix} x \\ s \end{smallmatrix} \right] - \gamma_0 = \sum_{k=1}^{n} \gamma_k x_k + \sum_{k=n+1}^{n+m} \gamma_k s_k - \gamma_0 = \sum_{k=1}^{n} \gamma_k x_j k + \sum_{i=1}^{m} \gamma_{n+i} \left(\sum_{k=1}^{n} A_{ik} x_k - b_i \right) - \gamma_0$$

$$= \sum_{k=1}^{n} \left(\gamma_k + \sum_{i=1}^{m} \gamma_{n+i} A_{ik} \right) x_k - \left(\gamma_0 + \sum_{i=1}^{m} \gamma_{n+i} b_i \right).$$

Therefore, we get the desired result by substitution. □

1.3.1 Standard intersection cuts (simple disjunctive cuts)

To tighten the linear relaxation $\widetilde{P}$, we derive valid cuts from a simple (one inequality per term) disjunction, expressed in the nonbasic space of variables:

$$\bigvee_{t \in \mathcal{T}} \left(\sum_{j \in \mathcal{N}} d^t_j s_j \geq d^t_0 \right).$$

Suppose that $\bar{x}$ violates each disjunctive term inequality. In this case, $d^t_0 > 0$ for all $t \in \mathcal{T}$ because $\bar{s} = 0$, i.e., $\bar{x}$ is the origin in the nonbasic space, and the disjunction can be rewritten

$$\bigvee_{t \in \mathcal{T}} \left(\sum_{j \in \mathcal{N}} \frac{d^t_j}{d^t_0} s_j \geq 1 \right).$$

From this, we can derive the standard intersection cut (SIC) (equivalent to Owen's cut [121]),

$$\sum_{j \in \mathcal{N}} \max_{t \in \mathcal{T}} \left\{ \frac{d^t_j}{d^t_0} \right\} s_j \geq 1.$$

To prove its validity, take any $(x, s) \in \widetilde{P}^{\hat{t}} := \{(x, s) \in \widetilde{P} : \sum_{j \in \mathcal{N}} d^{\hat{t}}_j s_j \geq d^{\hat{t}}_0\}$ for any fixed $\hat{t} \in \mathcal{T}$. Then $\sum_{j \in \mathcal{N}} d^{\hat{t}}_j s_j \geq d^{\hat{t}}_0$ by definition of $\widetilde{P}^{\hat{t}}$, and we conclude that

$$\sum_{j \in \mathcal{N}} \max_{t \in \mathcal{T}} \left\{ \frac{d^t_j}{d^t_0} \right\} s_j \geq \sum_{j \in \mathcal{N}} \frac{d^{\hat{t}}_j}{d^{\hat{t}}_0} s_j \geq 1.$$

We can also generalize this derivation for non-simple disjunctions such as (1.1). This disjunction can be expressed in the nonbasic space through the following transformation. Define b_N as the elements of b indexed by N, and define $s_\mathcal{N}$ as the nonbasic slack variables. We multiply the rows $A_N x - s_\mathcal{N} = b_N$ by A_N^{-1}, and, using the fact that $A_N \bar{x} = b_N$, we get a version of the optimal tableau:

$$A_N^{-1} A_N x - A_N^{-1} s_\mathcal{N} = A_N^{-1} b_N \quad \implies \quad x = \bar{x} + A_N^{-1} s_\mathcal{N} = \bar{x} + \sum_{j \in \mathcal{N}} r^j s_j. \tag{1.2}$$

Note that the components of the rays of $C(\mathcal{N})$ (in the structural space) are given by A_N^{-1}. We then substitute (1.2) into (1.1) to get the disjunction represented in the nonbasic space:

$$\bigvee_{t \in \mathcal{T}} (D^t x \geq D^t_0) \iff \bigvee_{t \in \mathcal{T}} (D^t(\bar{x} + A_N^{-1} s_\mathcal{N}) \geq D^t_0) \iff \bigvee_{t \in \mathcal{T}} (D^t A_N^{-1} s_\mathcal{N} \geq D^t_0 - D^t \bar{x}). \tag{1.3}$$

Lemma 1.2. *For each $t \in \mathcal{T}$, let $u_0^t \in \mathbb{R}_{\geq 0}^{m_t}$, and define $d_j^t := (u_0^t D^t A_N^{-1})_{j-n}$ for all $j \in \mathcal{N}$ and $d_0^t := u_0^t(D_0^t - D^t \bar{x})$. The following cut is valid for P_D:*

$$\sum_{j \in \mathcal{N}} \max_{t \in \mathcal{T}} d_j^t s_j \geq \min_{t \in \mathcal{T}} d_0^t.$$

Proof. For each $\hat{t} \in \mathcal{T}$ and all $x \in P^{\hat{t}}$, we have that $u_0^{\hat{t}} D^{\hat{t}} x \geq u_0^{\hat{t}} D_0^{\hat{t}}$. Hence, setting $s = Ax - b$,

$$\sum_{j \in \mathcal{N}} \max_{t \in \mathcal{T}} d_j^t s_j \geq \sum_{j \in \mathcal{N}} d_j^{\hat{t}} s_j \geq d_0^{\hat{t}} \geq \min_{t \in \mathcal{T}} d_0^t. \qquad \square$$

We will still call the cut from Lemma 1.2 a SIC (it is sometimes also called a *simple disjunctive cut*), but the formula is more general than before; when d_0^t as in the lemma is positive for all $t \in \mathcal{T}$, we can use u_0^t / d_0^t (instead of u_0^t) as the multipliers for each disjunctive term to recover the earlier formula.

A geometric perspective. The SIC has a nice geometric interpretation. It is the unique inequality that passes through a set of n affinely independent intersection points (or rays) obtained by following each ray r^j of $C(\mathcal{N})$ until the first disjunctive term it intersects, or, if there is no such term, following $-r^j$ until the *last* disjunctive term it intersects. We can also look at the convex P_I-free set corresponding to each simple disjunction:

$$S := \{(x, s) \in \mathbb{R}^n \times \mathbb{R}_{\geq 0}^m : \sum_{j \in \mathcal{N}} d_j^t s_j \leq d_0^t \text{ for all } t \in \mathcal{T}\}.$$

The SIC is a valid inequality for $\operatorname{cl} \operatorname{conv}(C(\mathcal{N}) \setminus \operatorname{int} S)$, where $\operatorname{int} S$ denotes the *interior* of S. When S is lattice-free, the geometric interpretation is even more appealing: the SIC is simply obtained by intersecting the rays of $C(\mathcal{N})$ with the *boundary* of S, denoted $\operatorname{bd} S$.

More concretely, let $j \in \mathcal{N}$ and consider the corresponding ray r^j of $C(\mathcal{N})$. The value of s_j corresponds to a distance along r^j (emanating from $\bar{x}$). Thus, for each $t \in \mathcal{T}$ such that $d_j^t \neq 0$, setting $s_j = d_0^t / d_j^t$, while keeping the rest of the nonbasic variables at zero, yields a point that satisfies $d^t s = d_0^t$. We call this the *intersection point* of r^j with disjunctive term t. Note that the value d_0^t / d_j^t may be negative; this happens when we intersect the *negative extension* of r^j with term t. For the terms in which $d_j^t = 0$, neither the ray r^j nor its negative extension intersect the term, and an "intersection point" does not exist, but we can instead use the ray r^j itself. For every ray r^j of $C(\mathcal{N})$, we can choose a term t_j and calculate an intersection point (or ray) in this way. The end result is a collection of n points and rays that are affinely independent as a result of the n rays of $C(\mathcal{N})$ being affinely independent. Consequently, there is a unique inequality going through these intersection points and rays that cuts off $(\bar{x}, \bar{s})$: $\sum_{j \in \mathcal{N}} (d_j^{t_j} / d_0^{t_j}) s_j \geq 1$. In general, this inequality will not be valid for P_D. For validity, for each $j \in \mathcal{N}$, the inequality must not cut away any intersection point along the positive direction of r^j, because each of these points belongs to P_D. At the same time, no intersection point along the negative extension of r^j should strictly satisfy the cut.[6]

[6]For informal geometric intuition, consider a cut defined as above and a line going through two intersection points defining the cut, where one of the points comes from using the negative extension of a ray r^j of $C(\mathcal{N})$. The cut is invalid if that line intersects the interior of $\operatorname{conv}(C(\mathcal{N}) \setminus \operatorname{int} S)$ at some point p. This is because all points on the line satisfy the cut at equality, and all points on $p + \lambda r^j$ for $\lambda \in \mathbb{R}_{>0}$ violate the cut and belong to $C(\mathcal{N})$; for a small enough λ, this means a point of $C(\mathcal{N}) \setminus \operatorname{int} S$ would be cut. Such a scenario can never occur when no point along any negative extension strictly satisfies the cut.

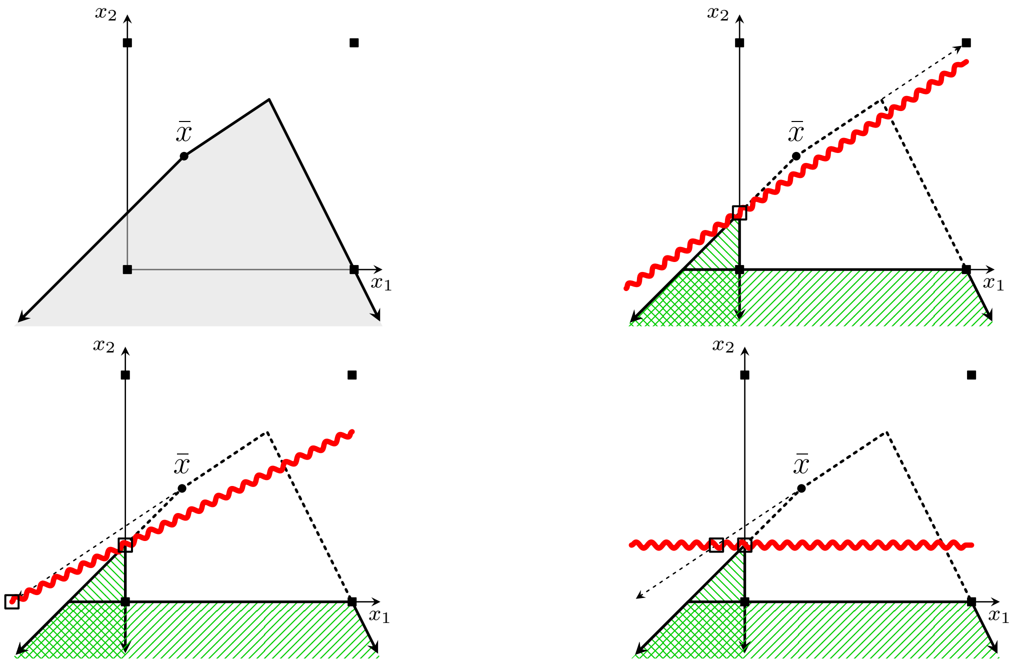


Figure 1.1: This example shows that SICs may have negative cut coefficients in the nonbasic space. The disjunction here is $(x_1 \leq 0) \vee (x_2 \leq 0)$. The top right panel shows the cut if only nonnegative coefficients are allowed. The bottom left panel shows the SIC, obtained by extending one of the rays emanating from $\bar{x}$ until it hits the *last* (not first) disjunctive term. The bottom right panel shows the unique facet-defining inequality for $\operatorname{cl} \operatorname{conv}(C(\mathcal{N}) \setminus \operatorname{int} S)$.

As we already mentioned, the formula for the SIC allows negative coefficients; namely, if $d^t_j < 0$ for all $t \in \mathcal{T}$, then the SIC coefficient on s_j will be negative. In contrast, as discussed by Balas and Kis [29], in the literature, intersection cuts are sometimes assumed to have nonnegative coefficients. This disparity is explained by an additional assumption sometimes made on S, that it is lattice-free, not arbitrarily P_I-free. If S is lattice-free (or it is bounded), then every ray of $C(\mathcal{N})$ either intersects the $\operatorname{bd} S$, or it is “parallel” to one of the facets of S, i.e., $d^t_j = 0$, from which it follows that all the coefficients of the SIC are indeed nonnegative. In general, however, imposing nonnegativity on the cut coefficients substantially restricts the attainable intersection cuts. This is demonstrated in Figure 1.1. The cut in the top right panel is the intersection cut restricted to having nonnegative coefficients, while the cut in the bottom left panel is what we get from applying the formula for the SIC.

Figure 1.1 also demonstrates an additional effect of using a P_I-free convex set. When S is lattice-free, the SIC is the unique facet-defining inequality of $\operatorname{cl} \operatorname{conv}(C(\mathcal{N}) \setminus \operatorname{int} S)$. On the other hand, as shown in the bottom right panel, when S is not lattice-free, the SIC may not define a facet of $\operatorname{cl} \operatorname{conv}(C(\mathcal{N}) \setminus \operatorname{int} S)$. The reason for this is the too-stringent requirement that no intersection point along any negative extension of a ray of $C(\mathcal{N})$ strictly satisfies the cut. That is sufficient, but not necessary, for validity. The intuition in Footnote 6 can be formalized to give a necessary condition for a cut to be valid for $C(\mathcal{N}) \setminus \operatorname{int} S$.

The topic of Chapter 2 is a generalization of SICs called *generalized intersection cuts* (GICs) introduced by Balas and Margot [31]. This is a framework for generating inequalities stronger than SICs by tightening $C(\mathcal{N})$ with additional valid inequalities before intersecting the resulting relaxation with the set S.

Next, we apply the SIC formula to an elementary split disjunction, which is equivalent to the Gomory cut when all nonbasic variables are continuous.

Standard intersection cut from an elementary split. The easiest SIC comes from using the elementary split disjunction $(x_k \le \lfloor \bar{x}_k \rfloor) \vee (x_k \ge \lceil \bar{x}_k \rceil)$ for any $k \in \mathcal{I}$ with $\bar{x}_k \notin \mathbb{Z}$. In the nonbasic space, by substituting the tableau row in place of x_k, this disjunction is

$$\left(\bar{x}_k + \sum_{j \in \mathcal{N}} r^j_k s_j \le \lfloor \bar{x}_k \rfloor \right) \bigvee \left(\bar{x}_k + \sum_{j \in \mathcal{N}} r^j_k s_j \ge \lceil \bar{x}_k \rceil \right).$$

Let $f_0 := \bar{x}_k - \lfloor \bar{x}_k \rfloor$ be the fractional component of $\bar{x}_k$. With this substitution, and ensuring the first disjunctive term inequality is in the right direction, the disjunction becomes

$$\left(-\sum_{j \in \mathcal{N}} r^j_k s_j \ge f_0 \right) \bigvee \left(\sum_{j \in \mathcal{N}} r^j_k s_j \ge 1 - f_0 \right).$$

The resulting SIC is

$$\sum_{j \in \mathcal{N}} \max \left\{ -\frac{r^j_k}{f_0}, \frac{r^j_k}{1 - f_0} \right\} s_j \ge 1.$$

This is a Gomory cut when all nonbasic variables are continuous, though expressed in terms of the rays r^j rather than their negations, where $-r^j$ corresponds to the entries of the optimal tableau (as used in some derivations of the Gomory cut). An equivalent expression for it is

$$\sum_{\substack{j \in \mathcal{N}: \\ r^j_k < 0}} \frac{r^j_k}{f_0} s_j + \sum_{\substack{j \in \mathcal{N}: \\ r^j_k > 0}} \frac{r^j_k}{1 - f_0} s_j \ge 1.$$

We now return to the geometric interpretation. When $r^j_k \ne 0$, there exists a strictly positive value $\lambda_j \in \mathbb{R}$ such that $\bar{x} + \lambda_j r^j \in \operatorname{bd} S$, which is the same as saying that $\bar{x}_k + \lambda_j r^j_k$ equals $\lfloor \bar{x}_k \rfloor$ or $\lceil \bar{x}_k \rceil$, or that λ_j is equal to $\max\{-f_0/r^j_k, (1-f_0)/r^j_k\}$, since exactly one of those terms is positive valued. This is the distance taken along r^j until it intersects $\operatorname{bd} S$, which is why the resulting cut, $\sum_{j \in \mathcal{N}} (1/\lambda_j) s_j \ge 1$, is called an intersection cut, where $\lambda_j := 0$ for all $j \in \mathcal{N}$ such that $r^j_k = 0$.

Standard intersection cut example. Consider the optimization problem below, where the coefficient a_2 is a strictly positive rational number that will be used as a parameter.

$$\begin{aligned} \max_{x_1, x_2, x_3} \quad & x_3 \\ & x_1 + x_3 \le 1 \\ & a_2 x_2 + x_3 \le 1/2 \\ & x_1, x_2, x_3 \ge 0 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

An optimal solution to the LP relaxation is $\bar{x} = (1/2, 0, 1/2)$. The optimal tableau, with respect to the cobasis comprising nonnegative slack variables s_2 (the slack on $x_2 \geq 0$, so $s_2 = x_2$), s_4 (on the first constraint), and s_5 (on the second constraint), is below.

$$\begin{aligned} x_1 &= 1/2 + a_2 s_2 - s_4 + s_5 \\ x_3 &= 1/2 - a_2 s_2 - s_5 \end{aligned}$$

First, we generate the SIC from $(x_1 \leq 0) \vee (x_1 \geq 1)$. There are three rays, corresponding to the nonbasic variables: r^2, r^4, and r^6. Let p^j denote the intersection point of ray r^j with $\operatorname{bd} S$, where $S = \{x : 0 \leq x_1 \leq 1\}$, and let λ_j be the distance along r^j until p^j. Thus, to calculate p^2, we find the positive value λ_2 such that

$$x_1 = 1/2 + a_2 \lambda_2 = 0 \text{ or } 1,$$

i.e., $\lambda_2 = 1/(2a_2)$, yielding the intersection point $p^2 = (1, 1/(2a_2), 0)$. Similarly, we can calculate that $\lambda_4 = \lambda_5 = 1/2$ with intersection points $p^4 = (0, 0, 1/2)$ and $p^5 = (1, 0, 0)$. The SIC, $\sum_{j \in \mathcal{N}} (1/\lambda_j) s_j \geq 1$, is

$$2a_2 s_2 + 2s_4 + 2s_5 \geq 1.$$

Substituting the slacks out using $s_2 = x_2$, $s_4 = 1 - x_1 - x_3$, and $s_5 = 1/2 - a_2 x_2 - x_3$, and simplifying a little, the SIC (in the structural space) is

$$x_1 + 2x_3 \leq 1.$$

If we look at the formulation of the problem in the nonbasic space, the disjunction is

$$\left(-a_2 s_2 + s_4 - s_5 \geq \frac{1}{2}\right) \vee \left(a_2 s_2 - s_4 + s_5 \geq \frac{1}{2}\right)$$

and the intersection points actually lie on the axes. This is depicted in Figure 1.2. This demonstrates that, in some sense, it may be simpler to derive a cut in the nonbasic space, as that is somehow a "natural" space for the cut due to the sparsity of the intersection points in that space, and only at the end substitute out the slacks to return to the space of the structural variables (via Lemma 1.1).

1.3.2 Gomory mixed-integer cuts (strengthened intersection cuts)

We incorporate nonbasic integrality information to strengthen the disjunction and generate the full, stronger, GMIC. As before, let k be the index of an integer variable with $\bar{x}_k \notin \mathbb{Z}$. Now suppose some of the nonbasic variables are also integer-restricted. That implies that one can choose values (to be optimized later) $z_k^j \in \mathbb{Z}$, $j \in \mathcal{N} \cap \mathcal{I}$, such that the tableau row corresponding to x_k can be written

$$\underbrace{x_k - \sum_{j \in \mathcal{N} \cap \mathcal{I}} z_k^j s_j}_{\text{integer-valued}} = \bar{x}_k + \sum_{j \in \mathcal{N} \cap \mathcal{I}} (r_k^j - z_k^j) s_j + \sum_{j \in \mathcal{N} \setminus \mathcal{I}} r_k^j s_j.$$

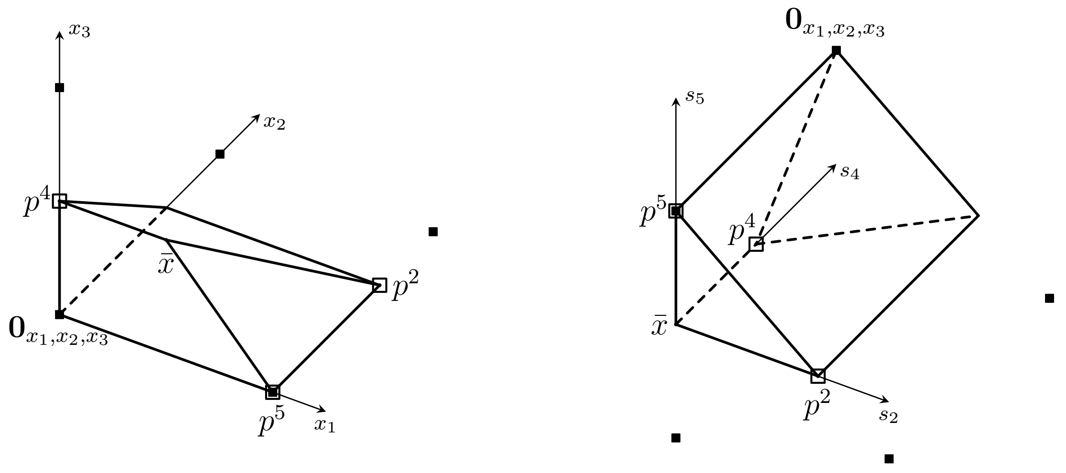


Figure 1.2: Moving from the structural to the nonbasic space, the cone $C(\mathcal{N})$ becomes the nonnegative orthant. Lattice points are shown as filled black squares, intersection points are shown as unfilled squares. Point p^5, for example, is both an intersection and a lattice point.

Since $\bar{s}_j = 0$ for all $j \in \mathcal{N}$, the disjunction $(\bar{x}_k \leq \lfloor \bar{x}_k \rfloor) \vee (\bar{x}_k \geq \lceil \bar{x}_k \rceil)$ can be replaced by the stronger valid disjunction

$$\left(x_k - \sum_{j \in \mathcal{N} \cap \mathcal{I}} z_k^j s_j \leq \lfloor \bar{x}_k \rfloor \right) \vee \left(x_k - \sum_{j \in \mathcal{N} \cap \mathcal{I}} z_k^j s_j \geq \lceil \bar{x}_k \rceil \right).$$

Letting $f_k^j := r_k^j - z_k^j$ for $j \in \mathcal{N} \cap \mathcal{I}$, we rewrite the disjunction in the nonbasic space as

$$\left(- \sum_{j \in \mathcal{N} \cap \mathcal{I}} f_k^j s_j - \sum_{j \in \mathcal{N} \setminus \mathcal{I}} r_k^j s_j \geq f_0 \right) \vee \left(\sum_{j \in \mathcal{N} \cap \mathcal{I}} f_k^j s_j + \sum_{j \in \mathcal{N} \setminus \mathcal{I}} r_k^j s_j \geq 1 - f_0 \right).$$

The intersection cut from this disjunction, by applying the formula, is

$$\sum_{j \in \mathcal{N} \cap \mathcal{I}} \max \left\{ \frac{-f_k^j}{f_0}, \frac{f_k^j}{1 - f_0} \right\} s_j + \sum_{j \in \mathcal{N} \setminus \mathcal{I}} \max \left\{ \frac{-r_k^j}{f_0}, \frac{r_k^j}{1 - f_0} \right\} s_j \geq 1.$$

The nonnegativity of the variables implies that stronger cuts have smaller coefficients, so the optimal choice of z_k^j is either $\lfloor r_k^j \rfloor$ or $\lceil r_k^j \rceil$. The coefficient on s_j, $j \in \mathcal{N} \cap \mathcal{I}$, is then

$$\min \left\{ \frac{r_k^j - \lfloor r_k^j \rfloor}{1 - f_0}, \frac{\lceil r_k^j \rceil - r_k^j}{f_0} \right\}.$$

Hence, $z_k^j = \lfloor r_k^j \rfloor$ when

$$\begin{aligned} \frac{\lceil r_k^j \rceil - r_k^j}{f_0} \geq \frac{r_k^j - \lfloor r_k^j \rfloor}{1 - f_0} &\iff (1 - f_0) \cdot \lceil r_k^j \rceil - (1 - f_0) \cdot r_k^j \geq f_0 \cdot r_k^j - f_0 \cdot \left(\lceil r_k^j \rceil - 1 \right) \\ &\iff r_k^j + f_0 \leq \lceil r_k^j \rceil. \end{aligned}$$

The closed-form expression for the resulting cut (in the nonbasic space) is

$$\sum_{\substack{j\in\mathcal{N}\cap\mathcal{I}:\\ r^j_k+f_0\le\lceil r^j_k\rceil}} \frac{r^j_k-\lfloor r^j_k\rfloor}{1-f_0}s_j+\sum_{\substack{j\in\mathcal{N}\cap\mathcal{I}:\\ r^j_k+f_0>\lceil r^j_k\rceil}} \frac{\lceil r^j_k\rceil-r^j_k}{f_0}s_j+\sum_{\substack{j\in\mathcal{N}\setminus\mathcal{I}:\\ r^j_k<0}}\frac{r^j_k}{f_0}s_j+\sum_{\substack{j\in\mathcal{N}\setminus\mathcal{I}:\\ r^j_k>0}}\frac{r^j_k}{1-f_0}s_j\ge 1,$$

which is exactly the Gomory mixed-integer cut (GMIC). As before, the one deviation from many traditional derivations is that we calculate using r^j and not $-r^j$ (which are the values in the optimal tableau, from assuming no variables that are nonbasic at their upper bounds).

Gomory mixed-integer cut example. Continuing the previous example, we observe that s_2 is an integer-restricted nonbasic variable. Thus, the SIC can be strengthened along ray r^2. Previously, we calculated the coefficient on s_2 by (the inverse of) the distance along r^2 to the intersection point on $\operatorname{bd} S$, which corresponded to $\lambda_2 = 1/(2a_2)$. Now, for some values of a_2 and using the integrality of s_2, we can actually go farther along r^2 while maintaining validity of the resulting cut. The value of f_0 is $1/2$. In the nonbasic space, the coefficient on s_2 was previously $2a_2$, when the integrality of s_2 was ignored; for the strengthened cut, the coefficient on s_2 in the nonbasic space will be

$$2\cdot\min\{r^j_k-\lfloor r^j_k\rfloor, \lceil r^j_k\rceil-r^j_k\} = 2\cdot\min\{\lceil a_2\rceil-a_2, a_2-\lfloor a_2\rfloor\}.$$

We can convert the mixed-integer cut, in this parametric form, to the structural space, resulting in the cut (after simplification)

$$x_1+\max\{\lfloor a_2\rfloor, 2a_2-\lceil a_2\rceil\}x_2+2x_3\le 1.$$

If $a_2\in(0,1/2]$, the "strengthened" coefficient is the same as the "unstrengthened" one. However, if $a_2\in[1/2,1]$, then the cut becomes

$$x_1+(2a_2-1)x_2+2x_3\le 1,$$

which is stronger than $x_1+2x_3\le 1$. This example is illustrated in Figure 1.3 for $a_2=3/4$.

As a_2 increases, the linear relaxation feasible region gets smaller and the GMIC gets monotonically stronger, though only weakly, as it remains $x_1+kx_2+2x_3\le 1$ for all $a_2\in[k,k+1/2]$ for any nonnegative integer k. Moreover, the GMIC slowly approaches, but is always strictly dominated by, the cut $x_2\le 0$ which is valid for P_I when $a_2>1/2$ and valid for the four-term disjunction

$$\begin{pmatrix}x_1\le 0\\ x_2\le 0\end{pmatrix}\vee\begin{pmatrix}x_1\ge 1\\ x_2\le 0\end{pmatrix}\vee\begin{pmatrix}x_1\ge 0\\ x_2\ge 1\end{pmatrix}\vee\begin{pmatrix}x_1\ge 1\\ x_2\ge 1\end{pmatrix}.$$

This demonstrates that split cuts can be strictly worse than cross cuts, and that finding cuts from the above four-term disjunction based on the integrality of x_1 and x_2 is not equivalent to taking a split disjunction on x_1 and afterwards incorporating the integrality of x_2.

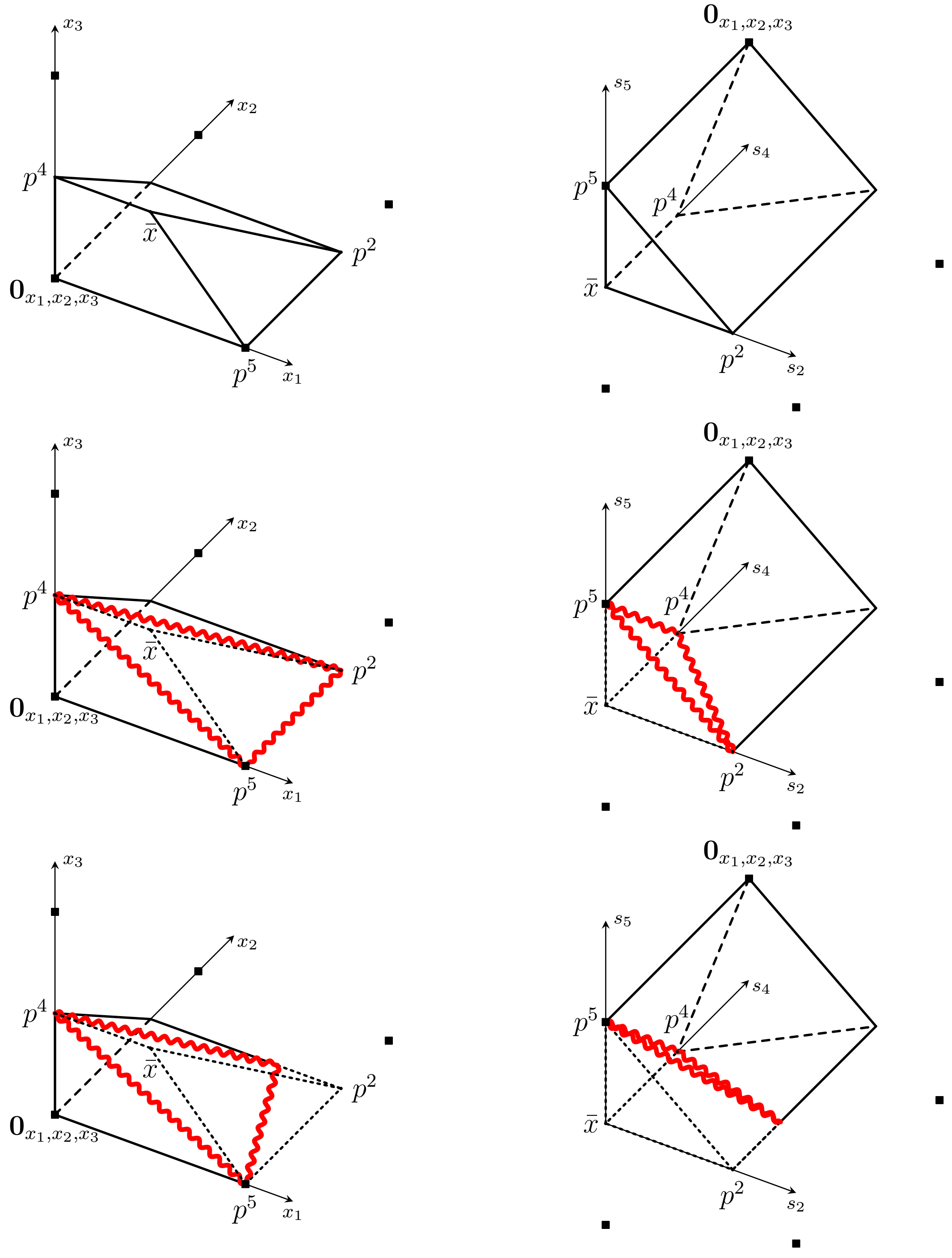


Figure 1.3: Feasible region of the linear relaxation as in the original formulation (top), after adding a SIC (middle), and after adding a GMIC (bottom), shown in the structural space (left) and the nonbasic space (right). Lattice points are marked by black squares.

1.4 Disjunctive cuts, irregularity, and lift-and-project

We first briefly review the recent literature on generating disjunctive cuts. We highlight recent findings that show that there exist disjunctive cuts that are strictly stronger than all possible SICs obtainable from that disjunction. We then discuss the lift-and-project approach for generating facet-defining disjunctive cuts and properties of the linear program used by the lift-and-project method for computing disjunctive cuts. Later, in Chapter 4, we identify some of the weaknesses of the lift-and-project machinery and build an alternative method that offers some computational advantages.

1.4.1 Background on disjunctive cuts

We have already seen in Section 1.3 how SICs arise as disjunctive cuts. Of course, the SIC is one cut, while there are, in general, many valid inequalities for the disjunctive hull. Lift-and-project is a method that can produce all valid disjunctive cuts, based on the theory developed by Balas [17]. It involves building an extended formulation of the disjunctive hull, from which cuts are separated. This extended formulation comes from finding *Farkas multipliers* that certify the validity of the cut. Concretely, for any inequality $\alpha^\mathsf{T} x \ge \beta$ valid for P_D, the inequality must be valid for each P^t, $t \in \mathcal{T}$. Farkas' lemma [81] implies that, for each $t \in \mathcal{T}$, there exist row vectors of nonnegative multipliers $(u^t, u_0^t) \in \mathbb{R}^m_{\ge 0} \times \mathbb{R}^{m_t}_{\ge 0}$ (where m_t denotes the number of rows of D^t) such that

$$\begin{aligned} \alpha^\mathsf{T} &= u^t A + u_0^t D^t \\ \beta &\le u^t b + u_0^t b. \end{aligned}$$

Though successful in improving contemporary codes in its first implementations [24, 25], the lift-and-project method later proved prohibitively expensive to work with due to the extended formulation. However, it turns out that for the special case of split disjunctions, every valid cut is equivalent to a SIC from some (possibly infeasible) basis of P [7, 32]. This relationship can be used to forego the extended formulation and merely pivot in the original space, leading to markedly more efficient algorithms [22, 32].

Unfortunately, the correspondence between generating cuts in the extended space and SICs in the original space breaks down for more general disjunctions. Andersen et al. [7] give an example with a two-term simple disjunction for which there is a cut that is strictly stronger than all of the SICs combined (including those from infeasible bases). A three-dimensional example of such a cut is given in the dissertation of Jörg [102, Chapter 3]. Disjunctive cuts for which there is no corresponding SIC were named *irregular cuts* and explored in more depth by Balas and Kis [30].[7] For non-simple disjunctions, a cut is irregular if there is no corresponding SIC using *any* convex set S such that $\operatorname{int} S \subseteq \mathbb{R}^n \setminus \cup_{t\in\mathcal{T}}(D^t x \ge D_0^t)$.

1.4.2 Irregular cuts

The primary feature of a SIC is that there exists a cobasis of $Ax \ge b$ from which it is generated. In terms of the Farkas multipliers $\{u^t\}_{t\in\mathcal{T}}$, this means that the set of rows of A

[7]We later refine this definition to a cut being irregular if it cannot be derived as a simple disjunctive cut, where the disjunction does not include the constraints of P.

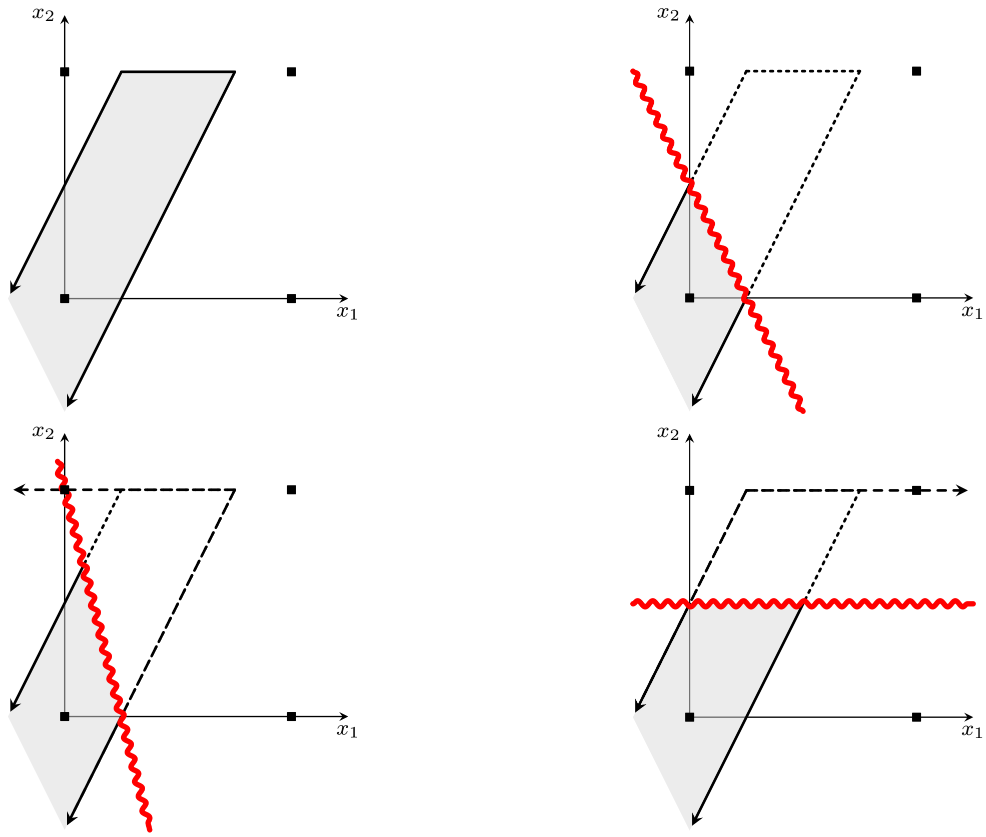


Figure 1.4: Example of an irregular cut from the disjunction $(x_1 \le 0) \vee (x_2 \le 0)$. The cut is stronger than the two possible SICs and removes portions of both corner polyhedra; the corner polyhedra (with dashed lines) and associated SICs are shown in the bottom panel.

getting positive weight (for any disjunctive term) is linearly independent and the associated submatrix has rank n. Let $K := \{i \in [m] : \text{there exists } t \in \mathcal{T} \text{ for which } u_i^t > 0\}$, and A_K denote the set of rows of A indexed by K. Thus, an irregular cut occurs if, for *every* feasible set of Farkas multipliers and the associated set K, either $|K| < n$ and $\operatorname{rank}(A_K) < |K|$, or $|K| > n$. An irregular cut of the former type is shown in the top right panel of Figure 1.4, while an irregular cut of the latter type is illustrated in the top right panel of Figure 1.5.

We emphasize that it is insufficient to demonstrate one set of "irregular" Farkas multipliers; one has to show that there is no way to certify the cut with exactly n linearly independent rows of A.[8] This is generally much more difficult for a non-simple disjunction, for which the cut must be irregular for every possible S that can be derived from the disjunction. Figure 1.6 illustrates such an irregular cut. The first panel depicts the feasible region of the polyhedron; the second panel shows the result of taking the non-simple disjunction $(x_1 \le 0, x_2 \le 0) \vee (x_1 \ge 1, x_2 \le 0) \vee (x_1 \le 0, x_2 \ge 1)$.

Now suppose the disjunction is simple, i.e., that the corresponding P_I-free set S is actually lattice-free. One remarkable feature of irregular cuts shown in [30] relates to *corner polyhedra* [89]. Given a (possibly infeasible) basic solution $\widetilde{x} \in \operatorname{int} S$, let $\mathcal{N}(\widetilde{x})$ denote the set

[8]With the additional restriction that the point $A_K^{-1} b_K$ does not belong to P_D.

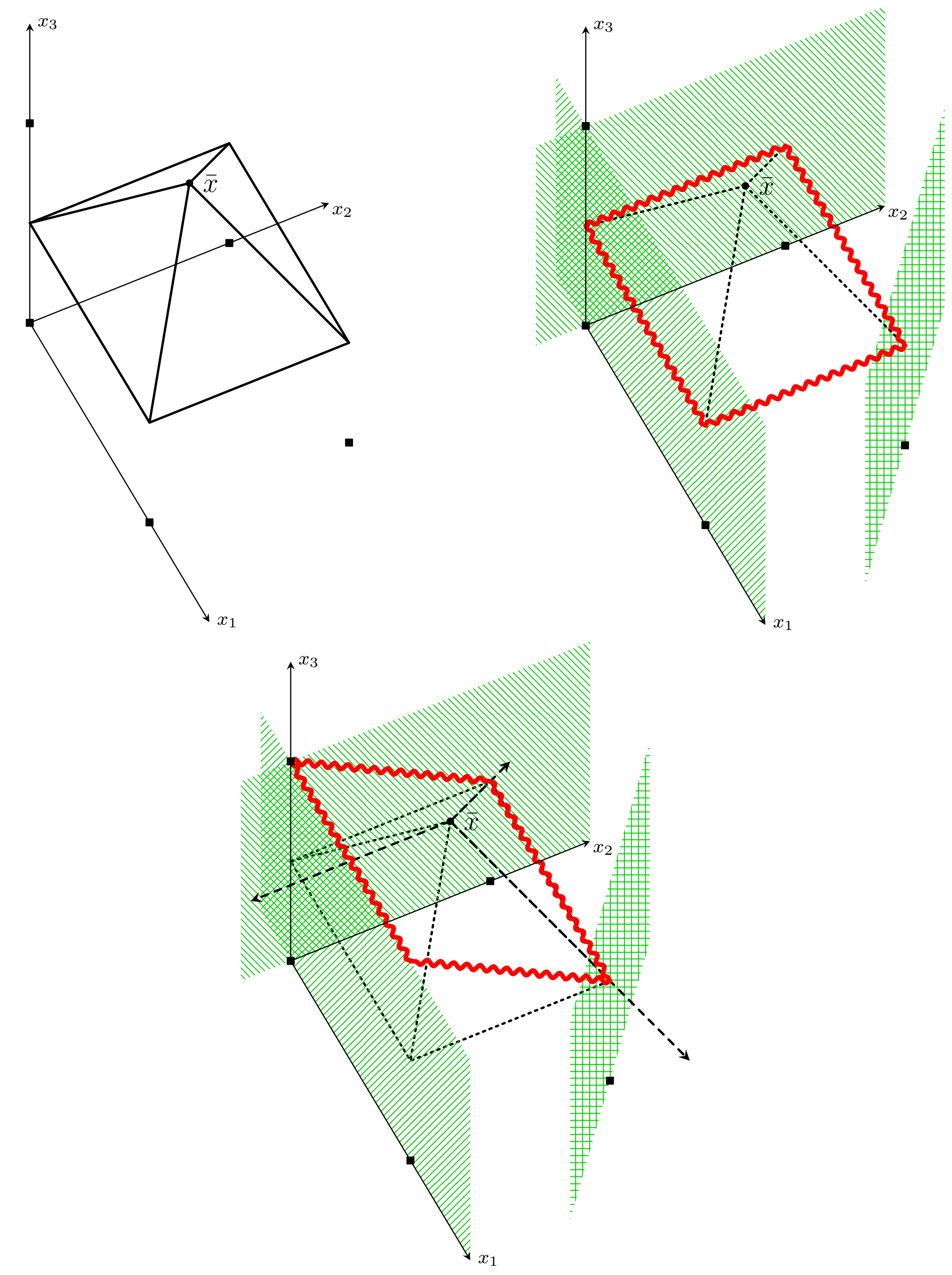


Figure 1.5: Example of an irregular cut requiring Farkas multipliers on more than n inequalities of P. The disjunction is $(x_1 \leq 0) \vee (x_2 \leq 0) \vee (x_1 + x_2 \geq 2)$, shown by the patterned areas. The bottom panel shows a SIC from one basis (the other three are similar).

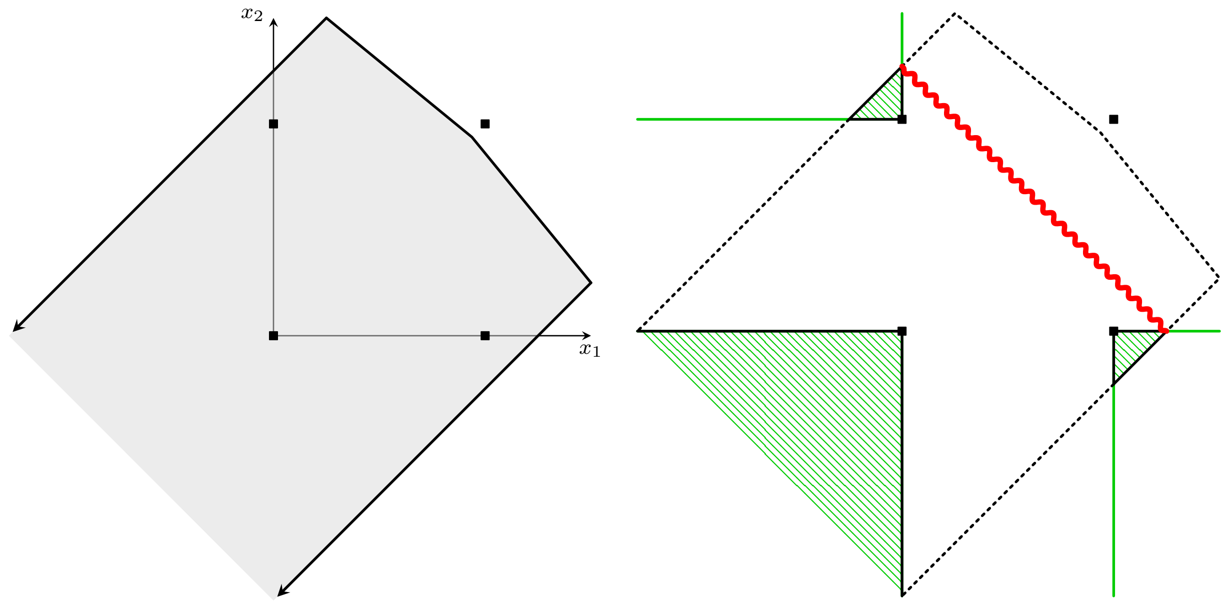


Figure 1.6: Example of an irregular cut from a non-simple disjunction.

of nonbasic variables at $\widetilde{x}$ and $C(\mathcal{N}(\widetilde{x}))$ be the associated basis cone. The corner polyhedron is defined as

$$\{x \in C(\mathcal{N}(\widetilde{x})) : x_j \in \mathbb{Z} \text{ for all } j \in \mathcal{I}\}.$$

The facet-defining inequalities of any corner polyhedron are intersection cuts [57], and every intersection cut from $\mathcal{N}(\widetilde{x})$ (for any lattice-free convex set with $\widetilde{x}$ in its interior) is valid for the corner polyhedron [21]. Irregular cuts, on the other hand, not only can be stronger than all possible SICs combined, but also may cut away portions of the corner polyhedron. As stated in [30, Theorem 19], a sufficient condition for a cut $\alpha^\mathsf{T} x \geq \beta$ to be invalid for a corner polyhedron associated with a cobasis of $\widetilde{x}$ is that some ray r of $C(\mathcal{N}(\widetilde{x}))$ satisfies $\alpha^\mathsf{T} r < 0$. This is, for example, satisfied by two of the vertices violating the cut in Figure 1.6.[9] (Conversely, if a cut $\alpha^\mathsf{T} x \geq \beta$ is irregular, it is not difficult to show that there *exists* a cobasis and ray r of the associated basis cone satisfying $\alpha^\mathsf{T} r \leq 0$.)

Moreover, an irregular cut might be invalid for the corner polyhedron from *every* possible basis for P. The earlier example in Figure 1.4 already shows such an irregular cut, $2x_1 + x_2 \leq 1/2$, produced from the disjunction $(x_1 \leq 0) \vee (x_2 \leq 0)$ (this is a disjunction for which the set S is not lattice-free; it would only weaken the attainable SICs if we added an additional disjunctive term such as $(x_1 + x_2 \geq 2)$ to make the resulting set S lattice-free). The example also demonstrates that irregularity is, of course, a property depending on the disjunction; the cut $x_1 \leq 0$ dominates the irregular cut, and it is valid for the split disjunction $(x_1 \leq 0) \vee (x_1 \geq 1)$, but not for the disjunction used in the example. In fact, for any given cut that is known to be valid for P_I, there (trivially) exists a disjunction for which the cut is a SIC, as we show in Proposition 1.3 (see also

[9]Irregularity of course does not imply that such a violated corner polyhedron exists. For example, modify Figure 1.6 by removing the point near $(1, 1)$ from the $\mathcal{V}$-polyhedral description of the linear relaxation.

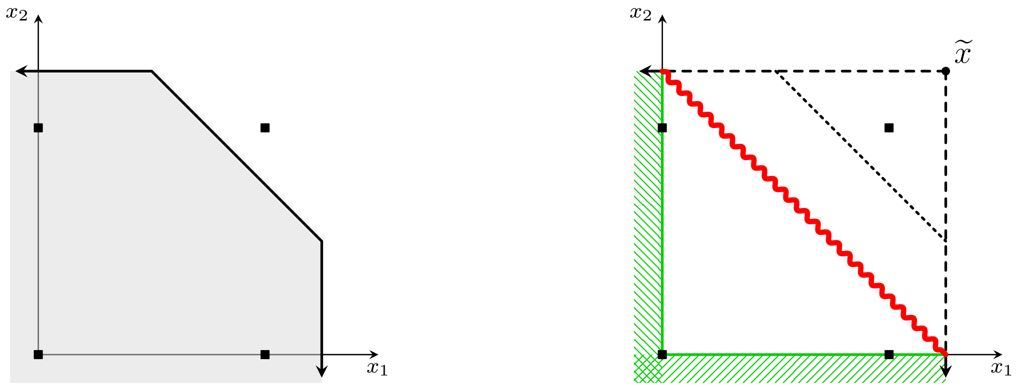


Figure 1.7: Example of a SIC that cuts off an element of the corner polyhedron, when the convex set defined by the disjunction is not lattice-free. The disjunction is $(x_1 \leq 0) \vee (x_2 \leq 0)$.

Proposition 1.3. *Suppose $\alpha^\mathsf{T} x \geq \beta$ is a valid cut for P_I that is not valid for P. Then there exists a P_I-free set from which this cut can be derived as a SIC.*

Proof. The validity of the cut implies that $S := \{x \in \mathbb{R}^n : \alpha^\mathsf{T} x \leq \beta\}$ is a P_I-free convex set. Let $v \in \arg\min\{\alpha^\mathsf{T} x : x \in P\}$. Since $\alpha^\mathsf{T} x \geq \beta$ is not valid for P, $\alpha^\mathsf{T} v < \beta$. Moreover, for any optimal cobasis $\mathcal{N}(v)$ of v, by optimality, every ray r of the cone $C(\mathcal{N}(v))$ satisfies $\alpha^\mathsf{T} r \geq 0$. It follows that each of these rays emanating from v either intersects $\alpha^\mathsf{T} x = \beta$, or $\alpha^\mathsf{T} r = 0$. These n intersection points and rays uniquely define the cut $\alpha^\mathsf{T} x \geq \beta$. □

A slightly subtle detail of the above discussion is the pair of assumptions that $\widetilde{x} \in \operatorname{int} S$ and S is lattice-free. The former assumption is necessary; it does not make sense to apply the formula for a SIC when $\widetilde{x}$ satisfies one of the disjunctive term inequalities. However, if $P^t = \emptyset$ for that term, then we can remove or ignore that term completely, yielding a modified disjunction corresponding now to a P_I-free convex set. This would violate the second assumption, which turns out to be crucial for guaranteeing that intersection cuts are valid for the corner polyhedron associated with $\widetilde{x}$. Figure 1.1 is an example of this: the point $(1,1)$ belonging to the corner polyhedron at $\bar{x}$ in that figure violates the intersection cut (even if, as in the top right panel, nonnegativity on the cut coefficients in the nonbasic space is imposed).

Another example, showing the effect of the assumption that $\widetilde{x} \in \operatorname{int} S$, is given in Figure 1.7. The polyhedron P_I is $\{x \in \mathbb{Z}^2 : x_1 + x_2 \leq 7/4,\ x_1 \leq 5/4,\ x_2 \leq 5/4\}$. If we use the set S defined as the triangle $\{x \in \mathbb{R}^3 : x_1 \geq 0,\ x_2 \geq 0,\ x_1 + x_2 \leq 2\}$, the one facet-defining inequality for the disjunctive hull is $x_1 + x_2 \leq 5/4$ (shown as the wavy line in the right panel of the figure). The SICs from the two bases corresponding to $(1/2, 5/4)$ and $(5/4, 1/2)$ are strictly weaker. Now consider the point $\widetilde{x} = (5/4, 5/4)$ (the only remaining basis for P). This lies outside of S, so we cannot use it to generate a SIC. Yet, the cut is not quite irregular. The disjunctive term defined by $x_1 + x_2 \geq 2$ contains no points feasible to P, and hence it can be ignored; i.e., we can instead use the P_I-free convex set $\{x \in \mathbb{R}^2 : x_1 \geq 0,\ x_2 \geq 0\}$. The facet-defining inequality can now be generated as a SIC from the basis defining $\widetilde{x}$. Observe that this inequality cuts away $(1,1)$, which belongs to the corner polyhedron at $\widetilde{x}$.

The preceding discussion leads us to an illuminating example underlining the effect of the restriction that $\widetilde{x}$ belongs to $\operatorname{int} S$. Namely, if we define irregularity with respect to SICs,

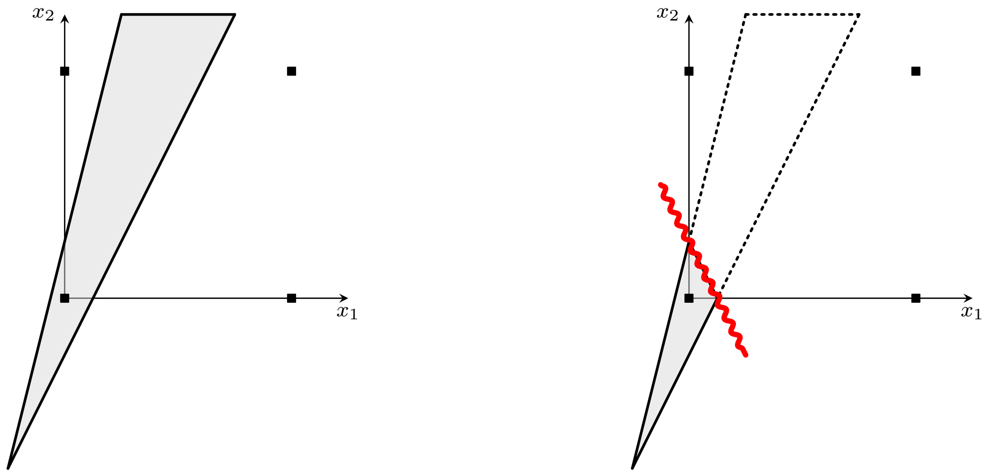


Figure 1.8: The cut in the right panel, derived from the disjunction $(x_1 \leq 0) \vee (x_2 \leq 0)$, cannot be generated as a SIC but does not satisfy the definition of irregularity that the number of positive Farkas multipliers is either fewer or more than n. It can, however, be generated as a simple disjunctive cut.

then the characterization based on K (the positive Farkas multipliers) is incomplete; there exists a third type of "irregular" cut, i.e., one that cannot be generated as a SIC but is also not covered by the definition of irregularity given earlier and in [30] with respect to K. The only remaining case is that A_K has rank n, but K corresponds to a cobasis from which we cannot derive an intersection cut. Figure 1.8 illustrates a concrete example of such an uncharacterized irregular cut. It is not difficult to verify that cut shown in the right panel of the figure is certified by Farkas multipliers leading to a matrix A_K with rank exactly $n = 2$. However, the corresponding vertex of P is a point feasible to the disjunction, implying it cannot be used to generate a SIC. Nevertheless, it can be shown that the cut is obtainable as a simple disjunctive cut, via Lemma 1.2. Using a modification of Theorem 9 in [30], one can show a general correspondence between *regular* bases and simple disjunctive cuts, as we discuss in Appendix A.

The existence of irregular cuts from a given particular disjunction, and their apparent strength, has been a strong motivation for this dissertation topic of pursuing efficient ways to produce inequalities from general valid disjunctions.

1.4.3 Cut-generating linear program for lift-and-project cuts

For convenience, we simplify notation (it will be unpacked when necessary). Let $A^t := \begin{bmatrix} A \\ D^t \end{bmatrix}$, $b^t := \begin{bmatrix} b \\ D_0^t \end{bmatrix}$, and $v^t := (u^t, u_0^t)$. With this notation, $P^t = \{x : A^t x \geq b^t\}$.

The lift-and-project method involves generating valid disjunctive cuts, and the associated Farkas multipliers v^t, as solutions of the following *cut-generating linear program* (CGLP), in which we have added a normalization constraint, determined by parameters $n^\alpha \in \mathbb{R}^n$,

$n_\beta \in \mathbb{R}$, $n^t \in \mathbb{R}^{m+m_t}$, and $n_0 \in \mathbb{R}$.

$$\begin{aligned}
\min_{\alpha,\beta,\{v^t\}_{t\in\mathcal{T}}} \quad & \alpha^\mathsf{T}\bar{x} - \beta & \\
& -\alpha^\mathsf{T} + v^t A^t = 0 & \text{for all } t \in \mathcal{T} \\
& \beta - v^t b^t \leq 0 & \text{for all } t \in \mathcal{T} \\
& \alpha^\mathsf{T} n^\alpha + \beta n_\beta + \sum_{t\in\mathcal{T}} v^t n^t = n_0 & \\
& v^t \geq 0 & \text{for all } t \in \mathcal{T}
\end{aligned} \tag{CGLP}$$

This linear program is polynomially-sized: it has $|\mathcal{T}| \cdot (n+1) + 1$ nontrivial constraints and $n+1+|\mathcal{T}|\cdot(m+m_t)$ variables. The choice of normalization can be very significant when generating L&PCs. For example, Fischetti et al. [84] showed that the normalization may prevent certain facet-defining inequalities of P_D from being generated, and that optimal solutions to (CGLP) may not even be supporting for the disjunctive hull (an undesirable property particularly given the expensive nature of solving a higher-dimensional linear program). We look at three common normalizations in Appendix B.

In Proposition 1.3, we showed that every valid disjunctive cut can be generated as a SIC from a P_I-free convex set, defined trivially by reversing the direction of the cut. We can also recover this result using the multipliers v^t. For a given solution to (CGLP), decomposing v^t back into (u^t, u_0^t), $t \in \mathcal{T}$, define (with dependence on $\{u_0^t\}_{t\in\mathcal{T}}$ implicit)

$$S := \{x \in \mathbb{R}^n : u_0^t D^t x \leq u_0^t D_0^t \text{ for all } t \in \mathcal{T}\},$$

which is a P_I-free convex set such that $\alpha^\mathsf{T} x \geq \beta$ is valid for $\operatorname{conv}(P \setminus \operatorname{int} S)$. When the disjunction (1.1) partitions $\mathbb{Z}^n$ (i.e., every integer point, feasible or infeasible, satisfies some term of the disjunction, not including the constraints of P), the resulting set S is not only P_I-free but also lattice-free. However, the cut might not be a SIC from S; to get that result, we first observe that the following disjunction is also valid:

$$\bigvee_{t\in\mathcal{T}} (A^t x \geq b^t),$$

which is clear as it simply adds inequalities to the already valid disjunction (1.1).

Proposition 1.4. *Let $(\alpha, \beta, \{v^t\}_{t\in\mathcal{T}})$ be a basic feasible solution to* (CGLP)*. Then $\alpha^\mathsf{T} x \geq \beta$ is a SIC from the disjunction $\vee_{t\in\mathcal{T}}(v^t A^t x \geq v^t b^t)$.*

Proof. Note that $v^t A^t = \alpha^\mathsf{T}$ for all $t \in \mathcal{T}$. Thus, constructing the P_I-free convex set S from the disjunction above using the multipliers v^t,

$$\bigvee_{t\in\mathcal{T}} (v^t A^t x \geq v^t b^t) \iff \bigvee_{t\in\mathcal{T}} (\alpha^\mathsf{T} x \geq v^t b^t).$$

Since the left-hand side is the same for each disjunctive term, we do not need the variables to be nonnegative to apply the formula from Lemma 1.2, which yields that the SIC from this disjunction $\alpha^\mathsf{T} x \geq \min_{t\in\mathcal{T}} v^t b^t$. We assumed the given solution is basic, so it must be the case that $\beta = \min_{t\in\mathcal{T}} v^t b^t$ (otherwise, for sufficiently small $\epsilon > 0$, $(\alpha, \beta, \{v^t\}_{t\in\mathcal{T}}) = (\alpha, \beta + \epsilon, \{v^t\}_{t\in\mathcal{T}}) + (\alpha, \beta - \epsilon, \{v^t\}_{t\in\mathcal{T}})$). This completes the proof. ☐

This result is in contrast to the existence of irregular cuts from arbitrary disjunctions: including the constraints of P in the definition of S reduces the P_I-free convex set to a single halfspace, from which there clearly is only one facet-defining cut (the halfspace reversed).

Next, we look at the dual to (CGLP), which lends useful insights into L&PCs, and in particular for interpreting the effect of normalization. We do this generically in the next section; the application to specific normalization choices is given in Appendix B.

1.4.4 Membership linear program

Below is the dual to (CGLP) after some minor transformations, which we refer to as the *membership linear program* (MLP), as in [47]. Here, γ is the "primal" (dual to the CGLP) variable associated with the normalization constraint.

$$\begin{aligned} \min_{\{y^t, y_0^t\}_{t\in\mathcal{T}},\gamma} \quad & n_0\gamma \\ & A^t y^t - b^t y_0^t + n^t\gamma \geq 0 && \text{for all } t \in \mathcal{T} \\ & y_0^t \geq 0 && \text{for all } t \in \mathcal{T} \\ & \sum_{t\in\mathcal{T}} y^t + n^\alpha \gamma = \bar{x} \\ & \sum_{t\in\mathcal{T}} y_0^t + n_\beta\gamma = 1 \end{aligned} \tag{MLP}$$

For $\gamma = 0$ (i.e., when the CGLP is unnormalized), the first two sets of constraints of (MLP) represent the set of feasible solutions to $\operatorname{cone}(P_D) := \operatorname{conv}(\cup_{t\in\mathcal{T}} \operatorname{cone}(P^t))$, where

$$\operatorname{cone}(P^t) := \{(y^t, y_0^t) : A^t y^t - b^t y_0^t \geq 0,\ y_0^t \geq 0\}.$$

This is the feasible region of P^t made into a homogeneous cone by adding an extra dimension for the right-hand side. The point we are trying to represent in $\operatorname{cone}(P_D)$ is $(\bar{x}, 1)$, which is the role of the last two sets of constraints of (MLP). Of course, as $\bar{x} \notin P_D$, the MLP is not feasible when $\gamma = 0$. This is the reason the normalization is added to system (CGLP). We will need an interpretation of the MLP in the presence of the normalization. Such an interpretation has previously partially been given in the dissertation of Soares [136, Chapter 3].

It is evident from the MLP that the normalization, in the primal space, acts to enlarge the feasible region (when $n^t \neq 0$) and/or to change the point that we wish to represent in $\operatorname{cone}(P_D)$, moving it from $(\bar{x}, 1)$ to $(\bar{x}, 1) - (n^\alpha, n_\beta)\gamma$.

To make this more concrete, consider a fixed solution $(\{\bar{y}^t, \bar{y}_0^t\}_{t\in\mathcal{T}}, \bar{\gamma})$ to the MLP. Define

$$\mathcal{T}^* := \{t \in \mathcal{T} : \bar{y}_0^t > 0\}.$$

Next we create some helper variables.

$$\hat{x}^t := \begin{cases} \bar{y}^t / \bar{y}_0^t & \text{for all } t \in \mathcal{T}^*, \\ \bar{y}^t & \text{for all } t \in \mathcal{T} \setminus \mathcal{T}^*. \end{cases}$$

$$\hat{\gamma}_t := \begin{cases} \bar{\gamma} / \bar{y}_0^t & \text{for all } t \in \mathcal{T}^*, \\ \bar{\gamma} & \text{for all } t \in \mathcal{T} \setminus \mathcal{T}^*. \end{cases}$$

Substituting these in, we can reformulate the MLP to see that $(\{\bar{y}^t, \bar{y}_0^t\}_{t\in\mathcal{T}}, \bar{\gamma})$ satisfies

$$\begin{aligned}
A^t \hat{x}^t &\geq b^t - n^t \hat{\gamma}_t && \text{for all } t \in \mathcal{T}^* \\
A^t \hat{x}^t &\geq -n^t \bar{\gamma} && \text{for all } t \in \mathcal{T} \setminus \mathcal{T}^* \\
\bar{y}_0^t &\geq 0 && \text{for all } t \in \mathcal{T} \\
\sum_{t\in\mathcal{T}^*} \bar{y}_0^t \hat{x}^t &= \bar{x} - n^\alpha \bar{\gamma} \\
\sum_{t\in\mathcal{T}^*} \bar{y}_0^t &= 1 - n_\beta \bar{\gamma}.
\end{aligned}$$

The first two sets of constraint represent the polyhedron P^t, or its recession cone, translated by $-n_i^t \hat{\gamma}_t$. Let $\hat{P}_D = \operatorname{conv}(\cup_{t\in\mathcal{T}} \hat{P}^t)$, where

$$\hat{P}^t := \begin{cases} \{x : A^t x \geq b^t - n^t \hat{\gamma}_t\} & \text{for all } t \in \mathcal{T}^*, \\ \{x : A^t x \geq -n^t \hat{\gamma}_t\} & \text{for all } t \in \mathcal{T} \setminus \mathcal{T}^*. \end{cases}$$

Thus, the solution to the MLP satisfies

$$(\bar{x} - n^\alpha \bar{\gamma}, 1 - n_\beta \bar{\gamma}) \in \operatorname{cone}(\hat{P}_D) \iff \left(\frac{\bar{x} - n^\alpha \bar{\gamma}}{1 - n_\beta \bar{\gamma}}, 1\right) \in \operatorname{cone}(\hat{P}_D) \iff \frac{\bar{x} - n^\alpha \bar{\gamma}}{1 - n_\beta \bar{\gamma}} \in \hat{P}_D.$$

The geometric interpretation is that the solution to the MLP corresponds to finding points $\hat{x}^t \in \hat{P}^t$ for $t \in \mathcal{T}^*$ so that we can take a convex combination of them to obtain the point $\bar{x}$ translated by $n^\alpha \bar{\gamma}$ then scaled by $(1 - n_\beta \bar{\gamma})$. By defining

$$\hat{y}_0^t := \bar{y}_0^t / (1 - n_\beta \bar{\gamma}),$$

we can see this via reformulating the MLP feasible region once more:

$$\begin{aligned}
A^t \hat{x}^t &\geq b^t - n^t \hat{\gamma}_t && \text{for all } t \in \mathcal{T}^* \\
A^t \hat{x}^t &\geq -n^t \bar{\gamma} && \text{for all } t \in \mathcal{T} \setminus \mathcal{T}^* \\
\hat{y}_0^t &\geq 0 && \text{for all } t \in \mathcal{T} \\
\sum_{t\in\mathcal{T}^*} \hat{y}_0^t \hat{x}^t &= (\bar{x} - n^\alpha \bar{\gamma})/(1 - n_\beta \bar{\gamma}) \\
\sum_{t\in\mathcal{T}^*} \hat{y}_0^t &= 1
\end{aligned}$$

Of course, we also have the following equivalence:

$$\frac{\bar{x} - n^\alpha \bar{\gamma}}{1 - n_\beta \bar{\gamma}} \in \hat{P}^t \iff \bar{x} \in (1 - n_\beta \bar{\gamma}) \hat{P}^t + n^\alpha \bar{\gamma}.$$

That is, after translating the constraints of P^t to get $\hat{P}^t$, we then scale and translate this new set so that $\bar{x}$ belongs to it. However, we prefer the interpretation that the feasible region of P^t will only be changed by the n^t coefficients, via translating the constraints.

1.5 Cut strengthening via modularization

There are two sources of the strength of a cut. The first is the disjunction from which the cut is generated; using a stronger disjunction will produce stronger cuts, and, in fact, it is known that one can produce all of the facet-defining inequalities for P_I through sufficiently strong disjunctive cuts [18, 99]. However, once a cut has been produced from a given disjunction, there is a second, a posteriori, source of strength that can be utilized, called *modularization*, that comes from the use of integrality information without actually changing the underlying disjunction producing the cut (though validity comes from an argument that the cut is valid for a stronger disjunction).[10] In fact, we have seen an example of this strength when producing a GMIC based on a SIC from a split disjunction. The direct generalization of that approach is called *standard modularization* by Balas and Qualizza [34], which applies to cuts obtained from simultaneously considering several elementary split disjunctions. For performing strengthening for cuts derived from arbitrary disjunctions, there are two paradigms in the literature: one based on the group-theoretic approach [75, 100], and one called *monoidal cut strengthening* [28]. In this section, we briefly discuss standard modularization and monoidal strengthening, in the style of the presentation in [34] and [17, Section 7].

1.5.1 Standard modularization

Standard modularization involves strengthening a disjunction by modularizing the rays of the optimal tableau, decomposing them into an integer and fractional part, as used in Section 1.3 to strengthen a SIC to a GMIC. Standard modularization was applied to cuts generated from multiple-branch elementary split disjunctions by Balas and Qualizza [34].[11] Essentially, as in Section 1.3, for any r^j_k, $j \in \mathcal{N} \cap \mathcal{I}$ and $k \in \mathcal{I}$, we can decompose the ray into a fractional and an integer component, by choosing values f^j_k and $z^j_k \in \mathbb{Z}$ (to be optimized) such that $r^j_k = f^j_k + z^j_k$. The condition $x_k \in \mathbb{Z}$ can now be replaced by $x_k - \sum_{j \in \mathcal{N} \cap \mathcal{I}} z^j_k s_j \in \mathbb{Z}$, or, in the nonbasic space, as $\bar{x}_k + \sum_{j \in \mathcal{N} \cap \mathcal{I}} f^j_k s_j + \sum_{j \in \mathcal{N} \setminus \mathcal{I}} r^j_k s_j$.

Formally, consider a set of K integer variables that take a fractional value at $\bar{x}$. Assume this set is $\{1, \dots, K\} \subseteq \mathcal{I}$ without loss of generality. A valid disjunction is

$$\bigvee_{\mathcal{Q} \subseteq [K]} \left(x_k \le \lfloor \bar{x}_k \rfloor \text{ if } k \in \mathcal{Q},\ x_k \ge \lceil \bar{x}_k \rceil \text{ if } k \notin \mathcal{Q} \right).$$

Expressed in the nonbasic space, with $f^0_k := \bar{x}_k - \lfloor \bar{x}_k \rfloor$, the disjunction is

$$\bigvee_{\mathcal{Q} \subseteq [K]} \begin{pmatrix} -\sum_{j \in \mathcal{N}} r^j_k s_j \ge f^0_k & \text{if } k \in \mathcal{Q} \\ \sum_{j \in \mathcal{N}} r^j_k s_j \ge 1 - f^0_k & \text{if } k \notin \mathcal{Q} \end{pmatrix}.$$

[10]As we discussed earlier in this chapter, the disjunction (1.1) also uses integrality information, but (typically) from (some) *basic* variables, whereas modularization is applied using the integrality of the *nonbasic* variables.

[11]This is a special case of a t-branch split disjunction as studied by Li and Richard [110], who consider a 2^t-term disjunction formed from combining t split disjunctions.

Applying the aforementioned modularization on the rays corresponding to nonbasic integer variables, the following disjunction is also valid:

$$\bigvee_{\mathcal{Q}\subseteq[K]} \left(\begin{array}{lr} -\sum_{j\in\mathcal{N}\cap\mathcal{I}} f_k^j s_j - \sum_{j\in\mathcal{N}\setminus\mathcal{I}} r_k^j s_j \geq f_k^0 & \text{if } k \in \mathcal{Q} \\ \sum_{j\in\mathcal{N}\cap\mathcal{I}} f_k^j s_j + \sum_{j\in\mathcal{N}\setminus\mathcal{I}} r_k^j s_j \geq 1 - f_k^0 & \text{if } k \notin \mathcal{Q} \end{array} \right).$$

For each $\mathcal{Q} \subseteq [K]$, let $u_0^{\mathcal{Q}} \in \mathbb{R}^K_{\geq 0}$. Any set of such multipliers produces a simple disjunction (sometimes referred to as the *parametric octahedron*). For $j \in \mathcal{N} \cap \mathcal{I}$, let

$$d_j^{\mathcal{Q}} := \sum_{k\notin\mathcal{Q}} u_{0k}^{\mathcal{Q}} f_k^j - \sum_{k\in\mathcal{Q}} u_{0k}^{\mathcal{Q}} f_k^j.$$

For $j \in \mathcal{N} \setminus \mathcal{I}$, let

$$d_j^{\mathcal{Q}} := \sum_{k\notin\mathcal{Q}} u_{0k}^{\mathcal{Q}} r_k^j - \sum_{k\in\mathcal{Q}} u_{0k}^{\mathcal{Q}} r_k^j.$$

Finally, let

$$d_0^{\mathcal{Q}} := \sum_{k\notin\mathcal{Q}} u_{0k}^{\mathcal{Q}} (1 - f_k^0) + \sum_{k\in\mathcal{Q}} u_{0k}^{\mathcal{Q}} f_k^0.$$

The valid simple disjunction from these multipliers is

$$\bigvee_{\mathcal{Q}\subseteq[K]} \left(\sum_{j\in\mathcal{N}} d_j^{\mathcal{Q}} s_j \geq d_0^{\mathcal{Q}} \right).$$

The resulting simple disjunctive cut is

$$\sum_{j\in\mathcal{N}} \max_{\mathcal{Q}\subseteq[K]} \left\{ \frac{d_j^{\mathcal{Q}}}{d_0^{\mathcal{Q}}} \right\} s_j \geq 1.$$

The cut is strongest when each coefficient is smallest, due to the nonnegativity of the slack variables. For $K = 2$, Balas and Qualizza [34, Theorem 6.3] show that it suffices to check two values for each z_k^j, either $\lfloor r_k^j \rfloor$ or $\lceil r_k^j \rceil$, which leads to 2^2 pairs of values to consider; for each pair of values, to find the coefficient on s_j, we simply find the maximum across the four terms of the disjunction, so the total complexity of computing the optimal strengthened coefficient is constant ($4 \times 4 = 16$ comparisons). Though not shown in that paper, it is relatively straightforward to extend their result to general K, in which case the complexity for computing the optimal strengthened coefficient becomes 2^{2K} (2^K possible ways to set $\{z_k^j\}_{k\in[K]}$ and 2^K disjunctive terms to evaluate for each pair).

We next discuss monoidal strengthening, which is more generic but requires the existence of a lower bound for each disjunctive term that is valid for P. Such lower bounds always exist for certain important classes of integer programs, such as when all integer variables are binary, but in general the requirement imposes a genuine limitation on the technique.

1.5.2 Monoidal cut strengthening

As we discussed in Section 1.4.3, any set of row vectors $u_0^t \in \mathbb{R}^{m_t}_{\geq 0}$, $t \in \mathcal{T}$, yields a P_I-free convex set S satisfying the inequalities $u_0^t D^t x \leq u_0^t D_0^t$ for all $t \in \mathcal{T}$. From a given convex set formed in this way, the simple disjunctive cut is defined by Lemma 1.2. We strengthen this cut by the following key tool, which is a subset of $\mathbb{Z}^{|\mathcal{T}|}$:

$$\mathbb{M} := \left\{ m \in \mathbb{Z}^{|\mathcal{T}|} : \sum_{t \in \mathcal{T}} m_t \geq 0 \right\}.^{12}$$

There are two requirements for employing monoidal strengthening, which (as for standard modularization) applies to the integer-restricted variables. The first we have already mentioned: a lower bound $D^t x \geq \ell^t$ that is valid for all $x \in P$. The second is nonnegativity on the coefficients being strengthened. While the former is restrictive, the latter is without loss of generality for nonbasic integer variables: variables are nonbasic in our formulation when they attain a bound in a solution $\bar{x}$, and (possibly by translating or complementing) we can assume that the bound is $x_k \geq 0$ for all $k \in \mathcal{N} \cap \mathcal{I}$. For simplicity, we actually assume that all variables have nonnegativity constraints in the system $Ax \geq b$ (though this assumption is unnecessary for any coefficient not being strengthened). Thus, we can divide $Ax \geq b$ into subsystems $\widetilde{A}x \geq \widetilde{b}$ of $\widetilde{m}$ constraints, and the remaining nonnegativity constraints $x \geq 0$. This formulation leads to a different CGLP. Suppose, for each $t \in \mathcal{T}$, $\widetilde{u}^t$ only refers to the multipliers on the $\widetilde{A}x \geq \widetilde{b}$ constraints (not $x \geq 0$; set u_k^t as the multiplier on $x_k \geq 0$). Define $\alpha_k^t := \widetilde{u}^t \widetilde{A}_{\cdot,k} + u_0^t D^t_{\cdot,k}$ for any $(\widetilde{u}^t, u_0^t) \in \mathbb{R}^{\widetilde{m}}_{\geq 0} \times \mathbb{R}^{m_t}_{\geq 0}$.

We first prove a useful property of vectors in $\mathbb{M}$.

Lemma 1.5. *For each $k \in \mathcal{I}$, let $m^k \in \mathbb{M}$. For every $x \geq 0$ such that $x_k \in \mathbb{Z}$ for all $k \in \mathcal{I}$, the following disjunction is valid:*

$$\left(\sum_{k \in \mathcal{I}} m_t^k x_k = 0 \textit{ for all } t \in \mathcal{T} \right) \vee \left(\bigvee_{t \in \mathcal{T}} \left(\sum_{k \in \mathcal{I}} m_t^k x_k \geq 1 \right) \right).$$

Proof. Observe that $\sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{I}} m_t^k x_k = \sum_{k \in \mathcal{I}} x_k \sum_{t \in \mathcal{T}} m_t^k \geq 0$, as $x \geq 0$. The result then follows: $\sum_{k \in \mathcal{I}} m_t^k x_k < 0$ for any term implies that, for some other term $t' \neq t$, $\sum_{k \in \mathcal{I}} m_{t'}^k x_k > 0$, which is at least 1 (because $m_t^k x_k \in \mathbb{Z}$ for all $t \in \mathcal{T}$ and $k \in \mathcal{I}$). ☐

The strengthening procedure requires a lower bound for each $t \in \mathcal{T}$ and $i \in [m_t]$,

$$\ell_i^t := \min_{x \in P} D_{i\cdot}^t x,$$

which we assume is finite. The value ℓ_i^t provides a lower bound for the corresponding disjunctive term inequality that is valid for P (whereas the disjunction is assumed to impose a tighter condition, i.e., $\ell_i^t < D_{0i}^t$). We can actually replace the condition $x \in P$ by any valid relaxation of P_I, e.g., by tightening P with cuts. Using the lower bounds ℓ^t, Lemma 1.6 shows how we can modify the given disjunction, which we then use for strengthening cuts.

[12] This set is a *monoid*, which is a set of elements equipped with an associative binary operation and containing an identity element. Equivalently, it is a *semigroup* with an identity element.

Lemma 1.6. *For each $k \in \mathcal{I}$, let $m^k \in \mathbb{M}$. The following disjunction is valid for P_I:*

$$\bigvee_{t \in \mathcal{T}} \left(\sum_{k \in \mathcal{I}} \left(D^t_{\cdot,k} + (D^t_0 - \ell^t) m^k_t \right) x_k + \sum_{k \in [n] \setminus \mathcal{I}} D^t_{\cdot,k} x_k \geq D^t_0 \right). \tag{1.4}$$

Proof. Let $\hat{x} \in P_I$. We show that $\hat{x}$ belongs to some term of (1.4), which we reformulate as

$$\bigvee_{t \in \mathcal{T}} \left(\sum_{k \in [n]} D^t_{\cdot,k} x_k \geq D^t_0 - (D^t_0 - \ell^t) \sum_{k \in \mathcal{I}} m^k_t x_k \right).$$

By Lemma 1.5, either all of these disjunctive terms are identical to the original disjunction (when $\sum_{k \in \mathcal{I}} m^k_t \hat{x}_k = 0$ for all $t \in \mathcal{T}$), or $\sum_{k \in \mathcal{I}} m^k_{\hat{t}} \hat{x}_k \geq 1$ for some $\hat{t} \in \mathcal{T}$, in which case

$$\sum_{k \in [n]} D^{\hat{t}}_{\cdot,k} \hat{x}_k \geq \ell^{\hat{t}} = D^{\hat{t}}_0 - (D^{\hat{t}}_0 - \ell^{\hat{t}}) \geq D^{\hat{t}}_0 - (D^{\hat{t}}_0 - \ell^{\hat{t}}) \sum_{k \in \mathcal{I}} m^k_{\hat{t}} \hat{x}_k.$$

Thus, each integer-feasible point satisfies at least one of the terms of the disjunction. □

In Theorem 1.7, we first handle the special case of a SIC that can be derived from the given disjunction. For this theorem, and the subsequent corollary, we do use the assumption that $x \geq 0$, because we state the result with respect to the disjunction (1.1). If we work in the nonbasic space and apply the theorem to (1.3) instead (to strengthening cut coefficients on integer nonbasic variables), the assumption of nonnegativity is automatically satisfied.

Theorem 1.7. *Assume $x \geq 0$ for all $x \in [n]$. Let $u^t_0 \in \mathbb{R}^{m_t}_{\geq 0}$ for each $t \in \mathcal{T}$. Then $\alpha^\intercal x \geq \beta$ is valid for the disjunctive hull, where*

$$\alpha_k = \begin{cases} \inf\limits_{m \in \mathbb{M}} \max\limits_{t \in \mathcal{T}} u^t_0 \left(D^t_{\cdot,k} + (D^t_0 - \ell^t) m_t \right) & \text{for all } k \in \mathcal{I}, \\ \max\limits_{t \in \mathcal{T}} u^t_0 D^t_{\cdot,k} & \text{for all } k \in [n] \setminus \mathcal{I}, \end{cases}$$
$$\beta = \min_{t \in \mathcal{T}} u^t_0 D^t_0.$$

Proof. For a fixed $m \in \mathbb{M}$, the strengthened cut is valid by Lemma 1.2, because of the validity of the disjunction (1.4) due to Lemma 1.6. Taking the infimum over all $m \in \mathbb{M}$ for each $k \in \mathcal{I}$ is valid because the coefficients are set independently of one another. □

When the multipliers u^t_0 satisfy $u^t_0 D^t_0 > 0$ and $u^t_0 (D^t_0 - \ell^t) = 1$ for all $t \in \mathcal{T}$, we can state a more convenient form of the above result, related to the first SIC formula in Section 1.3.

Corollary 1.8. *For each $t \in \mathcal{T}$, let $u^t_0 \in \mathbb{R}^{m_t}_{\geq 0}$ such that $u^t_0 D^t_0 > 0$ and $u^t_0 (D^t_0 - \ell^t) = 1$. Then the cut $\alpha^\intercal x \geq 1$ is valid for the disjunctive hull, where*

$$\alpha_k = \begin{cases} \inf\limits_{m \in \mathbb{M}} \max\limits_{t \in \mathcal{T}} \left\{ \dfrac{u^t_0 D^t_{\cdot,k} + m_t}{u^t_0 D^t_0} \right\} & \text{for all } k \in \mathcal{I}, \\ \max\limits_{t \in \mathcal{T}} \left\{ \dfrac{u^t_0 D^t_{\cdot,k}}{u^t_0 D^t_0} \right\} & \text{for all } k \in [n] \setminus \mathcal{I}. \end{cases}$$

Algorithm 1.1 Calculate Optimal Monoidal Strengthening for Corollary 1.8

```
Input: Polyhedron $P$; disjunction $\vee_{t\in\mathcal{T}}(D^t x \ge D_0^t)$; multipliers $u_0^t \in \mathbb{R}^{m_t}_{\ge 0}$, $t \in \mathcal{T}$.
1:  function StrengthenCut($P, \{D^t, D_0^t\}_{t\in\mathcal{T}}, \{u_0^t\}_{t\in\mathcal{T}}$)
2:      $\gamma_k^t \leftarrow u_0^t D^t_{\cdot,k}$ for each $k \in \mathcal{I}$ and $t \in \mathcal{T}$.
3:      $\gamma_0^t \leftarrow u_0^t D_0^t$ for each $t \in \mathcal{T}$.
4:      $\gamma_k \leftarrow \sum_{t\in\mathcal{T}} \gamma_k^t / \sum_{t\in\mathcal{T}} \gamma_0^t$ for each $k \in \mathcal{I}$.
5:      $\gamma_k \leftarrow \max_t\{\gamma_k^t/\gamma_0^t : t \in \mathcal{T}\}$ for each $k \in [n] \setminus \mathcal{I}$.
6:      for $k \in \mathcal{I}$ do
7:          $m_t^* \leftarrow \gamma_0^t \gamma_k - \gamma_k^t$ for each $t \in \mathcal{T}$.      ▷ Implies $(\gamma_k^t + m_t^*)/\gamma_0^t = \gamma_k$ for all $t \in \mathcal{T}$.
8:          $m_t \leftarrow \lfloor m_t^* \rfloor$ for each $t \in \mathcal{T}$.      ▷ $\sum_{t\in\mathcal{T}} m_t^* = 0$, so $0 \ge \sum_{t\in\mathcal{T}} m_t \ge -(|T|-1)$.
9:          for $(-\sum_{t\in\mathcal{T}} m_t)$ iterations do      ▷ At most $|\mathcal{T}|-1$ iterations.
10:             $m_t \leftarrow m_t + 1$, where $t \in \arg\min_t\{(\gamma_k^t + m_t + 1)/\gamma_0^t : t \in \mathcal{T}\}$.
11:         end for      ▷ At this point, $\sum_{t\in\mathcal{T}} m_t = 0$, so $\{m_t\}_{t\in\mathcal{T}} \in \mathbb{M}$.
12:         $\gamma_k \leftarrow \max_t\{(\gamma_k^t + m_t)/\gamma_0^t : t \in \mathcal{T}\}$.
13:     end for      ▷ Total cost (inner and outer loop) is at most $\mathcal{O}(|T|n)$.
14:     return Cut $\gamma^\mathsf{T} x \ge 1$.
15: end function
```

Applying monoidal strengthening involves choosing a set of values from $\mathbb{M}$ for each $k \in \mathcal{I}$. Efficient techniques exist that apply for a given set of disjunctive term multipliers u_0^t, $t \in \mathcal{T}$ (see, e.g., Algorithm 1.1 based on [28, Algorithm 2]). However, as stated by Balas [17]:

> Overall optimization would of course require the simultaneous choice of the $[u_0^t]$ and the $[m^k]$, but a good method for doing that is not yet available.

This remains an interesting open problem for which progress would have meaningful consequences on the cuts introduced in this dissertation. One possible approach for the multi-branch elementary split disjunction case, which has not been tested to the best of our knowledge, is solving the standard modularization problem via a bilinear program involving simultaneous optimization of u_0^t and f_k^j (the modularization of ray r_k^j).

1.5.3 Strengthening an arbitrary cut

In the discussion so far, we have essentially strengthened the simple disjunctive cut, given as in Lemma 1.2. Next, we apply monooidal strengthening to an arbitrary cut.

Theorem 1.9. *Let $(\alpha, \beta, \{\widetilde{u}^t, u_0^t\}_{t\in\mathcal{T}})$ such that $\alpha_k = \max_t\{\alpha_k^t : t \in \mathcal{T}\}$ for all $k \in \mathcal{I}$ and $\beta \le \min_t\{u^t\widetilde{b} + u_0^t D_0^t : t \in \mathcal{T}\}$. Suppose, for each $t \in \mathcal{T}$, there exists $\ell^t \in \mathbb{R}^{m_t}$ such that $D^t x \ge \ell^t$ for all $x \in P$. Then $\widetilde{\alpha}^\mathsf{T} x \ge \beta$ is valid for P_D, where*

$$\widetilde{\alpha}_k = \inf_{m\in\mathbb{M}} \max_{t\in\mathcal{T}} \left\{\alpha_k^t + u_0^t(D_0^t - \ell^t)m_t\right\} = \alpha_k + \inf_{m\in\mathbb{M}} \max_{t\in\mathcal{T}} \left\{-u_k^t + u_0^t(D_0^t - \ell^t)m_t\right\}$$

for all $k \in \mathcal{I}$, where u_k^t is the Farkas multiplier for $x_k \ge 0$, and $\widetilde{\alpha}_k = \alpha_k$ otherwise.

Proof. It is easy to adapt Lemmas 1.5 and 1.6 to this setting to show that

$$\bigvee_{t\in\mathcal{T}}\left(\begin{array}{c} \widetilde{A}x \geq \widetilde{b} \\ \sum_{k\in\mathcal{I}}(D^t_{\cdot,k} + (D^t_0 - \ell^t)m^k_t)x_k + \sum_{k\in[n]\setminus\mathcal{I}} D^t_{\cdot,k}x_k \geq D^t_0 \end{array}\right).$$

is a valid disjunction for any set of $m^k \in \mathbb{M}$, $k \in \mathcal{I}$. Thus, it is valid to set the coefficient on variable x_k, $k \in \mathcal{I}$, as

$$\widetilde{\alpha}_k = \inf_{m\in\mathbb{M}} \max_{t\in\mathcal{T}} \left\{\widetilde{u}^t\widetilde{A}_{\cdot,k} + u^t_0(D^t_{\cdot,k} + (D^t_0 - \ell^t)m_t)\right\} = \inf_{m\in\mathbb{M}} \max_{t\in\mathcal{T}} \left\{\alpha^t_k + u^t_0(D^t_0 - \ell^t)m_t\right\}.$$

Taking the infimum over all $m \in \mathbb{M}$ for each $k \in \mathcal{I}$ is valid because the coefficients are set independently of one another. The other equivalence comes from $\alpha_k = \alpha^t_k + u^t_k$. □

It is important in the preceding discussion to separate out the Farkas multipliers on the $x \geq 0$ rows of $Ax \geq b$. Without this step, there would be no term u^t_k when calculating the best monoidal strengthening, in which case the optimal $m \in \mathbb{M}$ for each coefficient $k \in \mathcal{I}$ would be $m = 0$, as $u^t_0(D^t_0 - \ell^t) \geq 0$.

Example with elementary split disjunction. Consider the example from Section 1.3 put into minimization form and appended with an additional bound on x_1:

$$\begin{aligned} \min_{x_1,x_2,x_3} \quad & -x_3 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \\ & -x_1 - x_3 \geq -1 \\ & -(3/4)x_2 - x_3 \geq -1/2 \\ & -x_1 \geq -1 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

The SIC from $(-x_1 \geq 0) \vee (x_1 \geq 1)$ and the unique cobasis for $\bar{x} = (1/2, 0, 1)$ can be derived as a feasible solution to (CGLP), with $\alpha^\mathsf{T} = (-1/2, 0, -1)$, $\beta = -1/2$, $u^1 = (0, 3/4, 0, 0, 1, 0)$, $u^1_0 = 1/2$, $u^2 = (0, 0, 0, 1, 0, 0)$, and $u^2_0 = 1/2$. Applying Theorem 1.9, the cut $\widetilde{\alpha}^\mathsf{T}x \geq \beta$ is valid for P_D, where $\widetilde{\alpha}_1 = \alpha_1$, $\widetilde{\alpha}_3 = \alpha_3$ and

$$\begin{aligned} \widetilde{\alpha}_2 &= \alpha_2 + \inf_{m\in\mathbb{M}} \max\left\{-u^1_2 + u^1_0(D^1_0 - \ell^1)m_1, -u^2_2 + u^2_0(D^2_0 - \ell^2)m_2\right\} \\ &= \inf_{m\in\mathbb{M}} \max\left\{-u^1_2 + u^1_0(D^1_0 - \ell^1)m_1, -u^2_2 + u^2_0(D^2_0 - \ell^2)m_2\right\} \\ &= \inf_{m\in\mathbb{M}} \max\left\{-3/4 + (1/2)m_1, (1/2)m_2\right\} = -1/4, \end{aligned}$$

achieved by $m = (1, -1)$. We used the fact that $0 \leq x_1 \leq 1$. The result is exactly the GMIC that we derived in Section 1.3.2.

1.5.4 Strengthening cuts from two-term disjunctions

In this section, we give special treatment to the frequently occurring case of a two-term disjunction. When $|T| = 2$, Algorithm 1.1 returns the optimal monoidal strengthening in at most one iteration per coefficient.

Corollary 1.10. *Suppose $|T| = 2$ and the conditions of Corollary 1.8 are satisfied. The optimal value of the strengthened coefficient on variable x_k, $k \in \mathcal{I}$, is*

$$\alpha_k = \min\left\{\frac{\gamma_j^1 + \lceil z^* \rceil}{\gamma_0^1}, \frac{\gamma_j^2 - \lfloor z^* \rfloor}{\gamma_0^2}\right\}.$$

Proof. For a two-term disjunction, for $m \in \mathbb{M}$, $m_1 + m_2 = 0$ implies that $m_1 = -m_2$, so it suffices to consider a single value $z \in \mathbb{Z}$. As in the algorithm, for $k \in \mathcal{I}$, we first choose z^* such that $(\gamma_k^1 + z^*)/\gamma_0^1 = (\gamma_k^2 - z^*)/\gamma_0^2$, which we can recalculate, or, using the formula,

$$z^* = \gamma_0^1 \gamma_k - \gamma_k^1 = \gamma_0^1\left(\frac{\gamma_k^1 + \gamma_k^2}{\gamma_0^1 + \gamma_0^2}\right) - \gamma_k^1 = \frac{\gamma_0^1\gamma_k^2 - \gamma_0^2\gamma_k^1}{\gamma_0^1 + \gamma_0^2} = \frac{\gamma_k^2/\gamma_0^2 - \gamma_k^1/\gamma_0^1}{1/\gamma_0^1 + 1/\gamma_0^2}.$$

One iteration in Algorithm 1.1 yields the optimal strengthening. In this case, the initial values are $m_1 = \lfloor z^* \rfloor$ and $m_2 = \lfloor -z^* \rfloor = -\lceil z^* \rceil$. If $z^* \in \mathbb{Z}$, then the solution is already optimal. Otherwise, we perform one update, in which either m_1 or m_2 increments based on

$$\min\left\{\frac{\gamma_k^1 + \lfloor z^* \rfloor + 1}{\gamma_0^1}, \frac{\gamma_k^2 - \lceil z^* \rceil + 1}{\gamma_0^2}\right\} = \min\left\{\frac{\gamma_k^1 + \lceil z^* \rceil}{\gamma_0^1}, \frac{\gamma_k^2 - \lfloor z^* \rfloor}{\gamma_0^2}\right\}.$$

The latter expression is precisely the desired cut coefficient. □

The result also can be derived for an arbitrary valid disjunctive cut.

Corollary 1.11. *Consider the setting of Theorem 1.9 with the additional assumptions that $|T| = 2$ and $D_0^t - \ell^t = 1$ for $t \in \{1, 2\}$. The optimal strengthened coefficient on x_k, $k \in \mathcal{I}$, is*

$$\widetilde{\alpha}_k = \min\left\{\alpha_k^1 + u_0^1 \lceil z^* \rceil, \alpha_0^2 - u_0^2 \lfloor z^* \rfloor\right\}$$

where

$$z^* = \frac{\alpha_k^2 - \alpha_k^1}{u_0^1 + u_0^2}.$$

Proof. From Theorem 1.9, for any $m \in \mathbb{M}$, the strengthened coefficient satisfies

$$\widetilde{\alpha}_k = \max\{\alpha_k^1 + u_0^1 m_1, \alpha_k^2 + u_0^2 m_2\}.$$

For $|T| = 2$, we can set $m_1 = -m_2$. In the best-case scenario, both terms in the maximum are equal. This is precisely achieved by $m_1 = z^*$. If $z^* \in \mathbb{Z}$, this is clearly the optimal solution. Otherwise, we perform one update step adapted from Algorithm 1.1. Namely, if $m_1 = \lfloor z^* \rfloor$ and $m_2 = \lfloor -z^* \rfloor = -\lceil z^* \rceil$, then $m_1 + m_2 = -1$. Updating m_1 to equal $\lceil z^* \rceil$, or $m_2 = -\lfloor z^* \rfloor$, results in $m \in \mathbb{M}$. The best choice is the one that minimizes the resulting maximum. The strengthened coefficient is then exactly as in the theorem statement. □

Application to split disjunctions. It is instructive to apply Corollary 1.10 to a SIC derived from a split disjunction $(-\pi^\mathsf{T} x \geq -\lfloor \pi^\mathsf{T} \bar{x} \rfloor) \vee (\pi^\mathsf{T} x \geq \lceil \pi^\mathsf{T} \bar{x} \rceil)$, where $\pi \in \mathbb{Z}^n$. We work with the disjunction expressed in the nonbasic space (as in (1.3)):

$$\left(-\pi^\mathsf{T} A_N^{-1} s_\mathcal{N} \geq f_0\right) \vee \left(\pi^\mathsf{T} A_N^{-1} s_\mathcal{N} \geq 1 - f_0\right).$$

Let r_j denote the entry of $\pi^\mathsf{T} A_N^{-1}$ corresponding to index $j \in \mathcal{N}$. We have the following values in the notation of Algorithm 1.1.

$$\begin{aligned} \gamma_0^1 &= f_0 & \\ \gamma_0^2 &= 1 - f_0 & \\ \gamma_j^1 &= -r_j && \text{for all } j \in \mathcal{N} \\ \gamma_j^2 &= r_j && \text{for all } j \in \mathcal{N} \end{aligned}$$

The SIC coefficient on variable s_j, $j \in \mathcal{N}$, is

$$\max\left\{\frac{-r_j}{f_0}, \frac{r_j}{1-f_0}\right\} = \max\left\{\frac{\gamma_j^1}{\gamma_0^1}, \frac{\gamma_j^2}{\gamma_0^2}\right\}.$$

Hence, to compute the optimal strengthening of the coefficient on a variable s_j, $j \in \mathcal{N} \cap \mathcal{I}$,

$$z^* = \frac{\gamma_0^1 \gamma_j^2 - \gamma_0^2 \gamma_j^1}{\gamma_0^1 + \gamma_0^2} = f_0 r_j + (1 - f_0) r_j = r_j.$$

Then the optimal strengthened coefficient (in the nonbasic space) is

$$\gamma_j = \min\left\{\frac{-r_j + \lceil r_j \rceil}{f_0}, \frac{r_j - \lfloor r_j \rfloor}{1 - f_0}\right\}.$$

When $\pi = e_k$ (i.e., the disjunction is simple), we recover precisely the formula for strengthening a SIC to a GMIC as in Section 1.3. As a generalization, Fischer and Pfetsch [83] recently show that, in the case of any two-term disjunction, monoidal cut strengthening is equivalent to a derivation of a generalized mixed-integer rounding cut. The preceding discussion also shows the equivalence of monoidal strengthening with standard modularization for split disjunctions.

1.6 Contributions of this dissertation

In Chapter 2, we discuss a generalization of SICs called *generalized intersection cuts* (GICs). The GIC paradigm for generating cuts is relatively recent and has attractive theoretical properties. We investigate this previously computationally unexplored paradigm and observe that a key hyperplane activation procedure embedded in it is not computationally viable. To overcome this issue, we develop a novel replacement to this procedure called *partial hyperplane activation* (PHA), introduce a variant of PHA based on a notion of hyperplane tilting, and prove the validity of both algorithms. We propose several implementation strategies and parameter choices for our PHA algorithms and provide supporting theoretical results. Our findings shed light on the the strengths of the PHA approach, as well as identify properties related to strong cuts that we subsequently utilize in later chapters.

Chapter 3 examines a different type of tilting that can be used to produce cutting planes from split disjunctions starting from any given valid inequality. Geometrically, this tilting involves changing the angle of the given inequality until it becomes supporting for both sides of the disjunction. We provide some computational evidence of the strength and practicality of this procedure and make connections with existing literature, such as lifting and coefficient strengthening, for which tilting offers a unifying perspective.

In Chapter 4, we introduce the *$\mathcal{V}$-polyhedral cut* (VPC) approach for generating valid inequalities from general disjunctions. We show how to efficiently obtain points and rays from which we build a linear program whose feasible solutions correspond to valid disjunctive cuts. This linear program is much smaller than the one for L&PCs. This enables us to test larger disjunctions than those previously used, coming from the leaf nodes of a partial branch-and-bound tree. We show the cuts from one large disjunction significantly improve the gap closed compared to the cuts produced from a union of many shallow disjunctive sets, where we compare against GMICs and the cuts from the default settings for the commercial solver `Gurobi`. Our cuts also decrease `Gurobi` solving time. However, this hinges on choosing the best partial tree size per instance; this is a difficult quantity to predict and doing so remains an open problem that would benefit from future work on a better understanding of the interaction between branch-and-bound and cuts.

Chapter 5 outlines ongoing work toward applying cut strengthening techniques (as discussed in Section 1.5) to VPCs. The main difficulty is that the we generate VPCs in the original space of P, rather than through the CGLP, and thus we lack the requisite Farkas multipliers needed to strengthen the cuts. Our contribution is a mapping from bases of the linear program generating VPCs to bases of (CGLP) from the same disjunction, providing an efficient mechanism for calculating the strengthening coefficients.

The appendices contain additional theoretical and supplemental results. Appendix A constitutes a brief discussion on the correspondence between simple disjunctive cuts and regular bases of the CGLP. In Appendix B, we give a geometric interpretation of three normalizations for (CGLP) and derive the optimal solution to (MLP) and (CGLP) under the so-called trivial normalization. It is known that the trivial normalization for (CGLP) yields a SIC, but this depends on the assumption that the point being separated is vertex of P. We show an example in which the result is an irregular cut when this assumption is not satisfied. The subsequent appendices provide examples and supplemental tables for the main chapters of the thesis.

Lastly, all code produced for this dissertation is available in an open-source implementation that interfaces with the COIN-OR Cut Generation Library [55, 112].[13]

1.7 Notation

Table 1.1 summarizes notation frequently appearing in the dissertation.

[13]`https://bitbucket.org/akazachk/pointcuts4/`

Table 1.1: Definitions of frequently used notation.

Notation	Description
$\mathbb{R}$; $\mathbb{Z}$	Set of real numbers; set of integer numbers
$\mathbb{R}_{\geq 0}$; $\mathbb{Z}_{\geq 0}$	Set of nonnegative reals; set of nonnegative integers
$\operatorname{int} S$; $\operatorname{relint} S$	Interior and relative interior of a region S
$\operatorname{bd} S$; $\operatorname{relbd} S$	Boundary and relative boundary of a region S
$\operatorname{conv} S$	Convex hull of the points in a set S
$[n]$	$\{1, \ldots, n\}$
$A_{i\cdot}$; $A_{\cdot j}$	Row i and column j of matrix A
P	Rational convex polyhedron defined by $\{x \in \mathbb{R}^n : A_{i\cdot}x \geq b_i,\ i \in [m]\}$
$\mathcal{I}$	Subset of $[n]$ indexing the integer variables
P_I	$\{x \in P : x_j \in \mathbb{Z} \text{ for all } j \in \mathcal{I}\}$
$\bar{x}$	Optimal extreme point solution to $\min_x\{c^{\mathsf{T}}x : x \in P\}$
$D^t x \geq D_0^t$	Disjunctive inequalities defining term $t \in \mathcal{T}$ of disjunction (1.1)
P^t	Disjunctive term $t \in \mathcal{T}$, $\{x \in P : D^t x \geq D_0^t\}$
P_D	Disjunctive hull, $\operatorname{conv}(\cup_{t \in \mathcal{T}} P^t)$
S	Convex set satisfying $P_I \cap \operatorname{int}(S) = \emptyset$
p^t	Optimal extreme point solution to $\min_x\{c^{\mathsf{T}}x : x \in P^t\}$
$\mathcal{N}(x)$	Cobasis, or index set of nonbasic variables, for a vertex $x \in P$
$\mathcal{N}$; $\mathcal{N}^t$	$\mathcal{N}(\bar{x})$; $\mathcal{N}(p^t)$
N	Indices of constraints of P corresponding to a given cobasis
$A_N x \geq b_N$	Subsystem of $Ax \geq b$ containing constraints indexed by N
$C(\mathcal{N}(x))$	LP basis cone, the intersection of the constraints corresponding to $\mathcal{N}(x)$

Chapter 2

Partial Hyperplane Activation for Generalized Intersection Cuts

This chapter is based on joint work with Egon Balas, François Margot, and Selvaprabu Nadarajah.

2.1 Introduction

As part of the recent active research on new general-purpose cutting planes, Balas and Margot [31] introduced the GIC paradigm with the motivation of finding stronger cutting planes that also possess favorable numerical properties, such as avoiding numerical inaccuracies that arise in traditional cutting plane approaches [139]. Despite offering some theoretical advantages to other modern cutting planes, GICs have remained unexplored computationally. In this chapter, we observe that generating GICs as outlined in [31] is computationally intractable due to the exponential size of the linear program used to generate cuts. We offer a solution by extending the notion of a GIC and devising new algorithms to generate such GICs that scale well with the size of the input data. We prove the validity of our algorithms, provide theoretical results that guide our implementation choices, and perform the first computational investigation with GICs. Our investigation identifies properties related to strong cuts that can be targeted in future approaches.

Let C be a relaxation of P defined by a subset of the inequalities defining P. Balas and Margot [31] produce GICs from a collection of intersection points and rays obtained by intersecting the edges of C with the boundary of a convex set S containing no points from P_I in its interior. The simplest GICs are the SICs [21], which use as C a simple polyhedral cone $\bar{C}$ with apex at a vertex $\bar{x}$ of P. GICs generalize SICs by using a tighter relaxation of P, by *activating* additional hyperplanes of P that are not included in the description of $\bar{C}$. However, hyperplane activation poses two computational issues: first, maintaining a description of C becomes challenging, and second, the number of points and rays can quickly grow too large for any practical use.

This chapter introduces a new method, called PHA, that addresses the issues with the aforementioned *full* hyperplane activation proposed in [31]. The key insight underlying PHA is that we can forgo a complete description of the polyhedron C by always activating hyperplanes on the initial cone $\bar{C}$, instead of on an iteratively refined relaxation. We show the PHA approach is not only valid, but also generates a collection of intersection points and rays that grows quadratically in size with the number of hyperplane activations, in contrast to the exponential growth exhibited by the full activation procedure.

Activating a hyperplane on $\bar{C}$ creates new vertices lying at most one edge of P away from $\bar{x}$. Consequently, intersection points are obtained from edges originating at $\bar{x}$ or these distance 1 vertices. Higher-distance PHA methods, with accompanying higher computational cost, can be easily defined by incorporating hyperplane intersections with edges a larger distance from $\bar{x}$, eventually recovering the full activation procedure. In other words, while we focus here on the details of a distance 1 PHA procedure, its underlying ideas extend to a hierarchy of PHA procedures delivering progressively "stronger" sets of intersection points.

An important observation related to the effectiveness of PHA is that weak intersection points are created whenever a hyperplane being activated intersects rays of $\bar{C}$ that do not intersect the boundary of S. To mitigate this issue, we introduce a class of *tilted* hyperplanes that can be used to avoid such rays. We also show that tilted hyperplanes offer more control over the number of intersection points generated with essentially no additional overhead, but they require a subtle condition to ensure that these points lead to cuts *valid* for P_I, i.e., to inequalities that do not cut off any points of P_I. We use this idea to design a modified PHA algorithm that creates a point-ray collection that grows linearly with the number of hyperplane activations but quadratically in the number of rays of $\bar{C}$ being cut. The notion of tilting may also have independent merit outside of the GIC paradigm whenever a $\mathcal{V}$-polyhedral partial description could be useful, i.e., describing a polyhedron by some of its vertices and edges (see, e.g., [140]).

Implementing PHA or its variant with tilting requires several algorithmic design choices, such as the decision of which particular hyperplanes to activate and which objective functions to use when optimizing over the derived collection of intersection points and rays. Our theoretical contributions provide geometric and structural insights into making some of these choices. Our experiments test the strength of GICs with respect to root node gap closed, a commonly used measure of cut strength, based on three hyperplane activation rules and four types of objective directions.

The first hyperplane activation rule we consider is based on the full activation method, which sequentially activates hyperplanes by pivoting to neighbors of $\bar{x}$. The second rule targets points that lie deep relative to the SIC. The third rule is motivated by a pursuit of a certain class of intersection points called *final*, relating to facet-defining inequalities for the S_k*-closure*, which is a relaxation of the split closure obtained by refining P through the addition of one elementary split disjunction on a variable x_k.

We then discuss the objective directions evaluated in our implementation, which contribute to a better understanding of the process by which a collection of strong cuts can be generated. The first two sets of objective functions, as with the hyperplane rules, are motivated by finding cuts that improve over the SIC, as well as generalizing the usual objective of maximizing violation with respect to $\bar{x}$. We then give a result motivating the next set of directions, by showing that the optimal objective value over the S_k-closure can be obtained using only final intersection points and describing how to attain this same value using cuts. The fourth objective function type capitalizes on information that can be used from the typical empirical setup in which multiple split disjunctions are applied in parallel.

Finally, we present the first computational results for any algorithm within the GIC paradigm, implemented in the open source COIN-OR framework [112] and using a set of benchmark instances from MIPLIB [43]. We find that our PHA approach creates a manageable number of intersection points from which we can generate GICs that effectively generalize SICs: GICs generated from our methodology improve the *integrality gap* closed by one round of SICs on 72.5% of the instances tested, closing on average an additional 5.5% of the gap over SICs for the instances with an improvement. Underlying these results is a detailed analysis of the various hyperplane activation and objective function choices for the linear program used to generate GICs. These experiments build an understanding of structural properties of strong GICs. In particular, we find that the objective functions inspired by final intersection points frequently find the strongest cuts from a given point-ray collection compared to the other objectives tested. However, such final intersection points appear infrequently in the point-ray collections generated by PHA, yielding a concrete direction for future work: to target final intersection points more directly.

Our research on GICs adds to the growing literature that generalizes and extends cutting plane methods based on intersection cuts and Gomory cuts. SICs are equivalent to Gomory cuts [89, 90, 91], a relationship surveyed in Conforti et al. [57], and are among the simplest, yet most effective, cuts used in practice. As SICs are generated using information from only one row of the simplex tableau, GICs are a natural generalization by using information from multiple rows. Using such information has been studied extensively in the past decade [9, 66, 68, 69, 70, 79, 114]. The focus of these approaches is to obtain stronger cuts by using a better cut-generating set obtained from the optimal solution to the continuous relaxation. GICs instead attempt to strengthen SICs using other rows but the same cut-generating set. The collection of intersection points and rays used to derive GICs can also be seen as defining a linear system for verifying the validity of cutting planes [73].

Broadly, GICs fall into the class of algorithms using the classical theoretical approach of polarity to generate cuts [17]. This includes similarities to local or target cuts [51], which use the polar of a set of points and rays to generate cuts, and to the procedures in [114, 127], which use row generation to individually certify the validity of each generated cut. Though utilizing some of the same theoretical tools, GICs derive points and rays from a fundamentally different perspective and in a way that immediately guarantees cut validity.

The rest of this chapter is organized as follows. In Section 2.2, we revisit full hyperplane activation from [31] and demonstrate the impediment to a practical GIC algorithm inherent in that approach. Section 2.3 then gives our main contribution, the PHA scheme. Section 2.4 shows the idea of tilting and gives an additional algorithm that can be used within the PHA procedure. Implementation choices and supporting theory are discussed in Section 2.5. Lastly, we give the results of our computational experiments in Section 2.7. Additional supporting theoretical results are contained in the appendix.

2.2 Full hyperplane activation

In this section, we show the impracticality of generating GICs by full hyperplane activation as proposed by Balas and Margot [31], which provides the motivation for developing our new methodology. For simplicity, we assume throughout that P is a full-dimensional pointed polyhedron, but our results extend to the general case with minor modifications.

Let S be a *P_I-free convex set*, a closed convex set such that its *interior*, denoted $\operatorname{int} S$, contains no points of P_I, and suppose that $\bar{x} \in \operatorname{int} S$. A commonly used such set is formed from an *elementary* split disjunction $(x_k \leq \lfloor \bar{x}_k \rfloor) \vee (x_k \geq \lceil \bar{x}_k \rceil)$ on a variable x_k, $k \in \mathcal{I}$:

$$S_k := \{x : \lfloor \bar{x}_k \rfloor \leq x_k \leq \lceil \bar{x}_k \rceil\}.$$

Let $\bar{\mathcal{N}} := \mathcal{N}(\bar{x})$ denote the set of nonbasic variables at $\bar{x}$. Let $\bar{C} := C(\bar{\mathcal{N}})$ be the polyhedral cone with apex at $\bar{x}$ and defined by the n hyperplanes corresponding to the nonbasic variables $\bar{\mathcal{N}}$; denote these hyperplanes by $\bar{\mathcal{H}}$. Let $\bar{\mathcal{R}}$ be the rays of $\bar{C}$.

GICs generalize SICs, as the cone $\bar{C}$ is but one possible relaxation of P. $\bar{C}$ has n extreme rays that can be intersected with $\operatorname{bd} S$ to obtain a set of intersection points $\mathcal{P}_0$ and a set of rays $\mathcal{R}_0$ of $\bar{C}$ that do not intersect $\operatorname{bd} S$, so that

$$\begin{aligned} \mathcal{R}_0 &:= \{r \in \bar{\mathcal{R}} : r \cap \operatorname{bd} S = \emptyset\}, \\ \mathcal{P}_0 &:= \{p^r : p^r := r \cap \operatorname{bd} S,\ r \in \bar{\mathcal{R}} \setminus \mathcal{R}_0\}. \end{aligned}$$

The intersection points in $\mathcal{P}_0$ and rays in $\mathcal{R}_0$ uniquely define the SIC, $\alpha_0^\mathsf{T} x \geq \beta_0$, obtained from $\bar{C}$ and S [17].

Letting $\mathcal{H}$ denote the set of hyperplanes of P, $\bar{C}$ can be replaced by a tighter relaxation of P, denoted C, obtained by activating a subset of hyperplanes from $\mathcal{H} \setminus \bar{\mathcal{H}}$, i.e., of those that define P but not $\bar{C}$. Subsequently, edges of C can be intersected with $\operatorname{bd} S$ to obtain a set of intersection points $\mathcal{P}$, and edges that do not intersect $\operatorname{bd} S$ yield a set of rays $\mathcal{R}$. In contrast to the situation when using $\bar{C}$, the number of intersection points and rays obtained from C intersected with $\operatorname{bd} S$ can be strictly greater than n, meaning the point-ray collection defines not just one cut, but a collection of valid cuts.

Theorem 4 in [31] defines valid GICs as these cuts and propose to generate them as any $(\alpha, \beta) \in \mathbb{R}^n \times \mathbb{R}$ satisfying $\alpha^\mathsf{T} \bar{x} < \beta$ and

$$\begin{aligned} \alpha^\mathsf{T} p &\geq \beta \quad \text{for all } p \in \mathcal{P} \\ \alpha^\mathsf{T} r &\geq 0 \quad \text{for all } r \in \mathcal{R} \end{aligned} \tag{2.1}$$

For a fixed β, define $\operatorname{CutRegion}(\beta, \mathcal{P}, \mathcal{R})$ as the vectors α feasible to the above system, i.e., the coefficients for inequalities with lower-bound β that are valid for $\mathcal{P}$ and $\mathcal{R}$. It suffices to consider $\beta \in \{-1, 0, 1\}$ to obtain all possible cuts.

Unfortunately, the above method requires maintaining the $\mathcal{V}$-polyhedral description of C. The size of this $\mathcal{V}$-polyhedral description, as well as the number of rows of (2.1), as shown in Proposition 2.1, can grow exponentially large in the number of hyperplanes defining C. Together, these two issues make the full hyperplane activation approach unviable.

Proposition 2.1. *Let C be formed by adding k_h hyperplanes to the description of $\bar{C}$, and let $(\mathcal{P}, \mathcal{R})$ denote the point-ray collection obtained from intersecting the edges of C with* $\operatorname{bd} S$. *The cardinality of $\mathcal{P}$ can grow exponentially large in k_h.*

Proof. Suppose we start with $\bar{C}$ and consider the activation of a halfspace H^+. Let $\mathcal{R}(H)$ denote the rays from $\bar{\mathcal{R}}$ that are intersected by H before $\operatorname{bd} S$. For each ray $r \in \mathcal{R}(H)$ that intersects H, $n - 1 - |\mathcal{R}(H)|$ new edges are created (not counting the edge back to $\bar{x}$ or the edges between new vertices created by activating H). If $|\mathcal{R}(H)| \approx n/2$, the size of the new point-ray collection will be $O(n^2)$. The desired result follows inductively. □

2.3 Partial hyperplane activation

In this section, we propose PHA, an approximate activation method that overcomes the exponential growth in the size of the point-ray collection when using full hyperplane activation. Specifically, our algorithm generates at most $O(k_r^2 k_h)$ intersection points and rays, where k_r and k_h are both parameters; $k_r \leq n$ is the maximum number of initial intersection points and rays that we remove via hyperplane activations and k_h is the number of activated hyperplanes.

Showing the validity of PHA requires a strict extension of the tools used in [31] to prove the validity of full hyperplane activation. We give this extension in Section 2.3.1 and employ it in Section 2.3.2, in which we describe a concrete variant of PHA called PHA$_1$ and prove its validity. To facilitate reading, Table 2.1 gives a summary of some of the most frequently used notation.

Table 2.1: Summary of frequently used notation in this chapter.

Notation	Description
$\mathcal{H}$	Hyperplanes defining P
H_h; H_h^+; H_h^-	$H_h = \{x : a_h^\mathsf{T} x = b_h\}$; $H_h^+ = \{x : a_h^\mathsf{T} x \geq b_h\}$; $H_h^- = \mathbb{R}^n \setminus H_h^+$
$\bar{\mathcal{N}}$	Set of nonbasic variables defining $\bar{x}$
$\bar{C}$	Simple cone defined by the nonbasic variables at $\bar{x}$
$\bar{\mathcal{R}}$	Rays of $\bar{C}$
$\bar{\mathcal{H}}$	Hyperplanes defining $\bar{C}$
$(\mathcal{P}, \mathcal{R})$	Set of points on $\operatorname{bd} S$ and rays not intersecting $\operatorname{bd} S$
$(\mathcal{P}_0, \mathcal{R}_0)$	Initial point-ray collection from $\bar{C} \cap \operatorname{bd} S$
$\operatorname{CutRegion}(\bar{\beta}, \mathcal{P}, \mathcal{R})$	Feasible region of (2.2)
$\mathcal{R}_\mathcal{P}$	$\{p - \bar{x} : p \in \mathcal{P}\}$
$\widehat{C}(\mathcal{P}, \mathcal{R})$	$\bar{x} + \operatorname{cone}(\mathcal{R}_\mathcal{P} \cup \mathcal{R})$
$\mathcal{G}(\mathcal{P}, \mathcal{R})$	$\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$
$\mathcal{R}(H_h)$	Rays from $\bar{\mathcal{R}}$ intersected by H_h before $\operatorname{bd} S$

2.3.1 Proper point-ray collections

We define a point-ray collection as a pair $(\mathcal{P}, \mathcal{R})$ of points and rays obtained from intersecting the polyhedron C with $\operatorname{bd} S$, where C is a relaxation of P. Let $\mathcal{K}$ denote the skeleton of the polyhedron C. Let $\mathcal{K}'$ denote the connected component of $\mathcal{K} \cap \operatorname{int} S$ that includes $\bar{x}$ and $\mathcal{K}''$ denote the union of the other components of $\mathcal{K} \cap \operatorname{int} S$.

Definition 2.2. *The point-ray collection $(\mathcal{P}, \mathcal{R})$ is called* proper *if $\alpha^\mathsf{T} x \geq \beta$ is valid for P_I whenever (α, β) is feasible to* (2.1) *and $\alpha^\mathsf{T} v < \beta$ for some $v \in \mathcal{K}'$.*

With this definition, Theorem 2.3 shows that full hyperplane activation creates a proper point-ray collection, which follows easily from the proof of Theorem 4 in [31].

Theorem 2.3. *The point-ray collection $(\mathcal{P}, \mathcal{R})$ obtained from intersecting all edges of C with $\operatorname{bd} S$, where C is defined by a subset of the hyperplanes defining P, is proper.*

Given a proper point-ray collection, a GIC is defined as a facet of the convex hull of points and rays, $\mathcal{G}(\mathcal{P}, \mathcal{R}) := \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$, referred to by $\mathcal{G}$ for short, that cuts off a vertex of $\mathcal{K}'$. Balas and Margot [31] show that these facets can be generated by solving the following linear program formed from the given points and rays and optimized with respect to a given objective direction $w \in \mathbb{R}^n$ and a fixed $\beta \in \{-1, 0, 1\}$:

$$\min_{\alpha}\{w^{\mathsf{T}}\alpha : \alpha \in \operatorname{CutRegion}(\beta, \mathcal{P}, \mathcal{R})\}. \tag{2.2}$$

In fact, we show something stronger in Lemma 2.4: the cuts we obtain are not only valid for P_I, but also for $\operatorname{conv}(C \setminus \operatorname{int} S)$. This extends Theorem 2.3 and will be invoked in proving the results in Section 2.3.2. For simplicity, we assume that the vertex $v \in \mathcal{K}'$ being cut in Definition 2.2 is $\bar{x}$ in what follows, unless specified otherwise.

Lemma 2.4. *Let $(\mathcal{P}, \mathcal{R})$ be the point-ray collection obtained from intersecting all edges of C with $\operatorname{bd} S$. If (α, β) is feasible to (2.1) and $\alpha^{\mathsf{T}}\bar{x} < \beta$, then*

$$F := \{x \in C : \alpha^{\mathsf{T}}x < \beta\} \subseteq \operatorname{int} S.$$

Proof. Assume for the sake of contradiction that $F \setminus \operatorname{int} S$ is nonempty. Then $F \cap \operatorname{bd} S$ is nonempty. We show that this implies an intersection point or ray of C will violate $\alpha^{\mathsf{T}}x \geq \beta$.

First, note that $\operatorname{cl} F \cap \operatorname{bd} S$ cannot contain any vertex violating the inequality $\alpha^{\mathsf{T}}x \geq \beta$, as each vertex of $\operatorname{cl} F \cap \operatorname{bd} S$ is an intersection point of C. Hence, as we are also assuming $F \cap \operatorname{bd} S$ is nonempty, the linear program $\min\{\alpha^{\mathsf{T}}x : x \in \operatorname{cl} F \cap \operatorname{bd} S\}$ must be unbounded. Consider initializing this program at $\bar{x}$ and proceeding by the simplex method using the steepest edge rule. As the simplex method follows edges of C and it will never increase the objective value, we will either eventually intersect $\operatorname{bd} S$ or end up on a ray of C that is parallel to $\operatorname{bd} S$. This is the desired contradiction: in the former case, this is an intersection point that violates the inequality, and in the latter case, this is a ray of C that does not intersect $\operatorname{bd} S$ and violates the inequality. $\square$

2.3.2 Algorithm and validity

We describe the PHA$_1$ approach for activating a hyperplane partially and prove that it yields a proper point-ray collection. The algorithm proceeds similarly to the full hyperplane activation procedure, in that it activates a hyperplane valid for P, but unlike the procedure from [31], PHA$_1$ can use any hyperplane valid for P (not necessarily one that defines P), and the activation is performed on the initial relaxation $\bar{C}$ instead of an iteratively refined relaxation that includes previously activated hyperplanes. We show this is not only valid but also computationally advantageous. Part of this advantage comes from that fact that PHA$_1$ only requires primal simplex pivots in the tableau of the LP relaxation to calculate new intersection points and rays.

The trade-off for the computational advantages of PHA$_1$ is that it potentially creates a weaker point-ray collection compared to full activation, in the sense that the cuts generated

Algorithm 2.2 Distance 1 Partial Hyperplane Activation

```
Input: Polyhedron $P$ defined by hyperplane set $\mathcal{H}$; $P_I$-free convex set $S$; hyperplane $H_h$
    valid for $P$; set of rays $\mathcal{R}_A \subseteq \bar{\mathcal{R}}$; proper point-ray collection $(\mathcal{P}, \mathcal{R})$ contained in $\bar{C}$.
 1: function PHA$_1(P, S, H_h, \mathcal{R}_A, (\mathcal{P}, \mathcal{R}))$
 2:     for all $r^j \in \mathcal{R}_A$ such that $\operatorname{dist}(H_h, r^j) < \operatorname{dist}(\operatorname{bd} S, r^j)$ do
 3:         Remove all points and rays in $(\mathcal{P}, \mathcal{R})$ that violate $H_h^+$ and originate from
                a vertex on $r^j$ but do not lie on or coincide with any ray from $\bar{\mathcal{R}} \setminus \mathcal{R}_A$.
 4:         for all $r^{jhi}$ originating at vertex $r^j \cap H_h$ do
 5:             if $\operatorname{dist}(H, r^{jhi}) < \operatorname{dist}(\operatorname{bd} S, r^{jhi})$ for some $H \in \bar{\mathcal{H}}$ not defining $r^j$ then
 6:                 Skip this ray.
 7:             else if $r^{jhi} \cap \operatorname{bd} S \neq \emptyset$ then
 8:                 Add $p^{jhi} := r^{jhi} \cap \operatorname{bd} S$ to $\mathcal{P}$.
 9:             else Add $r^{jhi}$ to $\mathcal{R}$.
10:     return $\mathcal{P}$ and $\mathcal{R}$.
11: end function
```

from this collection may be weaker. This is because all vertices created from the partial activations are restricted to being only one pivot away from $\bar{x}$, i.e., at (edge) distance 1 from $\bar{x}$ along the skeleton of P. For this reason, we refer to Algorithm 2.2 as PHA$_1$. A stronger version of PHA$_1$, such as a valid distance 2 (or higher distance) procedure, is not difficult to create. In other words, one could define a hierarchy of PHA procedures of increasing distance using the principles presented below, eventually leading to the full activation procedure.

We now define some notation used in the algorithm description. For a given $(a_h, b_h) \in \mathbb{R}^n \times \mathbb{R}$, let $H_h := \{x : a_h^\mathsf{T} x = b_h\}$ be a hyperplane such that the corresponding halfspace $H_h^+ := \{x : a_h^\mathsf{T} x \geq b_h\}$ is valid for P. Every element of a proper point-ray collection $(\mathcal{P}, \mathcal{R})$ arises from the intersection of an edge of some relaxation of P with $\operatorname{bd} S$; if the starting vertex of that edge lies on a ray $r^j \in \bar{\mathcal{R}}$, then we say that the point or ray *originates* from r^j. For each $r^j \in \bar{\mathcal{R}}$, let the distance to H_h along the ray r^j starting at $\bar{x}$ be

$$\operatorname{dist}(H_h, r^j) := \begin{cases} (b_h - a_h^\mathsf{T} \bar{x}) / a_h^\mathsf{T} r^j & \text{if } a_h^\mathsf{T} r^j > 0 \\ \infty & \text{otherwise.} \end{cases}$$

We will slightly overload this notation to also refer to the distance along ray r^j to $\operatorname{bd} S$, $\operatorname{dist}(\operatorname{bd} S, r^j)$. This will be the distance along r^j from $\bar{x}$ to the nearest facet of S (or ∞ if no facet of S is intersected). With these definitions, we present Algorithm 2.2, which takes as an input the polyhedron P, the cut-generating set S, the hyperplane H_h being activated, a set of rays $\mathcal{R}_A$ from $\bar{\mathcal{R}}$ that will be permitted to be cut by H_h (a parameter we will use only later), and any proper point-ray collection to be modified via the activation of H_h. We also assume that along with the point-ray collection, we have kept a history of which edge led to each point and ray to be added to the collection. This is used in the algorithm in step 3 when deciding which points and rays to remove.

Theorem 2.5 states the validity of Algorithm 2.2 for the special case when all rays of $\bar{C}$ are permitted to be cut. We show how to relax this restriction in Section 2.4.

Theorem 2.5. *Algorithm 2.2 when $\mathcal{R}_A = \bar{\mathcal{R}}$ outputs a proper point-ray collection.*

In the rest of this section, we will build to a proof of Theorem 2.5 with the help of several useful intermediate results. The validity of PHA_1 follows from proving that the inequalities that can be generated from (2.2) are valid for a relaxation of P_I related only to $\bar{x}$ and the points and rays in the collection. Unlike the proof of full hyperplane activation, this does not rely on any particular relaxation of P used to obtain the points and rays. Given a point-ray collection $(\mathcal{P}, \mathcal{R})$, let $\mathcal{R}_{\mathcal{P}} := \{p - \bar{x} : p \in \mathcal{P}\}$ and $\widehat{C}(\mathcal{P}, \mathcal{R}) := \bar{x} + \operatorname{cone}(\mathcal{R}_{\mathcal{P}} \cup \mathcal{R})$. That is, $\widehat{C}(\mathcal{P}, \mathcal{R})$ is the polyhedral cone with apex at $\bar{x}$ and rays including $\mathcal{R}$ and a ray from $\bar{x}$ through each intersection point in $\mathcal{P}$.

Lemma 2.6. *If $(\mathcal{P}, \mathcal{R})$ is a proper point-ray collection, then $P_I \subseteq \widehat{C}(\mathcal{P}, \mathcal{R})$.*

Proof. Otherwise there exists an inequality valid for $\mathcal{G}$ that cuts both $\bar{x}$ and a point of P_I. This contradicts the assumption that $(\mathcal{P}, \mathcal{R})$ is proper. ☐

The next two lemmas state inclusion properties for the sets of cuts obtainable from two different point-ray collections.

Lemma 2.7. *For $i \in \{1, 2\}$, let $\mathcal{P}^i$ denote a set of points on $\operatorname{bd} S$ and $\mathcal{R}^i$ a set of rays. If for every (α, β) such that α is feasible to CutRegion$(\beta, \mathcal{P}^2, \mathcal{R}^2)$ and $\alpha^\mathsf{T}\bar{x} < \beta$, α is also feasible to CutRegion$(\beta, \mathcal{P}^1, \mathcal{R}^1)$, then*

$$\widehat{C}(\mathcal{P}^1, \mathcal{R}^1) \subseteq \widehat{C}(\mathcal{P}^2, \mathcal{R}^2).$$

Proof. Let $\widehat{C}^i := \widehat{C}(\mathcal{P}^i, \mathcal{R}^i)$, $i \in \{1, 2\}$. Assume for the sake of contradiction that there exists an extreme ray r of $\widehat{C}^1$ that is not in $\widehat{C}^2$. If r does not intersect $\operatorname{bd} S$, define $q := r$; otherwise, let $q := r \cap \operatorname{bd} S$, and note that $q \notin \widehat{C}^2$. Then there exists an inequality $\widehat{\alpha}^\mathsf{T} x \geq \widehat{\beta}$ that separates $\bar{x}$ and q from $\mathcal{G}(\mathcal{P}^2, \mathcal{R}^2)$. This is a contradiction, as $\widehat{\alpha}$ is feasible to CutRegion$(\widehat{\beta}, \mathcal{P}^2, \mathcal{R}^2)$ and $\widehat{\alpha}^\mathsf{T}\bar{x} < \widehat{\beta}$, but it is not feasible to CutRegion$(\widehat{\beta}, \mathcal{P}^1, \mathcal{R}^1)$, since it violates the inequality corresponding to q. ☐

Lemma 2.8. *Let P^1 and P^2 be rational convex polyhedra such that $P^1 \subseteq P^2$, and $\bar{x} \in P^2$. Let $\mathcal{P}^i$ and $\mathcal{R}^i$ denote the intersection points and rays generated from the intersection of P^i with $\operatorname{bd} S$, for $i \in \{1, 2\}$. Then, for a given constant β, a vector α feasible to CutRegion$(\beta, \mathcal{P}^2, \mathcal{R}^2)$ and cutting $\bar{x}$ will also be feasible to CutRegion$(\beta, \mathcal{P}^1, \mathcal{R}^1)$.*

Proof. Since $P^1 \subseteq P^2$, we have $P^1 \cap \operatorname{bd} S \subseteq P^2 \cap \operatorname{bd} S \subseteq P^2 \setminus \operatorname{int} S$. By Lemma 2.4, $P^2 \setminus \operatorname{int} S \subseteq \{x : \alpha^\mathsf{T} x \geq \beta\}$, which completes the proof. ☐

Next, we remark on one subtlety of Algorithm 2.2, in step 2. Namely, whenever the hyperplane being activated intersects a ray of $\bar{C}$ outside of $\operatorname{int} S$, the algorithm skips that ray. In Proposition 2.9, we show this is valid because such intersections would lead to only *redundant* intersection points, in the sense that these points will never be part of a valid inequality that cuts away $\bar{x}$. We show the result only for points, as rays are never redundant. Recall that $\mathcal{K}$ denotes the skeleton of the polyhedron C, $\mathcal{K}'$ denotes the connected component of $\mathcal{K} \cap \operatorname{int} S$ that includes $\bar{x}$, and $\mathcal{K}''$ denotes the union of the other components of $\mathcal{K} \cap \operatorname{int} S$. Let $\mathcal{P}'$ and $\mathcal{P}''$ denote the intersection points between $\operatorname{bd} S$ and the edges of the closure of $\mathcal{K}'$ and $\mathcal{K}''$, respectively.

Proposition 2.9. *Any intersection points in $\mathcal{P}'' \setminus \mathcal{P}'$ are redundant.*

Proof. Let α be any feasible solution to $\mathrm{CutRegion}(\beta, \mathcal{P}', \mathcal{R})$ such that $\alpha^\mathsf{T}\bar{x} < \beta$. Suppose for the sake of contradiction that a point $w \in \mathcal{P}'' \setminus \mathcal{P}'$ violates the inequality. Since $w \in \mathcal{K}''$ and $\bar{x} \in \mathcal{K}'$, any path from $\bar{x}$ to w along the skeleton of C must intersect $\operatorname{bd} S$ at some point in $\mathcal{P}'$. Since C is convex and both $\bar{x}$ and w are cut, there must exist such a path entirely in $\{x : \alpha^\mathsf{T} x < \beta\}$, implying some intersection point in $\mathcal{P}'$ is cut, a contradiction. $\square$

We now use these results to prove Theorem 2.5.

Proof of Theorem 2.5. Let $(\mathcal{P}^*, \mathcal{R}^*)$ be the point-ray collection that Algorithm 2.2 outputs. Let $\widehat{C}^* := \widehat{C}(\mathcal{P}^*, \mathcal{R}^*)$. Observe that intersecting the edges of $\widehat{C}^*$ with $\operatorname{bd} S$ yields exactly the same point-ray collection $(\mathcal{P}^*, \mathcal{R}^*)$. Then, using Lemma 2.4 with $\widehat{C}^*$, it follows that all obtainable cuts will be valid for $\widehat{C}^* \setminus \operatorname{int} S$. To prove the theorem, it suffices to show the inclusion $P_I \subseteq \widehat{C}^*$, which we do next.

From Lemma 2.6, we have that $P_I \subseteq \widehat{C} := \widehat{C}(\mathcal{P}, \mathcal{R})$, where $(\mathcal{P}, \mathcal{R})$ is the proper point-ray collection given as an input to the algorithm. We need to show that the point-ray collection after activating H_h is proper. Let $\mathcal{P}^c$ and $\mathcal{R}^c$ denote the set of points and rays from $(\mathcal{P}, \mathcal{R})$ that violate H_h^+.

First we consider intersection points that lie on H_h. Let $P^1 := \widehat{C} \cap H_h$ and $P^2 := \bar{C} \cap H_h$, with $\mathcal{P}^i$ and $\mathcal{R}^i$ the corresponding intersection points and rays of P^i with $\operatorname{bd} S$, for $i \in \{1, 2\}$. Since $\mathcal{P}$ and $\mathcal{R}$ are contained in $\bar{C}$, $\widehat{C} \subseteq \bar{C}$. As a result, we can apply Lemma 2.8, which implies that every α (for a β such that $\alpha^\mathsf{T}\bar{x} < \beta$) that is feasible to $\mathrm{CutRegion}(\beta, \mathcal{P}^2, \mathcal{R}^2)$ will also be feasible to $\mathrm{CutRegion}(\beta, \mathcal{P}^1, \mathcal{R}^1)$.

Now we consider the entire halfspace H_h^+. The set of intersection points of $\widehat{C} \cap H_h^+$ with $\operatorname{bd} S$ is $\overline{\mathcal{P}} := \mathcal{P}^1 \cup \mathcal{P} \setminus \mathcal{P}^c$, and the set of rays is $\overline{\mathcal{R}} := \mathcal{R}^1 \cup \mathcal{R} \setminus \mathcal{R}^c$. In addition, we have that $\mathcal{P}^* = \mathcal{P}^2 \cup \mathcal{P} \setminus \mathcal{P}^c$, and $\mathcal{R}^* = \mathcal{R}^2 \cup \mathcal{R} \setminus \mathcal{R}^c$. Here we are assuming that all of the points in $\mathcal{P}^2$ belong to the same part of the skeleton of $\bar{C} \cap H_h^+$ that contains $\bar{x}$. This is without loss of generality as a result of Proposition 2.9.

Hence, for any β, suppose α is feasible to $\mathrm{CutRegion}(\beta, \mathcal{P}^*, \mathcal{R}^*)$ and satisfies $\alpha^\mathsf{T}\bar{x} < \beta$. Then α will also be feasible to $\mathrm{CutRegion}(\beta, \overline{\mathcal{P}}, \overline{\mathcal{R}})$. By Lemma 2.7, $\widehat{C}(\overline{\mathcal{P}}, \overline{\mathcal{R}}) \subseteq \widehat{C}^*$. Moreover, since $P_I \subseteq \widehat{C}$ and $P_I \subseteq H_h^+$, we have that $P_I \subseteq \widehat{C} \cap H_h^+$. The intersection points and rays obtained from intersecting $\widehat{C}(\overline{\mathcal{P}}, \overline{\mathcal{R}})$ with $\operatorname{bd} S$ are precisely $\overline{\mathcal{P}}$ and $\overline{\mathcal{R}}$. As a result, by Lemma 2.6, $P_I \subseteq \widehat{C}(\overline{\mathcal{P}}, \overline{\mathcal{R}})$. It follows that $P_I \subseteq \widehat{C}^*$, as desired. $\square$

Performing Algorithm 2.2 k_h times is dramatically more efficient than the $O(n^{k_h})$ complexity of full hyperplane activation. For every activation using Algorithm 2.2, letting $k_r := |\mathcal{R}_A|$, the hyperplane H_h can cut $O(k_r)$ rays to create $O(k_r^2)$ new intersection points. Therefore, if k_h hyperplanes are activated, the number of intersection points generated by partial activation is $O(k_r^2 k_h)$.

In the presence of rays in the initial point-ray collection, we next describe the role of the parameter $\mathcal{R}_A$ in PHA_1, which restricts the set of rays from $\bar{C}$ that can be cut by any activated hyperplane. Edges of C that do not intersect $\operatorname{bd} S$ can be viewed as intersection points infinitely far away, and Balas and Margot [31] show that this view suffices when extending $\mathrm{CutRegion}(\beta, \mathcal{P}, \mathcal{R})$ with only points to a valid system with both points and rays.

However, when generating GICs, rays play a fundamentally different role than points. In particular, when activating a new hyperplane cuts some $p \in \mathcal{P}_0$, the new intersection points that are created are either on the SIC or are deeper relative to p (see Theorem 3 in [31]). In contrast, any intersection points created as a result of cutting a ray from $\mathcal{R}_0$ will all lie on the SIC, as shown in Proposition 2.10.

Proposition 2.10. *Suppose $r \in \mathcal{R}_0$ is a ray that does not intersect $\operatorname{bd} S$, and H is an activated hyperplane that intersects r at $v := r \cap H$. Denote by Y the set of new rays on H that originate from v. Then the subset of rays Y' in Y that intersect $\operatorname{bd} S$ create intersections points on the hyperplane $\alpha_0^\mathsf{T} x = \beta_0$, where $\alpha_0^\mathsf{T} x \geq \beta_0$ is the SIC from $\bar{C}$ and S. The rays r' in $Y \setminus Y'$ satisfy $\alpha_0^\mathsf{T} r' = 0$.*

Proof. Let $\mathcal{R}_0^c \subseteq \mathcal{R}_0$ denote the set of initial rays cut by H before $\operatorname{bd} S$. When ray r intersects H, it creates $n - 1 - |\mathcal{R}_0^c|$ new rays emanating from v, which we denote by Y. There are $n - 1$ hyperplanes that define r. Any ray $r' \in Y$ lies on $n - 2$ of these hyperplanes, as well as on H. There exists a ray $\widehat{r} \in \mathcal{R}_0 \setminus \mathcal{R}_0^c$ that also lies on these $n - 2$ facets, and r' can be written as a nontrivial conic combination of r and $\widehat{r}$. Let F denote the two-dimensional face of $\bar{C}$ defined by all conic combinations of r and $\widehat{r}$.

Suppose $\widehat{r}$ intersects $\operatorname{bd} S$ and let $\widehat{p} := \widehat{r} \cap \operatorname{bd} S$. By the remark above, the half-line $L := \widehat{p} + \theta r$, $\theta \geq 0$ is contained in $\{x : \alpha_0^\mathsf{T} x = \beta_0\} \cap \operatorname{bd} S$, since $r \in \mathcal{R}_0$ does not intersect $\operatorname{bd} S$. Further, L is also contained in F, which is parallel to r, $\widehat{r}$, and r'. Since r' is a nontrivial conic combination of $\widehat{r}$ and r, it intersects $\operatorname{bd} S$. The resulting intersection point is on L, which lies on the SIC.

Now suppose $\widehat{r}$ does not intersect $\operatorname{bd} S$. Then $\alpha_0^\mathsf{T} r = \alpha_0^\mathsf{T} \widehat{r} = 0$. Since r' is a nontrivial conic combination of r and $\widehat{r}$, it too satisfies $\alpha_0^\mathsf{T} r' = 0$. $\square$

The consequence of Proposition 2.10 is that activating hyperplanes on rays of $\bar{C}$ that do not intersect $\operatorname{bd} S$ will create weak points, as these points actually lie on the SIC, not deeper as desired. Thus, the input $\mathcal{R}_A$ to Algorithm 2.2 will, in our implementation, always be chosen from the set of rays of $\bar{C}$ that intersect $\operatorname{bd} S$, i.e., from $\bar{\mathcal{R}} \setminus \mathcal{R}_0$. Choosing $\mathcal{R}_A$ arbitrarily may in general lead to invalid cuts; the validity of our choice follows from the results in Section 2.4. The full details of how we implement PHA$_1$ to generate cuts are given in Algorithm 2.3.

2.4 PHA$_1$ with tilted hyperplanes

An outcome of proving the validity of PHA$_1$ is that the hyperplane given as an input to Algorithm 2.2 can be an arbitrary valid hyperplane for P, not necessarily one from the hyperplane description of P. One application of this insight is to choose hyperplanes that only cut rays of $\bar{C}$ that intersect $\operatorname{bd} S$, to avoid the weak intersection points that would otherwise be created as shown in Proposition 2.10. However, it is not practical to search for arbitrary valid hyperplanes that satisfy specific desired properties such as which particular rays of $\bar{C}$ are or are not cut. Instead, we propose to activate *tilted* versions of hyperplanes defining P, and we show this can be done using information from the original (not tilted) hyperplane without a need to explicitly calculate the tilted hyperplane; that is, tilting does

Algorithm 2.3 Generalized Intersections Cuts by PHA$_1$

```
Input: Convex rational polyhedron $P$ defined by a set of hyperplanes $\mathcal{H}$; set $\mathcal{I} \subseteq [n]$
    denoting the integer variables, objective coefficient vector $c$; classes of objectives $\mathcal{O}$;
    number non-tilted hyperplanes $k_h$; fractionality threshold $\epsilon_{\text{frac}}$.
 1: function PHA$_1$CutGenerator$(P, \mathcal{I}, c, \mathcal{O}, k_h, \epsilon_{\text{frac}})$
 2:     $\bar{x} \leftarrow$ a solution from $\arg\min\{c^\mathsf{T} x : x \in P\}$.
 3:     $\sigma \leftarrow \{j \in \mathcal{I} : \max\{\bar{x}_j - \lfloor \bar{x}_j \rfloor, \lceil \bar{x}_j \rceil - \bar{x}_j\} \geq \epsilon_{\text{frac}}\}$.
 4:     $\mathcal{H}_A \leftarrow \emptyset$; $\mathcal{C} \leftarrow \emptyset$.
 5:     for $k \in \sigma$ do
 6:         $\mathcal{R}_k^{\|} \leftarrow \{r \in \bar{\mathcal{R}} : \operatorname{dist}(\operatorname{bd} S_k, r) = \infty\}$; $\mathcal{R}_k \leftarrow \mathcal{R}_k^{\|}$; $\mathcal{P}_k \leftarrow \{r \cap \operatorname{bd} S_k : r \in \bar{\mathcal{R}} \setminus \mathcal{R}_k\}$.
 7:     for $h \in \{1, \ldots, k_h\}$ do
 8:         for $k \in \sigma$ do
 9:             Choose a hyperplane $H$ to activate.
10:             $(\mathcal{P}_k, \mathcal{R}_k) \leftarrow$ PHA$_1(P, S_k, H, \bar{\mathcal{R}} \setminus \mathcal{R}_k^{\|}, (\mathcal{P}_k, \mathcal{R}_k))$.
11:         Add cuts to $\mathcal{C}$ by solving (2.2) with objective types $\mathcal{O}$, ensuring $\bar{x}$ is cut.
12:     return $\mathcal{C}$.
13: end function
```

not lead to any additional computational burden. A byproduct of the tilting theory is that choosing $\mathcal{R}_A$ as in Algorithm 2.3 is valid.

Algorithm 2.4 is an alternative, but not mutually exclusive, approach to Algorithm 2.3 for using PHA$_1$. Namely, we provide a way to use tilting to reduce the size of the point-ray collection when the number of rays of $\bar{C}$ that are cut is large, which is desirable when seeking cuts stronger than the SIC. When cutting k_r rays and activating k_h hyperplanes, the algorithm creates $O(k_r k_h^2)$ points and rays, which has a clear advantage to the $O(k_r^2 k_h)$ points and rays that are created from Algorithm 2.3 when k_r is large. In our implementation in Section 2.7, we compare the two ideas when used individually and together.

We now define a tilted hyperplane. Let H be a hyperplane defining P that is not tight at $\bar{x}$. The degenerate case is handled separately, in Appendix C.1. Any n affinely independent points on H define the hyperplane; to define a tilted version of H, we will only need to change some of these points. We use the intersection of H with the n affinely independent rays of $\bar{C}$ (possibly in their negative directions) to define H.

Suppose $\bar{\mathcal{N}} = \{1, \ldots, n\}$ to remove indexing issues. For an arbitrary $d \in \mathbb{R}^n \cup \{\infty\}$, for $j \in \bar{\mathcal{N}}$ and the corresponding $r^j \in \bar{\mathcal{R}}$, if $d_j \notin \{0, \infty\}$, define $v(d_j) := \bar{x} + d_j r^j$, and otherwise, set $v(d_j) := r^j$. Let $\mathcal{V}(d) := \{v(d_j) : j \in \bar{\mathcal{N}}\}$.

Let d^H be a vector of length n with entry j equal to $\min\{\operatorname{dist}(H, r^j), -\operatorname{dist}(H, -r^j)\}$ if r^j or $-r^j$ intersects H, and $d_j^H = 0$ otherwise. Each of the vertices in $\mathcal{V}(d^H)$ is affinely independent from the rest because the rays $\bar{\mathcal{R}}$ are affinely independent. Furthermore, all the rays r^j such that $d_j^H = 0$ are affinely independent, and none of the vertices created lie on these rays. Thus, the the set $\mathcal{V}(d^H)$ of n vertices and rays uniquely define the hyperplane H.

A *tilting* of H is defined via a vector $\delta \in \mathbb{R}^n \cup \{\infty\}$, so that the tilted hyperplane $\widetilde{H}$ is the unique hyperplane through the vertices and rays given in $\mathcal{V}(d^H + \delta)$. While tilting hyperplanes can be used in many ways in conjunction with PHA$_1$, arbitrary values of δ would lead to invalid cuts. Indeed, it is clear that δ should be chosen in a way that ensures $\widetilde{H}$ is valid for P. However, in Definition 2.11, we further restrict the range of values for δ based on our motivation of selectively cutting only certain rays of $\bar{C}$ instead of all of the rays. These tighter limits essentially restrict the tilted hyperplane to intersect each ray $r \in \bar{\mathcal{R}}$ either in the original vertex $H \cap r$, or in the initial intersection point $r \cap \operatorname{bd} S$.

Definition 2.11. *Let $\mathcal{R}^c(\bar{C}) \subseteq \bar{\mathcal{R}}$ denote the rays of $\bar{C}$ that have been cut by a set of previously activated hyperplanes. Let H be a hyperplane defining P that is not tight at $\bar{x}$. Then $\delta \in \mathbb{R}^n \cup \{\infty\}$ defines a* targeted tilting *of H (with respect to S) if $\delta_j \in \{0, \operatorname{dist}(\operatorname{bd} S, r^j) - d^H_j\}$ for all $j \in \{1, \ldots, n\}$ such that $0 < d^H_j < \operatorname{dist}(\operatorname{bd} S, r^j)$ and $r^j \notin \mathcal{R}^c(\bar{C})$, and $\delta_j = 0$ otherwise.*

The motivation for targeted tilting comes not only from Proposition 2.10, to avoid cutting the rays in $\mathcal{R}_0$, but also from our desire to find GICs strictly stronger than the SIC from the same P_I-free convex set S. This may require cutting many rays of $\bar{C}$. However, using Algorithm 2.3 for this purpose could involve unnecessarily cutting some rays of $\bar{C}$ multiple times. This will happen when we have already cut a certain ray of $\bar{C}$ but a hyperplane being subsequently activated intersects it again; not only might this effort be wasted, but also it may create redundant or weak intersection points. As an alternative, we present the targeted tilting algorithm, Algorithm 2.4, which is tailored to the task of cutting the initial intersection points and rays more efficiently.

Algorithm 2.4 Generalized Intersection Cuts by PHA$_1$ and Targeted Tilting

```
Input: Polyhedron $P$ defined by a set of hyperplanes $\mathcal{H}$; a vertex $\bar{x}$ of $P$; indices of fractional
    integer variables $\sigma$; hyperplane selection criterion $\mathcal{SC}$; classes of objectives $\mathcal{O}$; set $\mathcal{R}_A \subseteq \bar{\mathcal{R}}$
    for the rays of $\bar{C}$ to cut.
 1: function PHA$_1$CutGeneratorWithTargetedTilting($P, \bar{x}, \sigma, \mathcal{SC}, \mathcal{O}, \mathcal{R}_A$)
 2:     for $k \in \sigma$ do
 3:         $\mathcal{R}_k \leftarrow \{r \in \bar{\mathcal{R}} : \operatorname{dist}(\operatorname{bd} S_k, r) = \infty\}$; $\mathcal{P}_k \leftarrow \{r \cap \operatorname{bd} S_k : r \in \bar{\mathcal{R}} \setminus \mathcal{R}_k\}$.
 4:     for $j \in \{1, \ldots, n\}$ do                    ▷ Assume sorted by decreasing reduced cost
 5:         If $r^j \notin \mathcal{R}_A$, skip this ray and return to step 4.
 6:         for $k \in \sigma$ do
 7:             $\mathcal{H}^j \leftarrow \{H \in \mathcal{H} \setminus \bar{\mathcal{H}} : \operatorname{dist}(H, r^j) < \operatorname{dist}(\operatorname{bd} S_k, r^j)\}$, $\delta \leftarrow \{0\}^n$.
 8:             if $\mathcal{H}^j \neq \emptyset$ then
 9:                 Select a hyperplane $H_j \in \mathcal{H}^j$ according to selection criterion $\mathcal{SC}$.
10:                 for all $i \in \{j+1, \ldots, n\}$ do
11:                     $\delta_i \leftarrow \max\{0, \operatorname{dist}(\operatorname{bd} S_k, r^i) - \operatorname{dist}(H_j, r^i)\}$.
12:                 $(\mathcal{P}_k, \mathcal{R}_k) \leftarrow$ PHA$_1(P, S_k, H_j, \{r^1, \ldots, r^j\} \cap \mathcal{R}_A, (\mathcal{P}_k, \mathcal{R}_k))$.
13:         if $\log(j) \in \mathbb{Z}$ or $j = n$ then
14:             Add cuts to $\mathcal{C}$ by solving (2.2) with objective types $\mathcal{O}$, ensuring $\bar{x}$ is cut.
15: end function
```

Algorithm 2.4 focuses on cutting one ray of $\bar{C}$ at a time. The hyperplane chosen for each ray is tilted in step 11 in a way that avoids cutting all other rays, other than the ones that have been previously cut, as required in Definition 2.11. However, this requirement creates a nonmonotonicity when performing targeted tilting, in the sense that cutting more rays of $\bar{C}$ could actually lead to generating weaker cuts. This is due to our observation that cutting a ray multiple times may lead to the addition of weak intersection points to the collection. We counteract this in step 14 of Algorithm 2.4, by performing cut generation before all hyperplanes have been chosen and activated. To reduce computational effort, this step is only performed $\lceil \log(n) \rceil$ times.

Although targeted tilting may reduce the number of weak intersection points that result from cutting the same rays of $\bar{C}$ repeatedly, it may also miss improving the point-ray collection in certain directions. This creates an inherent trade-off when deciding whether to use PHA$_1$ as part of Algorithm 2.3 or Algorithm 2.4. We discuss this as part of our numerical study in Section 2.7.

Using $\mathcal{V}(d^H + \delta)$, one can easily compute the closed-form expression for the tilted hyperplane $\widetilde{H}$, which can then be used in PHA$_1$ to calculate the new point-ray collection. However, the next theorem shows that when a hyperplane obtained by targeted tilting is being activated, the formula for the tilted hyperplane is not required to be computed. The effect is that throughout PHA$_1$, all calculations can be performed using H and not $\widetilde{H}$.

Theorem 2.12. *Let H be a hyperplane defining P and $\mathcal{R}^c(\bar{C}) \subseteq \bar{\mathcal{R}}$ the set of rays previously cut, $\widetilde{H}$ be a hyperplane obtained via a valid targeted tilting δ of H, and $\mathcal{R}_A := \{r^j \in \bar{\mathcal{R}} : 0 < d^H_j + \delta_j < \operatorname{dist}(\operatorname{bd} S, r^j)\}$. Given a proper point-ray collection $(\mathcal{P}, \mathcal{R})$, the point-ray collection returned by PHA$_1(P, S, H, \mathcal{R}_A, (\mathcal{P}, \mathcal{R}))$ is proper.*

Proof. We show that PHA$_1(P, S, H, \mathcal{R}_A, (\mathcal{P}, \mathcal{R}))$ returns an identical point-ray collection to PHA$_1(P, S, \widetilde{H}, \bar{\mathcal{R}}, (\mathcal{P}, \mathcal{R}))$, which we have already proven is proper in Theorem 2.5. Let $r^j \in \bar{\mathcal{R}}$. We analyze, case-by-case, the steps of Algorithm 2.2 when activating $\widetilde{H}$ over all the rays $\bar{\mathcal{R}}$ versus activating H over the rays $\mathcal{R}_A$.

Case 1: $d^H_j + \delta_j \geq \operatorname{dist}(\operatorname{bd} S, r^j)$. In this case, r^j is not selected in step 2 of Algorithm 2.2. This is because, when activating H, r^j is not in $\mathcal{R}_A$, and when activating $\widetilde{H}$, $\operatorname{dist}(\widetilde{H}, r^j) \geq \operatorname{dist}(\operatorname{bd} S, r^j)$. Note that if $\delta_j > 0$, then we are in this case, so we assume $\delta_j = 0$, i.e., $d^H_j = d^{\widetilde{H}}_j$, for the remainder of the proof.

Case 2: $d^H_j \leq 0$. The ray r^j is again not considered in step 2 of the algorithm, this time because $\operatorname{dist}(H, r^j) = \operatorname{dist}(\widetilde{H}, r^j) = \infty \geq \operatorname{dist}(\operatorname{bd} S, r^j)$.

Case 3: $0 < d^H_j < \operatorname{dist}(\operatorname{bd} S, r^j)$. In this case, the ray is processed in step 2 for both H (it belongs to $\mathcal{R}_A$) and $\widetilde{H}$. As we noted, $v^j := r^j \cap \widetilde{H}$ is the same as the intersection $r^j \cap H$. It will be useful in the subsequent discussion to compare the edges that originate at v^j for $\bar{C} \cap \widetilde{H}$ and $\bar{C} \cap H$. Other than the edge from v^j to $\bar{x}$, each such edge is directed towards a ray $r^k \in \bar{\mathcal{R}} \setminus \{r^j\}$. Let e_k be the edge of $\bar{C} \cap H$ in the direction of r^k, and let $\widetilde{e}_k$ be the edge of $\bar{C} \cap \widetilde{H}$ in that same direction. When $\delta_k = 0$, we have that $e_k = \widetilde{e}_k$, in which case the steps of Algorithm 2.2 relating to this edge will be same regardless of whether $\widetilde{H}$ or H is being activated. Thus, we focus on the case that $\delta_k > 0$, i.e., when $0 < d^H_k < \operatorname{dist}(\operatorname{bd} S, r^k)$ and $d^H_k + \delta_k = \operatorname{dist}(\operatorname{bd} S, r^k)$. Note that this means e_k intersects r^k somewhere in $\operatorname{int} S$.

We now check that the same points and rays are removed in step 3 for both procedures. Consider a point or ray, call it p, that is removed in this step by the activation of $\widetilde{H}$. Let e

be the edge that had resulted in the addition of p to $(\mathcal{P}, \mathcal{R})$. It must be that e originates at a vertex that lies between $\bar{x}$ and v^j, and e does not intersect r^k anywhere in $\operatorname{int} S$. Moreover, e crosses $\widetilde{e}_k$ for some edge $\widetilde{e}_k$ emanating from v^j. Clearly this remains the case when the corresponding edge e_k coincides with $\widetilde{e}_k$. Else, as we observed, $e_k \cap r^k$ occurs in $\operatorname{int} S$. Meanwhile, $\widetilde{e}_k$ does not intersect r^k in $\operatorname{int} S$. As e_k and $\widetilde{e}_k$ both originate at v^j, it follows that in order to cross $\widetilde{e}_k$, e must first cross e_k, which implies that p will also be removed in step 3 when activating H, as desired. This argument is illustrated in Figure 2.1. Conversely, if a point is not removed by the activation of $\widetilde{H}$, it is easy to see that the point will not violate H either.

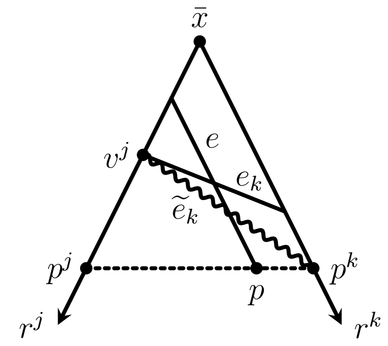


Figure 2.1: Illustration of case 3 of the proof of Theorem 2.12. When e_k and $\widetilde{e}_k$ do not coincide, it must be the case that e_k intersects r^k in the interior of S and that the point p will be cut by both H and $\widetilde{H}$.

Finally, we show that the same points and rays are added in the for loop starting at step 4. These steps are identical when e_k and $\widetilde{e}_k$ coincide, so we again suppose that $e_k \neq \widetilde{e}_k$. The activation of $\widetilde{H}$ would lead to adding the initial intersection point or ray $r^k \cap \operatorname{bd} S$. However, this point or ray already exists in $(\mathcal{P}, \mathcal{R})$, so it suffices to show that activating H will neither remove this existing point or ray, nor add an extraneous point or ray. The first part is a consequence of the fact that $r^k \notin \mathcal{R}_A$. For the second part, that $e_k \cap r^k$ occurs prior to $\operatorname{bd} S$ means that the condition in step 5 holds, so this edge is skipped and no extraneous points or rays are added to the point-ray collection. □

One effect of Theorem 2.12 is that it is valid to simply ignore all intersections of hyperplanes with rays in $\mathcal{R}_0$, as is done in Algorithm 2.3. To see this, for any hyperplane H valid for P, we can simply set $\delta_j = \infty$ when $d_j^H > 0$ and $\delta_j = 0$ when $d_j^H \leq 0$, resulting in the tilted hyperplane $\widetilde{H}$ not intersecting r^j before $\operatorname{bd} S$. The activation performed in step 10 of Algorithm 2.3 is valid because the rays in $\mathcal{R}_0$ are always ignored, so that the conditions of Theorem 2.12 apply. Similarly, the theorem applies for step 12 of Algorithm 2.4.

An arbitrary sequence of valid tilted hyperplanes would not enable us to use this implicit calculation. Two subtle conditions appearing in Definition 2.11 are of particular importance for Theorem 2.12 to hold. First, we require that $\delta_j = 0$ for all rays of $\bar{C}$ that have previously been cut. Second, when δ_j is nonzero, we force it to take the exact value $\operatorname{dist}(\operatorname{bd} S, r^j)$. Appendix C.2 contains an example showing that relaxing either of these conditions may lead to invalid cuts.

2.5 Implementation choices for PHA_1

Algorithms 2.3 and 2.4 both involve steps in which a hyperplane is selected and then, after the points and rays have been collected, a set of objective directions is used for generating GICs from the resulting (2.2). In this section, we discuss the choices we make for these steps and provide theoretical motivation for the decisions.

2.5.1 Choosing hyperplanes to activate

Hyperplane activation in Algorithm 2.4

We first state the three criteria we consider for choosing hyperplanes to activate in step 9 of Algorithm 2.4, i.e., as the parameter $\mathcal{SC}$. For an index $k \in \mathcal{I}$, we consider optimizing over the S_k-closure, i.e., $\operatorname{conv}(P \setminus \operatorname{int} S_k)$.

(H1) Choose the hyperplane that first intersects some ray $r \in \bar{\mathcal{R}} \setminus \mathcal{R}_0$ before $\operatorname{bd} S_k$. This corresponds to pivoting to the nearest neighbor from $\bar{x}$ along the ray r, which would be the same hyperplane selected in the procedure from [31].

(H2) Choose the hyperplane that yields a set of points with highest average depth. Here, *depth* is calculated as the Euclidean distance to the SIC (using the same split set). (H2) seeks a set of points that are far from the cut we are trying to improve upon. The idea is that the resulting cuts will then be deeper as well.

(H3) Choose the hyperplane creating the most final intersection points. A *final* intersection point is defined as follows.

Definition 2.13. *Suppose $(\mathcal{P}_k, \mathcal{R}_k)$ is a proper point-ray collection. An intersection point in $\mathcal{P}_k$ or a ray in $\mathcal{R}_k$ is* final *(with respect to S_k) if it belongs to P and $\operatorname{bd} S_k$, meaning it cannot be cut away by any valid hyperplane activations.*

We denote the set of final intersection points by $\mathcal{P}_k^F$ and final rays by $\mathcal{R}_k^F$.

Proposition 2.14. *Suppose $C = P$. Then, the intersection points in $\mathcal{P}_k^F$ define vertices of the S_k-closure and the rays $\mathcal{R}_k^F$ define extreme rays of S_k-closure.*

Proof. The S_k-closure is defined as $P_k = \operatorname{conv}(P \setminus \operatorname{int} S_k)$. Therefore, the points $p \in \mathcal{P}^F$ are in the S_k-closure since they belong to $P \cap \operatorname{bd} S_k$. Furthermore, the rays in $\mathcal{R}^F$ are in the S_k-closure since they are extreme rays of P that do not intersect $\operatorname{bd} S_k$.

Consider an arbitrary side of the split disjunction, $S_k^1 := \{x : x_k = \lfloor \bar{x}_k \rfloor\}$. Suppose that a vertex $p = r \cap S_k^1$ in $\mathcal{P}_k$ is not a vertex of the S_k-closure. Then it can be written as a convex combination of other vertices of $P \cap S_k^1$ that are also intersection points created from edges of P. This implies that r, an edge of P, can be written as a conic combination of edges in P that intersect S_k^1, which is a contradiction. Showing that rays in $\mathcal{R}_k$ are extreme rays of S_k-closure follows similar reasoning. □

Thus, (H3) targets a set of points that lead to facet-defining inequalities for the set $\operatorname{conv}(P \setminus \operatorname{int} S)$. It turns out to be useful to distinguish between intersection points that are in P and those that are not.

Hyperplane activation in Algorithm 2.3

Hyperplanes in step 9 of Algorithm 2.3 are chosen by combining the above options into one rule as follows. For each hyperplane H that is a candidate for activation, we compute the set of intersection points that would result from activating H. We calculate the number of new final intersection points created, as well as the average depth of the points with respect to the SIC. We choose the best hyperplane H^* based on (H3) and use (H2) as a tie-breaking rule, as several hyperplanes may all create the most final intersection points. We then recompute the number of new final intersection points and average depth of the points for the remaining hyperplanes (after H^* is activated) and repeat the process, until k_h hyperplanes are selected.

2.5.2 Choosing objective functions

Even if we know that strong cuts exist from a given point-ray collection, it remains to generate these GICs using (2.2). To do this, we need to appropriately choose objective function coefficient vectors. We consider the following objective directions:

(R) Ray directions of $\bar{C}$ (i.e., the initial intersection points)

(V) Vertices that are generated from hyperplane activations

(T) Intersection points obtained for S_k (the *tight point* heuristic)

(S) Intersection points from other splits

The first set of objectives, (R), is incentivized by the observation that obtaining GICs that strictly dominate the SIC from the same P_I-free convex set requires cutting many of the initial intersection points. This observation is made concrete in the Theorem 2.18 in Appendix 2.6; we show that finding a strictly dominating cut requires reducing the dimension of the convex hull of the initial intersection points and rays. Another objective typically used for a cut-generating linear program is to minimize $\bar{x}^\mathsf{T}\alpha$, to obtain the most violated cut with respect to $\bar{x}$. We generalize this with (V), by optimizing to find the most violated cuts with respect to each of the vertices created on $\bar{C}$ by hyperplane activations.

The third set of directions, (T), is referred to as the tight point heuristic as we are simply trying to find a cut tight on each of the rows of (2.2) corresponding to points. In other words, we minimize $\alpha^\mathsf{T}p$ for every $p \in \mathcal{P}$ from the current split. Obviously, we will not be able to cut away p; the purpose of this objective function is to place a cut as close to the chosen intersection point as possible. The fourth set, (S), capitalizes on the fact that multiple split sets are considered simultaneously in practice, so that we can share information across the sets to obtain stronger cuts.

The objective directions (T) and (S) switch the typical perspective used for linear programs that generate cuts. Instead of finding a cut with maximal violation, the cuts we obtain from these objectives aim to approximate the convex hull of intersection points and rays from every direction, thereby obtaining as many of the facets of that convex hull as possible and a set of cuts with a wider diversity of angles, which is a desirable computational quality.

Next, we discuss the theoretical motivation for (T). Consider optimizing over the S_k-closure: $\min\{c^\mathsf{T}x : x \in \operatorname{conv}(P \setminus \operatorname{int} S_k)\}$. We show that intersection points can be used to find an optimal solution to this problem without using (2.2), and we address how the heuristic (T) aims to obtain the bound implied by this optimal solution.

Theorem 2.15 shows what bounds can be computed on the optimal value over the S_k-closure using the point-ray collection when $P \subseteq C$, i.e., not all hyperplanes may have been activated. We define $\overline{p}^k \in \arg\min_{p \in \mathcal{P}_k^F} c^\mathsf{T} p$ (when $\mathcal{P}_k^F$ is nonempty) and $\underline{p}^k \in \arg\min_{p \in \mathcal{P}_k} c^\mathsf{T} p$. Let $\overline{z} = c^\mathsf{T} \overline{p}^k$ (defined to be $+\infty$ when $\mathcal{P}_k^F$ is empty) and $\underline{z} = c^\mathsf{T} \underline{p}^k$.

Theorem 2.15. *Suppose $\mathcal{P}_k^F$ is nonempty. If $\underline{z} < \overline{z}$, then $\underline{z}$ is a lower bound and $\overline{z}$ is an upper bound on the minimum over the S_k-closure. Otherwise, $\overline{z} \leq \underline{z}$ and $\overline{z}$ is the minimum over the S_k-closure.*

Proof. By Proposition 2.14, $\overline{p}^k$ is in the S_k-closure. Therefore, $\overline{z}$ always provides an upper bound on $c^\mathsf{T} p^*$, where p^* is a minimum over the S_k-closure.

When $\underline{z} < \overline{z}$, $\underline{z} = \min\{c^\mathsf{T} x : x \in C \cap \operatorname{bd} S_k\}$ provides a valid lower bound on $c^\mathsf{T} p^*$, since $P \cap \operatorname{bd} S_k \subseteq C \cap \operatorname{bd} S_k$. On the other hand, when $\overline{z} \leq \underline{z}$, we have $\overline{z} = \min\{c^\mathsf{T} x : x \in C \cap \operatorname{bd} S_k\}$, which implies that $\overline{z}$ provides a lower bound on $c^\mathsf{T} p^*$. Therefore, it must be a minimum over the S_k-closure. □

Corollary 2.16 follows immediately for the special case when $C = P$.

Corollary 2.16. *If $C = P$, then $\overline{p}^k$ is an optimal solution over the S_k-closure.*

Thus, $\underline{p}^k$ is readily available from $\mathcal{P}_k$ and implies a lower bound on the value of the optimal solution over the S_k-closure. The same bound $\underline{z}$ implied by $\underline{p}^k$ can be obtained through GICs, but it may require many cuts generated from (2.2). We may be able to obtain one inequality tight at $\underline{p}^k$ by using $\underline{p}^k$ itself as the objective coefficient vector, since $\alpha^\mathsf{T} \underline{p}^k \geq \beta$ for all (α, β) feasible to (2.1), and there will exist a facet-defining inequality $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ for $\mathcal{G}$ satisfying $\bar{\alpha}^\mathsf{T} \underline{p}^k = \bar{\beta}$. For validity of this inequality to be guaranteed, we must additionally verify that the inequality cuts a point $v \in \mathcal{K}'$. However, even if this is satisfied, it is unlikely that the one cut will imply the bound $c^\mathsf{T} x \geq \underline{z}$ on the objective value. In the worst case, n facets of $\mathcal{G}$ that are tight at $\underline{p}^k$ may be required to obtain this bound via cuts. To get the other facets tight at $\underline{p}^k$, we can use points in $\mathcal{P}_k$ that lie close to $\underline{p}^k$. This is precisely what we do in approach (T), though we do not only use $\underline{p}^k$ and points in its vicinity, but also other points from $\mathcal{P}_k$ to encourage diversity of the cut collection.

Finally, we give more details and motivation for the objective directions (S). When using a single split set S_k, no intersection point generated from that split can be cut by any inequality generated through (2.2) from the point-ray collection for S_k. However, different splits (more generally, different P_I-free convex sets) give rise to different point-ray collections, and the buildup of intersection points and rays can and should be done in parallel for several splits. The first reason for this is computational efficiency: one can intersect an edge with the boundaries of more than one split set and store these intersection points or rays separately. The second reason is that, when using multiple split disjunctions, an intersection point on the boundary of one split set may lie in the interior of another and hence can be cut away by a facet of the point-ray collection from this second split. With this in mind, let σ be the indices of a set of integer variables that are fractional at $\bar{x}$, i.e., $\sigma \subseteq \{j \in \mathcal{I} : \bar{x}_j \notin \mathbb{Z}\}$. For every intersection point p generated from intersecting an edge of C with the boundary of some split disjunction, we follow the edge to find the *last* split disjunction S_k, $k \in \sigma$, that this edge intersects. If p lies in the interior of S_k, then we will use p as an objective for (2.2) with feasible region determined by the point-ray collection from S_k.

2.6 Theoretical results

2.6.1 Existence of strictly dominating generalized intersection cuts

In this section, we provide some theoretical motivation for Algorithm 2.4 by giving necessary and sufficient conditions for the existence of a GIC that strictly dominates the SIC after activation of a single hyperplane.

Definition 2.17 ([31]). *Consider two inequalities that are valid for P_I but not necessarily P. Inequality 2 dominates 1 on P if for every $x \in P$, the fact that x satisfies Inequality 2 implies that x satisfies Inequality 1. Inequality 2 strictly dominates 1 if, in addition, there exists $x \in P$ such that x violates Inequality 2 but satisfies Inequality 1.*

The theorem proved in this section strengthens Theorem 5 in Balas and Margot [31] for the case when S is a split disjunction. Theorem 5 of the aforementioned paper gives sufficient conditions for a GIC to strictly dominate the SIC, given that dominance holds. We show that this condition is also necessary for strict dominance when S is a split disjunction. For ease of exposition, Theorem 2.18 assumes that all rays of C intersect $\operatorname{bd} S$ because Proposition 2.10 shows that intersecting rays cannot lead to deeper points.

Suppose $S = \{x : 0 \le x_k \le 1\}$ is a split disjunction on a variable x_k. Let $S^1 := \{x : x_k = 0\}$, and $S^2 := \{x : x_k = 1\}$. We partition the intersection point set $\mathcal{P}$ into $\mathcal{P}^1$ and $\mathcal{P}^2$, where $\mathcal{P}^1 := \mathcal{P} \cap S^1$, and $\mathcal{P}^2 := \mathcal{P} \cap S^2$. Recall that $\mathcal{P}_0$ and $\mathcal{R}_0$ are the points and rays obtained from intersecting $\bar{C}$ with $\operatorname{bd} S$. We also partition $\mathcal{P}_0$ into $\mathcal{P}_0^1 := \mathcal{P}_0 \cap S^1$ and $\mathcal{P}_0^2 := \mathcal{P}_0 \cap S^2$. Intuitively, the theorem shows that a strictly dominating cut with respect to the SIC must reduce the dimension of $\operatorname{conv}(\mathcal{P}_0^1)$ or $\operatorname{conv}(\mathcal{P}_0^2)$.

Theorem 2.18. *Suppose that $\mathcal{R}_0 = \emptyset$, $\mathcal{P}_0^1 \neq \emptyset$ and $\mathcal{P}_0^2 \neq \emptyset$, and $(\mathcal{P}, \mathcal{R})$ is a proper point-ray collection obtained from activating a single hyperplane H valid for P. There exists a basic feasible solution to (2.1) corresponding to a cut strictly dominating $\alpha_0^\mathsf{T} x \ge \beta_0$ if and only if $\operatorname{relint}(H^+) \cap \mathcal{P}_0^t = \emptyset$ and $H^- \cap \mathcal{P}_0^t \neq \emptyset$, for at least one side t of the split disjunction, $t \in \{0, 1\}$.*

Proof. For the "if" direction of the proof, suppose without loss of generality that $\operatorname{relint}(H^+) \cap \mathcal{P}_0^1 = \emptyset$ and $H^- \cap \mathcal{P}_0^1 \neq \emptyset$. Any point in $\mathcal{P}^1$ lying on the SIC is in $\operatorname{conv}(\mathcal{P}_0^1)$. Because $\operatorname{relint}(H^+) \cap \mathcal{P}_0^1 = \emptyset$, it holds that $(\mathcal{P}^1 \setminus \mathcal{P}_0^1) \cap \operatorname{conv}(\mathcal{P}_0^1) = \emptyset$. This implies that any point p in $\mathcal{P}^0 \setminus \mathcal{P}_0^0$ satisfies $\alpha_0^\mathsf{T} p > \beta_0$. Recall that $|\mathcal{P}_0^1| + |\mathcal{P}_0^2| = n$. Since some point of $\mathcal{P}_0^1$ lies in H^-, $|\mathcal{P}_0^1 \cap H^+| \le |\mathcal{P}_0^1| - 1$. It follows that at most $n-1$ intersection points from $\mathcal{P}_0$ remain in $\mathcal{P}$. This added degree of freedom and the aforementioned depth of points in $\mathcal{P}^1 \setminus \mathcal{P}_0^1$ allows the SIC to be tilted to obtain a GIC that strictly dominates the SIC. The "only if" direction follows from Theorem 5 in Balas and Margot [31]. $\square$

The above result shows that any single hyperplane is unlikely to directly lead to a strictly dominating cut. Instead of looking for one such hyperplane, in our implementation we focus on activating a set of hyperplanes that together cut away large parts of $\operatorname{conv}(\mathcal{P}_0^1)$ and $\operatorname{conv}(\mathcal{P}_0^2)$. We do this by targeting each of the intersection points in $\mathcal{P}_0$ one at a time in step 4 of Algorithm 2.4.

Although strict dominance is difficult to attain, our next result shows that activating hyperplanes is monotonic in the sense that the lower bound implied by the point-ray collection can only be improved by activating hyperplanes. This complements Theorem 3 of Balas and Margot [31], in which it is shown that activating hyperplanes increases the depth of points with respect to the SIC. This leaves open the question of whether the lower bound on the objective value (implied by the points) improves after activating hyperplanes, which Proposition 2.19 resolves. Using the notation from Section 2.5.2, we show that when the ray creating the least cost intersection point $\underline{p}^k$ is cut by a hyperplane, the objective value implied by the new intersection points is greater than or equal to $\underline{z}$.

Proposition 2.19. *Let r be the edge of C that leads to $\underline{p}^k$, i.e., $\underline{p}^k = r \cap \operatorname{bd} S_k$, H be a hyperplane intersected by r before $\operatorname{bd} S_k$, and $\mathcal{P}'$ denote the set of intersection points originating at $r \cap H$, obtained by activating H. Then $\min_p\{c^\mathsf{T} p : p \in \mathcal{P}'\} \geq \underline{z}$.*

Proof. Suppose S_k^1 is the facet of S_k containing $\underline{p}^k$, and let S_k^2 be the opposite facet. We have that $\underline{z} = \min_x\{c^\mathsf{T} x : x \in C \cap S_k^1\} \leq \min_x\{c^\mathsf{T} x : x \in C \cap S_k^2\}$. Since each of the points $p \in \mathcal{P}'$ is either (possibly strictly) in $C \cap S_k^1$ or $C \cap S_k^2$, the result follows. □

2.6.2 Characterizing bounded objective functions for the PRLP

We turn to an analysis of (2.2). It is possible for the optimal solution to (2.2) to be unbounded, a behavior we have in fact observed in our numerical implementation. To better understand this, in this section we present some structural properties of $\operatorname{CutRegion}(\bar{\beta}, \mathcal{P}, \mathcal{R})$, the feasible region to (2.2) for a fixed right-hand size $\bar{\beta}$, that characterize the objective function choices leading to unboundedness.

We begin by studying when the system $\operatorname{CutRegion}(\bar{\beta}, \mathcal{P}, \mathcal{R})$ has valid cuts for a given proper point-ray collection. Recall that $\mathcal{K}'$ denotes the connected component of the skeleton of P that includes $\bar{x} \cap \operatorname{int} S$, and any inequality feasible to (2.1) that cuts a point $v \in \mathcal{K}'$ is valid. We will consider the system

$$\mathcal{G}^\# := \{\alpha : \alpha \in \operatorname{CutRegion}(\bar{\beta}, \mathcal{P}, \mathcal{R});\ v^\mathsf{T}\alpha < \bar{\beta}\}.$$

Theorem 2.20. *Let $(\mathcal{P}, \mathcal{R})$ be a proper point-ray collection and let $v \in \mathcal{K}'$. The system CutRegion$(\bar{\beta}, \mathcal{P}, \mathcal{R})$ has valid cuts as feasible solutions in the following cases: (1) for $\bar{\beta} = 1$ if and only if $0 \notin \mathcal{G}$ and $v \notin \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{P} \cup \mathcal{R}) = \mathcal{G} + \operatorname{cone}(\mathcal{P})$, (2) for $\bar{\beta} = -1$ if and only if $v \notin \operatorname{conv}(\mathcal{G} \cup \{0\})$, and (3) for $\bar{\beta} = 0$ if and only if $v \notin \operatorname{cone}(\mathcal{P} \cup \mathcal{R})$.*

Proof. Let Q be the $|\mathcal{P}| \times n$ matrix containing the intersection points in $\mathcal{P}$ as its rows, and R be the $|\mathcal{R}| \times n$ matrix with rows comprised of the rays in $\mathcal{R}$. Let e denote the n-vector of all ones. Using the nonhomogeneous Farkas' lemma [116], $\mathcal{G}^{\#}$ has a feasible solution if and only if the following two systems are infeasible:

$$\left\{\lambda, \mu \geq 0: \begin{array}{l} Q^{\mathsf{T}}\lambda + R^{\mathsf{T}}\mu = v \\ \bar{\beta} e^{\mathsf{T}}\lambda \geq \bar{\beta} \end{array}\right\} \qquad \left\{\lambda, \mu \geq 0: \begin{array}{l} Q^{\mathsf{T}}\lambda + R^{\mathsf{T}}\mu = 0 \\ \bar{\beta} e^{\mathsf{T}}\lambda > 0 \end{array}\right\}$$

When $\bar{\beta} = 1$, the first system is infeasible if and only if $v \notin \mathcal{G} + \operatorname{cone}(\mathcal{P})$, and the second system is infeasible if and only if $0 \notin \mathcal{G}$, as the existence of a solution (λ, μ) implies $(\lambda / e^{\mathsf{T}}\lambda, \mu)$ is also feasible. When $\bar{\beta} = -1$, the first system is infeasible if and only if $v \notin \operatorname{conv}(\mathcal{G} \cup \{0\})$, and the second system is always infeasible, since $\lambda \geq 0$. When $\bar{\beta} = 0$, the first system is infeasible if and only if $v \notin \operatorname{cone}(\mathcal{P} \cup \mathcal{R})$, and the second system is always infeasible. □

The feasible region to (2.2) is $\operatorname{CutRegion}(\bar{\beta}, \mathcal{P}, \mathcal{R})$, not $\mathcal{G}^{\#}$. However, if we assume that v is used as the objective to (2.2), then Theorem 2.20 can be used to show when (2.2) has a finite solution. Observe that (2.2) implicitly ranks valid inequalities and picks the most violated cut with respect to v. If there exists a homogeneous inequality valid for $\mathcal{G}$ that cuts off v, this ranking breaks down, since all homogeneous inequalities can be scaled to have arbitrarily large violation and hence are unbounded directions in (2.2).

From the $\bar{\beta} = 0$ case in Theorem 2.20, it follows that the linear program (2.2) is bounded if and only if v belongs to $\operatorname{cone}(\mathcal{P} \cup \mathcal{R})$. Corollary 2.21 characterizes the two open objective function sets within $\operatorname{cone}(\mathcal{P} \cup \mathcal{R})$ that admit valid cuts of only one type, either with right-hand side 1 or -1.

Corollary 2.21. *The system* (2.2) *has valid inequalities that cut off a point v only for*

1. $\bar{\beta} = 1$ *if and only if* $0 \notin \mathcal{G}$ *and* $v \in \operatorname{conv}(\mathcal{G} \cup \{0\}) \setminus \mathcal{G}$.
2. $\bar{\beta} = -1$ *if and only if* $v \in (\mathcal{G} + \operatorname{cone}(\mathcal{P})) \setminus \mathcal{G}$.

Proof. Notice that

$$\operatorname{conv}(\mathcal{G} \cup \{0\}) \setminus \mathcal{G} \subseteq \operatorname{conv}(\mathcal{G} \cup \{0\}) \subseteq \operatorname{cone}(\mathcal{P} \cup \mathcal{R})$$

and $\{\operatorname{conv}(\mathcal{G} \cup \{0\}) \setminus \mathcal{G}\} \cap \{\mathcal{G} + \operatorname{cone}(\mathcal{P})\} = \emptyset$. Therefore, the result in part 1 follows from Theorem 2.20. The proof of the second part is similar. □

2.7 Computational results

This section contains the results from computational experiments with PHA_1 as used in Algorithm 2.3 from Section 2.3 and Algorithm 2.4 from Section 2.4. The purpose of these experiments is exploratory; that is, we seek conditions and implementation choices that lead to strong GICs. To do this, we evaluate the effect of a variety of parameters, including those discussed in Section 2.5, and identify structural properties of instances that can be taken advantage of to find stronger cuts. In particular, our results indicate that it can be beneficial to seek a multitude of intersection points that are deep or final as discussed in Section 2.5.1, and to subsequently target GICs tight on these points through objective functions.

We test Algorithm 2.3 and Algorithm 2.4 when used independently, as well as in combination. The purpose of this is to test the strength of targeted tilting relative to non-tilted hyperplanes as in Algorithm 2.3. When used together, we first perform Algorithm 2.4 and afterwards perform non-tilted activations. This is equivalent to inserting the steps of Algorithm 2.4 before step 7 of Algorithm 2.3.

2.7.1 Experimental setup

Parameters. Several parameters are fixed throughout the experiments, while others are varied, summarized in Table 2.2 and elaborated on below.

Table 2.2: Parameters that are varied and the values considered.

Parameter	Values Considered
Hyperplane scoring function, $\mathcal{SC}$	{H1, H2, H3}
Objective functions, $\mathcal{O}$	$\{(R), (V), (T), (S)\}$
Targeted tilting	{On, Off}
# non-tilted hyperplanes, k_h	$\{0, 1, 2, 3, 4\}$

The fixed parameters are as follows. The sets used for cut generation are all the elementary splits. The time limits are one hour per set of parameters and at most five seconds per objective function for each time (2.2) is resolved. The instances are not preprocessed, and presolve is turned off for (2.2), as using presolve adversely affected the quality of the solutions. At most 1,000 GICs are generated per instance (a limit seldom attained). As the cardinality of $\mathcal{P}$ may be large, using all the intersection points as objective directions for (2.2) may be prohibitively expensive; as a result, we limit the number of points used for objective functions to 1,000 per instance. The overall zero tolerance is set to 10^{-7}, while ϵ_{frac} (used in Algorithm 2.2) is set to 10^{-3}; i.e., a point is considered fractional if it is at least 10^{-3} from the nearest integer.

We also perform some standard checks to avoid numerically unstable cuts. Cuts with dynamism higher than 10^6 are rejected, where dynamism is the ratio of the largest and smallest cut coefficients. Additionally, for each candidate cut that is generated, its orthogonality with every existing cut is checked. Given two cuts $\alpha_1^\intercal x \geq \beta_1$ and $\alpha_2^\intercal x \geq \beta_2$, their orthogonality is computed as $1 - \frac{\alpha_1 \cdot \alpha_2}{\|\alpha_1\|_2 \|\alpha_2\|_2}$. If the cut already in the pool that is most parallel to the candidate cut has orthogonality less than 0.01 and one of these two cuts is sparser or has a larger Euclidean distance from the LP optimal solution, then only that cut is kept.

While these parameters are fixed, we evaluate the effect of varying which hyperplanes are to be activated and which objective functions are to be used for (2.2). In Algorithm 2.3, we vary k_h from 0 to 4, where $k_h = 0$ implies we do not perform any non-tilted activations. Section 2.7.3 contains computational results relating to the hyperplane activation rules described in Section 2.5.1. Section 2.7.4 presents the results for the objective functions discussed in Section 2.5.2.

Cut generation. We generate cuts in the nonbasic space. The vertex $\bar{x}$ of $\bar{C}$ in this space corresponds to the zero vector, and the jth ray of $\bar{C}$ in the nonbasic space has a single nonzero jth entry. As a result, the intersection points or rays defining SICs (one pivot from $\bar{x}$) and GICs from PHA on $\bar{C}$ (two pivots from $\bar{x}$) have one and two nonzero coordinates, respectively. Thus, the constraint coefficient matrix of (2.2) is sparse. This improves the time complexity of implementing Algorithm 2.2. In addition, it implies that it is sufficient to consider cuts with $\beta = 1$ to obtain all valid inequalities that cut $\bar{x}$.

Environment. All algorithms are implemented in C++ in the COIN-OR framework [112], with `Clp version 1.15` as the LP solver, using a 64-bit `PowerEdge R515` with 64GB of memory and twelve `AMD Opteron 4176` processors clocked at 2.4GHz. The operating system is `Linux Fedora 19` and compiler is `g++ version 4.8.3 20140911 (Red Hat 4.8.3-7)`.

Cut evaluation. The metric we use to evaluate the cuts obtained from a specific set of parameters is *percent gap closed*. The gap is defined as the difference between the optimal values of the integer program and its linear programming relaxation. Denoting the optimal value of the integer program by z_{IP}, of its linear relaxation by z_{LP}, and of the linear relaxation with cuts added by z', we have

$$\%\text{ gap closed} := 100 \times (z' - z_{\mathrm{LP}})/(z_{\mathrm{IP}} - z_{\mathrm{LP}}).$$

The baseline we use is the percent gap closed by SICs, which are the simplest GICs. We then add GICs along with the SICs to assess what additional effect GICs have on the percent gap closed in the presence of the SICs.

Instance selection. We test forty instances selected from MIPLIB [5, 43, 44, 109] (all versions) based on the following criteria meant to identify small problems so that many different parameter settings can be tested (over 200 in our experiments): (1) The number of rows and number of columns must be no more than 500 each. (2) The instance has to be integer-feasible with $z_{\mathrm{LP}} < z_{\mathrm{IP}}$, and the gap closed by SICs is not 100%. (3) There must be at least one non-final intersection point created from intersecting the rays of $\bar{C}$ with $\operatorname{bd} S$. (4) The instance must not be known to have 0% gap closed from split cuts based on previous experiments [35, 67]. Criterion 3 exists because we do not cut rays of $\bar{C}$ that do not intersect $\operatorname{bd} S$, so if all intersection points are final, no hyperplanes will be activated.

We modify the `stein15`, `stein27`, and `stein45` instances to reduce symmetry, by replacing the objective $\sum_{j=1}^{n} x_j$ with $\sum_{j=1}^{n} j x_j$. We also remove the cardinality constraint $\sum_{j=1}^{n} x_j \geq (n-1)/2$, as this is not present in the initial formulation of these instances [86]. In addition, we exclude instances `go19` and `pp08aCUTS`: the former because its continuous relaxation solves exceptionally slowly, and the latter because it is simply a strengthened version of `pp08a`.

2.7.2 Point-ray collection statistics

Consider the following statistics averaged across all the instances and all the splits used for each instance, for the best combination of parameters for each instance. The first statistic shows the high prevalence of rays of $\bar{C}$ that do not intersect $\operatorname{bd} S$, which demonstrates the impact of Proposition 2.10 and our resulting decision to always set the parameter $\mathcal{R}_A$ of

Algorithm 2.2 to a subset of the rays of $\bar{C}$ that intersect bd S. Of the rays of the initial cone $\bar{C}$, 58% belong to $\mathcal{R}_0$. An additional 5% of the initial rays lead to final intersection points, leaving possibly few rays that can be cut for some instances. For eight instances, only 10% of the rays are able to be cut.

We can also provide an idea of the number of rows in (2.2) in practice, i.e., the number of points and rays we generate. The number of points ranges from about 21 to 35,000, with an average of 2,350 and median of 217, though there exist instances with split sets leading to as few as 2 and as many as 46,000 points. The number of rays in (2.2) ranges from 0 to 1,600, with an average of 220 and median of 160. Figure 2.2 plots the number of generated points across instances (for the best parameter combination for each instance) as a function of the number of rays of $\bar{C}$ that can be cut; we observe a quadratic relationship, as predicted by our analysis in Section 2.3.

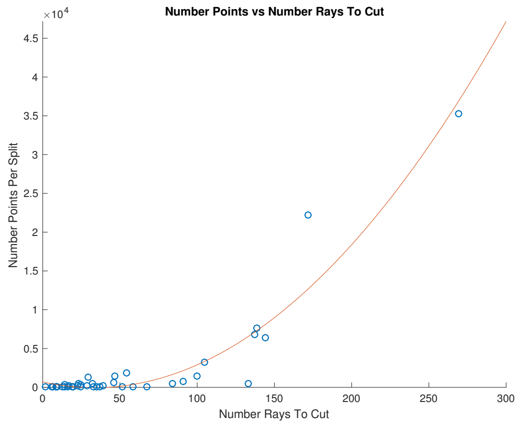


Figure 2.2: Number generated points versus number of rays that can be cut.

2.7.3 Effect of hyperplanes activated

In this section, we analyze how the choice of hyperplanes to activate, as described in Section 2.5.1, affects the percent gap closed by GICs. In addition, we consider whether targeted tilting leads to better cuts. In Section 2.4, we discussed that, theoretically, neither Algorithm 2.3 nor targeted tilting is strictly stronger than the other. As a result, computational experiments are necessary to assess the practical strength of each approach.

Table 2.3 shows the effect of hyperplane activation choices on gap closed for the 29 instances in which GICs close additional gap over SICs. All possible objective functions (discussed in Section 2.5.2) for (2.2) are used. The best percent gap closed is shown per instance in: column 2 for SICs; column 3 for GICs across all parameter settings; columns 4 through 7 for Algorithm 2.3 with $k_h \in \{1, 2, 3, 4\}$; columns 8 through 10 for targeted tilting using each of the hyperplane scoring functions (H1), (H2), (H3); and columns 11 through 14 for when targeted tilting is used in conjunction with Algorithm 2.3 (for $k_h \in \{0, 1, 2, 3\}$). Note that column T+0 is simply the best result from columns 8 through 10.

The results show that Algorithm 2.3 closes 98% of the best percent gap closed, and that it achieves the best result across all settings for 55% of the instances without using any

Table 2.3: Best percent gap closed by hyperplane activation choice for instances with any improvement over SICs. Some values differ in the thousandths digit.

			Only Alg. 2.3				Only Alg. 2.4			Alg. 2.4 with Alg. 2.3			
Instance	SIC	Best	+1	+2	+3	+4	H1	H2	H3	T+0	T+1	T+2	T+3
bell3a	44.74	59.48	59.07	59.07	59.48	59.48	58.94	58.68	58.94	58.94	59.08	59.08	59.08
bell3b	44.57	59.10	59.10	59.10	59.10	59.10	51.36	53.45	51.36	53.45	53.45	53.45	53.45
bell4	23.37	26.52	24.67	24.67	25.26	25.26	23.82	26.35	23.83	26.35	26.35	26.52	26.52
bell5	14.53	85.37	85.37	85.37	85.37	85.37	17.58	19.99	17.58	19.99	19.99	19.99	19.99
blend2	16.00	18.72	16.28	16.28	16.28	16.28	18.72	16.94	18.72	18.72	18.72	18.72	18.72
bm23	5.92	10.98	8.00	10.55	10.98	10.98	8.88	9.94	8.94	9.94	9.94	9.94	9.94
egout	51.57	51.60	51.57	51.57	51.57	51.57	51.59	51.60	51.59	51.60	51.60	51.60	51.60
k16x240	7.56	7.71	7.71	7.71	7.71	7.71	7.56	7.65	7.56	7.65	7.65	7.65	7.65
lseu	4.57	4.65	4.57	4.65	4.65	4.65	4.65	4.57	4.65	4.65	4.65	4.65	4.65
mas74	3.30	4.31	4.30	4.31	4.31	4.31	4.11	3.81	3.98	4.11	4.11	4.11	4.11
mas76	2.37	2.46	2.38	2.46	2.46	2.46	2.37	2.37	2.37	2.37	2.37	2.37	2.37
mas284	0.38	0.41	0.38	0.38	0.38	0.38	0.41	0.41	0.41	0.41	0.41	0.41	0.41
misc05	3.60	4.48	3.60	3.60	3.60	3.60	3.60	4.48	3.60	4.48	4.48	4.48	4.48
mod008	1.31	1.38	1.31	1.38	1.38	1.38	1.31	1.31	1.31	1.31	1.31	1.31	1.31
mod013	4.41	7.37	4.41	4.72	4.72	7.28	4.72	7.28	4.72	7.28	7.37	7.37	7.37
modglob	9.59	13.79	9.62	13.79	13.79	13.79	9.97	10.04	9.97	10.04	10.05	10.05	10.05
p0033	1.83	5.19	1.83	2.59	2.59	2.59	5.19	5.19	5.19	5.19	5.19	5.19	5.19
p0282	3.67	5.05	4.65	4.81	4.81	5.05	4.09	4.81	4.09	4.81	4.83	4.87	4.87
p0291	27.78	40.07	39.80	40.07	40.07	40.07	27.92	31.81	27.92	31.81	31.81	31.81	31.81
pipex	0.81	1.44	1.36	1.36	1.36	1.36	1.44	1.44	1.44	1.44	1.44	1.44	1.44
pp08a	51.44	54.46	52.47	54.41	54.46	54.46	52.76	53.36	52.76	53.36	53.37	53.37	53.37
probportfolio	25.14	25.27	25.14	25.14	25.14	25.14	25.17	25.27	25.17	25.27	25.27	25.27	25.27
sample2	5.86	10.25	5.86	5.86	5.86	5.86	5.86	10.25	5.86	10.25	10.25	10.25	10.25
sentoy	10.38	12.97	10.38	11.10	12.97	12.97	11.89	12.45	12.39	12.45	12.45	12.45	12.45
stein15*	50.00	58.33	50.00	50.00	58.33	58.33	56.03	50.00	56.03	56.03	57.16	57.16	57.16
stein27*	7.41	9.48	7.41	7.86	7.86	7.86	9.47	9.48	9.47	9.48	9.48	9.48	9.48
stein45*	7.10	7.51	7.51	7.51	7.51	7.51	7.36	7.47	7.36	7.47	7.47	7.47	7.47
vpm1	9.99	10.18	9.99	9.99	9.99	9.99	9.99	10.18	9.99	10.18	10.18	10.18	10.18
vpm2	10.19	11.71	11.04	11.19	11.19	11.71	11.15	11.23	11.15	11.23	11.23	11.23	11.23
Average	15.50	21.04	19.65	20.05	20.46	20.57	17.17	17.65	17.18	17.94	17.99	18.00	18.00
Number wins			4	5	4	3				11	1	1	0

targeted tilting. This is achieved when $k_h = 4$, though the marginal impact of activating more hyperplanes diminishes as k_h increases. The table also shows that the effect of targeted tilting is mixed. Targeted tilting alone wins 11 times, meaning it achieves the best result for 38% of the instances. Moreover, we see that targeted tilting is effective in its goal of generating cuts that improve over SICs. Indeed, GICs improve over SICs for all instances (using three decimal places) when using targeted tilting without any non-tilted activations, i.e., no Algorithm 2.3 (column T+0), whereas when using Algorithm 2.3 on its own, six instances (`egout`, `mas284`, `misc05`, `probportfolio`, `sample2`, and `vpm1`) see no improvement at all over SICs. On the other hand, targeted tilting may lead to weaker cuts, as reflected in the fact that the best average percent gap closed is achieved by Algorithm 2.3 on its own, and that Algorithm 2.3 seems to have a relatively insignificant impact on improving the results once targeted tilting has been used.

In our experiments, we find that though final intersection points are targeted, they represent a small percent (12% on average) of all generated points. This does not necessarily imply bad cuts: for example, GICs close an additional 8% of the gap for `stein15_nosym` over SICs alone, but only 2 of the 300 points created are final. Nevertheless, this suggests that one promising future direction to obtain stronger cuts is to target more final intersection points, as this will lead to more facet-defining inequalities for $\operatorname{conv}(P \setminus \operatorname{int} S)$.

2.7.4 Evaluating objective function choices

Next, we examine which objective functions for (2.2) lead to the strongest cuts. Table 2.4 shows the effect of the possible values of the parameter $\mathcal{O}$ when all other parameters are allowed to vary freely. The first column gives the best percent gap closed across all parameter settings. The next column provides the best percent gap closed when *only* objective functions (R) are used. In the next three columns, we show the percent gap closed when (R) is used in conjunction with one of the other types of approaches. The last three columns show the effect of using (R) and any two of the remaining approaches.

Our experiments show that the simplest and fastest approach, of trying to cut along the ray directions, works well in practice, closing 98% of the gap closed when using the more involved heuristics and achieving the best result for 12 of the 29 instances with nonzero extra gap closed. The next best choice of objective function is (T). Using (R) and (T) together achieves the best result across all parameter settings for 22 of the 29 instances with nonzero extra percent gap closed. Moreover, over 76% of the cuts that are active at the post-cut optimum come from procedures (R) and (T), on average.

We also tested a bilinear program that finds a cut with maximal violation with respect to a vertex of P with all SICs added, but it did not yield additional strong cuts.

2.7.5 Strength of GICs

Finally, we look at the best percent gap closed across all parameter settings. We compare the percent gap closed by using GICs and SICs together to using SICs on their own. When the GICs close an additional amount of the integrality gap, it is clear that we are getting a tighter relaxation of the integral hull. The converse is not true; that the extra gap closed by GICs is zero does not imply that the cuts have no effect on tightening the relaxation.

Table 2.4: Best percent gap closed by objective function used, for instances with any improvement over SICs. Some values differ in the hundredths digit.

Instance	Best	R	R+V	R+S	R+T	R+V+S	R+V+T	R+S+T
bell3a	59.48	59.48	59.48	59.48	59.48	59.48	59.48	59.48
bell3b	59.10	59.10	59.10	59.10	59.10	59.10	59.10	59.10
bell4	26.52	26.34	26.34	26.34	26.52	26.34	26.52	26.52
bell5	85.37	85.37	85.37	85.37	85.37	85.37	85.37	85.37
blend2	18.72	16.29	18.72	16.29	18.69	18.72	18.72	18.69
bm23	10.98	9.48	10.43	10.64	10.98	10.64	10.98	10.98
egout	51.60	51.60	51.60	51.60	51.60	51.60	51.60	51.60
k16x240	7.71	7.71	7.71	7.71	7.71	7.71	7.71	7.71
lseu	4.65	4.65	4.65	4.65	4.65	4.65	4.65	4.65
mas74	4.31	4.30	4.30	4.30	4.30	4.31	4.30	4.31
mas76	2.46	2.38	2.38	2.38	2.46	2.38	2.46	2.46
mas284	0.41	0.39	0.39	0.41	0.41	0.41	0.41	0.41
misc05	4.48	4.48	4.48	4.48	4.48	4.48	4.48	4.48
mod008	1.38	1.37	1.37	1.38	1.37	1.38	1.37	1.38
mod013	7.37	7.37	7.37	7.37	7.37	7.37	7.37	7.37
modglob	13.79	13.79	13.79	13.79	13.79	13.79	13.79	13.79
p0033	5.19	2.59	2.59	5.19	5.19	5.19	5.19	5.19
p0282	5.05	4.82	4.87	5.01	4.92	5.01	4.92	5.05
p0291	40.07	39.91	39.91	40.00	40.07	40.00	40.07	40.07
pipex	1.44	1.43	1.43	1.44	1.44	1.44	1.44	1.44
pp08a	54.46	53.52	54.46	54.45	53.58	54.46	54.46	54.45
probportfolio	25.27	25.27	25.27	25.27	25.27	25.27	25.27	25.27
sample2	10.25	5.86	10.25	10.25	10.25	10.25	10.25	10.25
sentoy	12.97	12.45	12.45	12.97	12.85	12.97	12.85	12.97
stein15*	58.33	58.33	58.33	58.33	58.33	58.33	58.33	58.33
stein27*	9.48	9.48	9.48	9.48	9.48	9.48	9.48	9.48
stein45*	7.51	7.47	7.47	7.51	7.51	7.51	7.51	7.51
vpm1	10.18	10.03	10.18	10.18	10.18	10.18	10.18	10.18
vpm2	11.71	11.71	11.71	11.71	11.71	11.71	11.71	11.71
Average	21.04	20.59	20.89	20.93	21.00	21.02	21.03	21.04
Number wins		12	5	7	10	0	0	2

Table 2.5: Best percent gap closed and number of cuts.

	Best % gap closed			# cuts			
Instance	SIC	GIC	**Diff**	SICs	Active SICs	GICs	Active GICs
bell3a	44.74	59.48	**14.74**	32	11	150	32
bell3b	44.57	59.10	**14.53**	35	20	90	42
bell4	23.37	26.52	**3.15**	46	20	800	36
bell5	14.53	85.37	**70.85**	25	12	186	77
blend2	16.00	18.72	**2.72**	6	2	169	7
bm23	5.92	10.98	**5.06**	6	0	1000	6
egout	51.57	51.60	**0.03**	38	35	95	77
flugpl	11.74	11.74	**0.00**	10	6	17	6
gt2	83.13	83.13	**0.00**	11	11	16	12
k16x240	7.56	7.71	**0.15**	14	6	57	8
lseu	4.57	4.65	**0.08**	12	6	172	9
mas74	3.30	4.31	**1.00**	12	1	1000	52
mas76	2.37	2.46	**0.10**	11	2	817	12
mas284	0.38	0.41	**0.03**	20	0	901	3
misc05	3.60	4.48	**0.87**	11	3	305	53
mod008	1.31	1.38	**0.07**	5	0	916	2
mod013	4.41	7.37	**2.96**	5	2	105	18
modglob	9.59	13.79	**4.20**	29	15	350	63
p0033	1.83	5.19	**3.35**	6	4	46	24
p0040	6.65	6.65	**0.00**	4	4	24	22
p0282	3.67	5.05	**1.37**	26	4	799	41
p0291	27.78	40.07	**12.29**	10	2	130	20
pipex	0.81	1.44	**0.62**	6	2	559	28
pp08a	51.44	54.46	**3.02**	53	43	303	73
probportfolio	25.14	25.27	**0.13**	125	68	1000	177
sample2	5.86	10.25	**4.39**	12	4	597	17
sentoy	10.38	12.97	**2.59**	8	0	1000	6
stein15*	50.00	58.33	**8.33**	5	2	40	9
stein27*	7.41	9.48	**2.08**	27	2	516	4
stein45*	7.10	7.51	**0.42**	45	3	341	6
timtab1	17.54	17.54	**0.00**	136	38	307	88
vpm1	9.99	10.18	**0.18**	15	10	132	37
vpm2	10.19	11.71	**1.53**	31	12	593	52
Average	17.23	22.10	**4.87**				
glass4	0.00	0.00	**0.00**	72	36	14	12
misc01	0.00	0.00	**0.00**	12	0	90	5
misc02	0.00	0.00	**0.00**	8	8	23	14
misc03	0.00	0.00	**0.00**	22	8	109	14
misc07	0.00	0.00	**0.00**	26	12	909	293
p0201	0.00	0.00	**0.00**	40	19	95	29
rgn	0.00	0.00	**0.00**	19	11	28	13

Table 2.5 shows the best result for percent gap closed for each instance, across all parameter settings. The percent of the integrality gap closed by SICs and by GICs is given in columns 2 and 3. Column 4 shows the difference between columns 3 and 2. Columns 5 and 6 show the number of SICs generated as well as how many of the SICs are active at the optimum of the LP with all cuts added. Columns 7 and 8 show the same for GICs. The top part of the table shows those instances for which GICs or SICs close any gap over the LP relaxation, and the bottom part of the table contains the remaining instances.

GICs close extra gap over SICs for 72.5% of the instances. The average extra percent gap closed across all instances is around 4%, around 4.9% on those instances in which SICs have any effect, and around 5.5% on those instances with nonzero gap closed. This latter number represents a 35% improvement over just using SICs.

Another way to assess strength of cuts is to check activity at the optimum of the LP relaxation after adding all cuts. On average, over 80% of the active cuts are GICs, and though we generate many more GICs than SICs, the GICs contain 1.8 times as many *active* cuts as the *total* number of SICs. Moreover, one can select for the strong cuts with reasonable success. We adopt a common cut selection criterion (see, e.g., [2]) that sorts cuts based on a combination of their *efficacy* (the Euclidean distance by which they cut $\bar{x}$) and their orthogonality to cuts that have already been selected. Adding the cuts in this way and using just five times as many total GICs as SICs, we can close 21.4% gap over SICs (compared to 22.1% using all the GICs), that is, 98% of the improvement from adding all GICs.

2.7.6 Summary

The GICs generated from our PHA approach close a significant percentage of the integrality gap with respect to SICs. This validates the premise of PHA, that stronger intersection cuts can be obtained from collecting a manageable number of intersection points and rays. We examine the trade-off in using PHA$_1$ with and without the targeted tilting algorithm. Using tilting permits more hyperplanes to be activated and is somewhat better at improving over SICs, but can create weaker points that are avoided by the procedure without tilting. Moreover, our experiments identify one aspect of PHA that can be improved to lead to stronger cuts. Only 12% of the points we generate are final, which indicates that the GICs generated from our approach are far from the split closure and motivates future work on new methods targeting such final intersection points. We also evaluate several different objective functions that can be used in (2.2) and determine one (the tight point heuristic) that seems particularly effective, which has implications for any future GIC computational experiments. One of the motivations of the GIC procedure is the ability to generate diverse cuts, which we find is indeed possible. However, we do not wish to add a large number of cuts to our LP relaxation. We show that in fact a a small set of diverse GICs achieves nearly the same result as using all the GICs.

Chapter 3

Cutting Planes by Tilting on Split Disjunctions

3.1 Introduction

This chapter is a primarily theoretical exploration of the generation and strengthening of general-purpose cutting planes for mixed-integer linear programs via a simple tilting framework. Tilting is defined, abstractly, as any operation that uses one inequality to derive another, by changing the angle or right-hand side of the base inequality. In the literature, this idea reappears under a medley of different names, including tilting, but also rotation, lifting, and coefficient strengthening. We introduce a framework that aggregates several existing presolve and cut-generation techniques that utilize the concept of tilting, but are typically not viewed as equivalent. Our contribution is not only proposing an efficient implementation of tilting, but also explicitly discussing connections among, and a unifying geometric perspective for, a variety of these prior closely-related techniques.[1]

Recall our notation from Section 1: $P_I = \{x \in \mathbb{R}^n : Ax \geq b,\ x_j \in \mathbb{Z} \text{ for all } j \in \mathcal{I}\}$, P is the relaxation of P obtained by removing the integrality restrictions, (IP) is the integer programming problem (with respect to objective function c) being solved, (LP) is the associated linear program, $\bar{x}$ is an optimal solution to (LP), and our goal is to tighten the relaxation P by adding valid *cutting planes* (cuts), which are inequalities that are valid for P_I but not P.

The tilting procedure in this chapter is conceptually simple: given a valid inequality for P_I and a split disjunction, tilt the inequality until it is *supporting* for the *disjunctive hull*, which is the convex hull of the union of P intersected with each disjunctive term; we will refer to the disjunctive hull for a split disjunction as the *(single) split hull.* In other words, we tilt the base inequality until it touches a point in P on each side of the split disjunction. The resulting split cuts will typically compare favorably to SICs [21] and GMICs [88], but they might not necessarily be facet-defining for the split hull. We also explore extensions that produce stronger valid cuts that may cut away parts of the split hull, and we show how to utilize several split disjunctions at once to produce one tilted inequality that aggregates information across these splits.

We propose a particularly computationally efficient way to implement tilting from *strong*

[1]Parts of this chapter appeared as a poster at the Mixed Integer Programming Workshop in 2016 [106].

branching or *probing* information that is already readily available in mixed-integer programming solvers. These are methods (see, e.g., [4, 131]) that involve tentatively branching on elementary split disjunctions and using the resulting logical implications to either tighten the formulation P or select a variable to branch on during branch-and-bound. This is an expensive process, in that it involves solving two linear programs per variable (though perhaps not to optimality), but it is empirically effective, such as resulting in drastically reduced branch-and-bound tree sizes. Typically only the optimal value for each program is used, but a basis is also available; we propose a way of using these bases for cut generation, which is a way to further compensate the computational burden of probing and strong branching.

The tilted inequalities are actually strengthened versions of the base inequalities for certain important classes of problems, such as those in which all integer variables are binary. In this special case, the tilting procedure can be used to replace the base inequalities without adding any constraints to the original system. The drawback of tilting is that the new feasible region, while a tighter approximation of the integer hull, may have a denser description and hence lead to slower solving times, which leads to a tradeoff that we do not address directly but merits further investigation. In particular, when the tilted row is denser, it may be worthwhile to keep it as a cut instead of replacing the original inequality in the formulation.

The idea of tilting has consistently recurred in the integer programming literature in various contexts: for cut generation [54, 107, 126], as lifting techniques [74, 137], and in presolve methods [3, 10, 131]. The distinctions among these methods include the choice of base inequality (whether the starting point is a cut or an inequality defining P), the tilting direction (typically elementary split disjunctions), and the relaxation of P_I used for tilting (e.g., the distinction between exact and approximate lifting). It is sensible to study these techniques under one umbrella and discuss the connections to our work. A more computationally demanding, but also more general, tilting method was investigated by Chvátal et al. [54]. Espinoza et al. [80] connected this more general tilting to lifting [137], which we extend by highlighting the relationship of tilting to coefficient strengthening [3, 10, 131]. As a further extension, we discuss the correspondence to *mixing* [95] (equivalent to *star inequalities* [14]), which we will show is a form of tilting. The idea of using strong branching information to tilt has also been proposed in the context of mixed-integer nonlinear programming (using the objective vector for a base inequality) by Kılınç et al. [108].

The types of inequalities generated in this chapter can be seen as somewhere in between the generalized intersection cuts generated in Chapter 2 and the disjunctive cuts from Chapter 4. Namely, unlike the former chapter, tilted cuts will be supporting for the split hull and they are relatively easier to compute, though they only apply to split disjunctions (or intersections of split disjunctions) as opposed to arbitrary simple disjunctions. Compared to the method of the latter chapter, tilted cuts are weaker but can be generated much more quickly, which is useful in practical settings, in which simple procedures that sacrifice strength for efficiency are often preferred.

The exact tilting method we propose and test as our algorithmic contribution is not currently implemented in practice, to the best of our knowledge. Our computational experiments, while preliminary, are suggestive of the practical potential of our approach, although significantly more empirical evaluation is required. The integrated tilting framework we develop leverages many existing tilting techniques, which we feel is conducive to future research on understanding and deciding when to use each tilting variant in a solver.

3.2 Tilting an inequality using a split disjunction

In Section 3.2.1, we apply tilting to the special case of objective cuts. We present the general case in Section 3.2.2 and an example in Section 3.2.3. Section 3.2.4 concerns convergence properties of tilting when it is applied recursively. In Section 3.2.5, we discuss a particular computationally efficient tilting algorithm that uses strong branching information. In Section 3.2.6, we derive Farkas multipliers certifying the validity of tilted cuts, which are useful in strengthening approaches, considered in Section 3.2.7.

3.2.1 Tilted objective cut

To simplify exposition, we focus on the case of an elementary split disjunction, though the results straightforwardly extend to any split disjunction. Let $P^1 := \{x \in P : x_k \leq \lfloor \bar{x}_k \rfloor\}$ and $P^2 := \{x \in P : x_k \geq \lceil \bar{x}_k \rceil\}$, where $k \in \mathcal{I}$, and let $S := \{x \in \mathbb{R}^n : \lfloor \bar{x}_k \rfloor \leq x_k \leq \lceil \bar{x}_k \rceil\}$. Let $\operatorname{int} S$ denote the *interior* of S.

We first show how to tilt (equivalently, strengthen) the inequality $c^\mathsf{T} x \geq c^\mathsf{T} \bar{x}$. In fact, this is identical to a result of Kılınç et al. [108, Proposition 3] in the context of mixed-integer nonlinear programming, so we present Proposition 3.1 without proof. For $t \in \{1, 2\}$, let p^t denote an optimal solution to $\min_x\{c^\mathsf{T} x : x \in P^t\}$.

Proposition 3.1. *The following inequality is valid for* $\operatorname{conv}(P^1 \cup P^2)$.

$$c^\mathsf{T} x \geq c^\mathsf{T} p^1 + (c^\mathsf{T} p^2 - c^\mathsf{T} p^1)(x_k - \lfloor \bar{x}_k \rfloor).$$

Proposition 3.1 will be generalized by Theorem 3.3 in Section 3.2.2. The next proposition shows that the tilted objective cut, as in Propostion 3.1, is stronger than the objective cut $c^\mathsf{T} x \geq \min\{c^\mathsf{T} p^1, c^\mathsf{T} p^2\}$.

Proposition 3.2. *The optimal objective value of* (LP) *after adding the tilted objective cut will be* $\min\{c^\mathsf{T} p^1, c^\mathsf{T} p^2\}$. *This value is strictly greater than* $c^\mathsf{T} \bar{x}$ *if and only if* $\{x \in P : c^\mathsf{T} x = c^\mathsf{T} \bar{x}\} \subseteq \operatorname{int} S$.

Proof. Without loss of generality, assume $c^\mathsf{T} p^1 \leq c^\mathsf{T} p^2$. Then the tilted objective cut implies that $c^\mathsf{T} x \geq c^\mathsf{T} p^1$ because $(c^\mathsf{T} p^2 - c^\mathsf{T} p^1)(x_k - \lfloor \bar{x}_k \rfloor) \geq 0$ whenever $x_k \geq \lfloor \bar{x}_k \rfloor$, and, by definition of p^1, $c^\mathsf{T} x \geq c^\mathsf{T} p^1$ for all $x \in P$ with $x_k \leq \lfloor \bar{x}_k \rfloor$. For the second statement, if the optimal face of P lies entirely in $\operatorname{int} S$, then by definition of optimal face, $c^\mathsf{T} p^1 > c^\mathsf{T} \bar{x}$. Similarly, for the converse, if the objective value after adding the tilted objective cut is strictly greater than $c^\mathsf{T} \bar{x}$, then the entire optimal face of P must be removed by the cut; the only points of P that violate the cut lie in $\operatorname{int} S$, so the optimal face of P must belong entirely to $\operatorname{int} S$. □

The tilted objective cut is not parallel to the objective function when $c^\mathsf{T} p^1 \neq c^\mathsf{T} p^2$. This is a desirable property due to computational issues that arise with dual degeneracy when adding inequalities that are parallel to the objective. However, the tilted objective cut we have defined only differs in one coefficient from the original objective. In Section 3.3, we give a method to further tilt using information from several split disjunctions at once, which will yield a tilted objective cut that may be drastically different than the original objective.

3.2.2 Tilted inequalities, the general case

In this section, we show how to tilt a *base inequality* $h^\intercal x \geq b_h$ using a valid split disjunction, $(\pi^\intercal x \leq \pi_{01}) \vee (\pi^\intercal x \geq \pi_{02})$ with $\pi_{01} < \pi^\intercal \bar{x} < \pi_{02}$. Let $P^1 := \{x \in P : \pi^\intercal x \leq \pi_{01}\}$ and $P^2 := \{x \in P : \pi^\intercal x \geq \pi_{02}\}$, which we again assume are nonempty, and let $S := \{x \in \mathbb{R}^n : \pi_{01} \leq \pi^\intercal x \leq \pi_{02}\}$. Let $S^{t=} := \{x \in \mathbb{R}^n : \pi^\intercal x = \pi_{0t}\}$, for $t \in \{1, 2\}$, denote side t of the split disunction. We permit wide split disjunctions ($\pi_{02} > \pi_{01}+1$) as in [48]. Theorem 3.3 provides the formula for computing a *tilted inequality*. Geometrically, the operation we perform is to tilt the base inequality using the hinge $\{x \in S^{2=} : h^\intercal x = b_h\}$ until it is supporting for P^1, and then do the same for the new inequality intersected with $S^{1=}$ until it is a support for P^2. In Section 3.2.3, we will give an example of applying the tilting technique.

Theorem 3.3. *Given $h \in \mathbb{R}^n$, $\beta_1 \in \mathbb{R}$, and $\beta_2 \in \mathbb{R}$ such that $h^\intercal x \geq \beta_t$ is a valid inequality for P^t, $t \in \{1, 2\}$, consider the inequality $\alpha^\intercal x \geq \beta$ defined as*

$$\sum_{j=1}^{n} \left(h_j - \left(\frac{\beta_2 - \beta_1}{\pi_{02} - \pi_{01}} \right) \pi_j \right) x_j \geq \beta_1 - \left(\frac{\beta_2 - \beta_1}{\pi_{02} - \pi_{01}} \right) \pi_{01}. \tag{3.1}$$

The inequality $\alpha^\intercal x \geq \beta$ is valid for $\operatorname{conv}(P^1 \cup P^2)$ *if (1) $P \subseteq S$, or (2) $\alpha^\intercal p < \beta$ for some $p \in P \cap \operatorname{int} S$, or (3) for each $t \in \{1, 2\}$, there exists a point $q^t \in P^t$ such that $\alpha^\intercal q^t = \beta$.*

Proof. For $t \in \{1, 2\}$, we have that

$$\begin{aligned}\{x \in P \cap S^{t=} : \alpha^\intercal x \geq \beta\} &= \{x \in P : h^\intercal x \geq \beta_1 + \left(\frac{\beta_2 - \beta_1}{\pi_{02} - \pi_{01}} \right) (\pi^\intercal x - \pi_{01}),\ \pi^\intercal x = \pi_{0t}\} \\ &= \{x \in P : h^\intercal x \geq \beta_t,\ \pi^\intercal x = \pi_{0t}\}.\end{aligned}$$

This shows that $\alpha^\intercal x \geq \beta$ is valid for $P \cap S^{t=}$, $t \in \{1, 2\}$, and hence valid for $\operatorname{conv}(P^1 \cup P^2)$ under condition (1).

Now consider the other two conditions for validity for $\operatorname{conv}(P^1 \cup P^2)$. (2) If $\alpha^\intercal p < \beta$ for some $p \in P \cap \operatorname{int} S$, then it follows from convexity that the inequality is valid for P^1 and P^2. (3) Suppose, for each $t \in \{1, 2\}$, there exists a point $q^t \in P^t$ satisfying $\alpha^\intercal x = \beta$. Let p be a point in P^1 and $\widehat{p}$ be the point on the line segment pq^2 such that $\pi^\intercal \widehat{p} = \pi_{01}$. Since $\alpha^\intercal q^2 = \beta$ and $\alpha^\intercal \widehat{p} \geq \beta$, convexity implies that $\alpha^\intercal p \geq \beta$. Applying the analogous argument for a point from P^2, we conclude that the inequality is valid for both P^1 and P^2, as desired. □

Note that when we work with an elementary split disjunction on a variable x_k, the tilted cut (3.1) reduces to a straightforward generalization of Proposition 3.1:

$$h^\intercal x \geq \beta_1 + (\beta_2 - \beta_1)(x_k - \lfloor \bar{x}_k \rfloor).$$

Naturally, it is desirable to have a sense of the theoretical strength of tilted inequalities. One proxy for strength is whether is an inequality is facet-defining (or supporting) for the split hull, $\operatorname{conv}(P^1 \cup P^2)$. Unfortunately, in general, tilting is not guaranteed to produce a facet-defining, or even supporting, inequality. On the other hand, we will not tilt using arbitrary coefficient vectors $h \in \mathbb{R}^n$. A mild condition suffices for guaranteeing the tilted inequality to be supporting for $\operatorname{conv}(P^1 \cup P^2)$. We give this condition in Proposition 3.4. The proposition also provides a sufficient condition for the tilted inequality to be a proper cut (i.e., not valid for P) and facet-defining for the split hull.

Proposition 3.4. *Given $(h, b_h) \in \mathbb{R}^n \times \mathbb{R}$, let $\beta_t := \min_x\{h^\mathsf{T} x : x \in P^t\}$ be finite. Let $\alpha^\mathsf{T} x \geq \beta$ be the tilted inequality as defined in* (3.1)*. Suppose that $h^\mathsf{T} x \geq b_h$ is valid for P. (1) If $P \subseteq S$ or there exists a point $p \in P \cap \operatorname{int} S$ such that $h^\mathsf{T} p = b_h$, the inequality $\alpha^\mathsf{T} x \geq \beta$ is supporting for $P \cap S^{t=}$, $t \in \{1, 2\}$. (2) If p as above exists and $b_h < \max\{\beta_1, \beta_2\}$, then $\alpha^\mathsf{T} p < \beta$. (3) Finally, if $h^\mathsf{T} x \geq \beta_t$ defines a face of dimension d_t for $P \cap S^{t=}$, $t \in \{1, 2\}$, then $\alpha^\mathsf{T} x \geq \beta$ defines a face of* $\operatorname{conv}(P^1 \cup P^2)$ *with dimension at least* $\max\{d_1 + 1, d_2 + 1\}$.

Proof. Fix $t \in \{1, 2\}$. To show the first claim, we prove that there exists a point $q^t \in P \cap S^{t=}$ with $\alpha^\mathsf{T} q^t = \beta$. Equivalently, we need q^t to satisfy $h^\mathsf{T} q^t = \beta_t$. Let $q \in P^t$ such that $h^\mathsf{T} q = \beta_t$ (q is guaranteed to exist by definition of β_t). Clearly, $q \in S^{t=}$ when $P \subseteq S$. Now consider the second sufficient condition. Define q^t as the point on the line segment between p and q for which $\pi^\mathsf{T} x = \pi_{0t}$. By convexity of P, $q^t \in P^t$, so that $h^\mathsf{T} q^t \geq \beta_t$, and $\beta_t = h^\mathsf{T} q \geq h^\mathsf{T} q^t \geq h^\mathsf{T} p$, from which it follows that $h^\mathsf{T} q^t = \beta_t$, as desired.

Next, assume that $b_h < \max\{\beta_1, \beta_2\}$. Assume without loss of generality that $\beta_2 \geq \beta_1$. It holds that either $\beta_2 > \beta_1$ or $\beta_1 > b_h$ (or both) and $\pi^\mathsf{T} p \geq \pi_{01} = \pi^\mathsf{T} q^1$. Hence,

$$\alpha^\mathsf{T} p = \alpha^\mathsf{T}(q^1 + p - q^1) = \beta + h^\mathsf{T} p - h^\mathsf{T} q^1 - \left(\frac{\beta_2 - \beta_1}{\pi_{02} - \pi_{01}}\right) \cdot (\pi^\mathsf{T} p - \pi^\mathsf{T} q^1) < \beta.$$

For the last claim, for each $t \in \{1, 2\}$, there exist $d_t + 1$ affinely independent points from $P \cap S^{t=}$ satisfying $\alpha^\mathsf{T} x = \beta$. Any point from the other side of the split disjunction is affinely independent of these $d_t + 1$ points, giving the desired result. □

Corollary 3.5. *If $h^\mathsf{T} x \geq \beta_1$ is valid for and defines an $(n-1)$-dimensional face of $P \cap S^{1=}$, and $h^\mathsf{T} x \geq \beta_2$ is valid for P^2, then $\alpha^\mathsf{T} x \geq \beta$ as in* (3.1) *is facet-defining for the split hull.*

Proposition 3.4 is useful for identifying which base inequalities should be tilted. In particular, let $h^\mathsf{T} x \geq b_h$ be any inequality that is valid for P. Then we can find $\beta_t = \min_x\{h^\mathsf{T} x : x \in P^t\}$ and define the tilted inequality as in Theorem 3.3. As long as $h^\mathsf{T} p = b_h$ for some point $p \in P \cap S$, e.g., using any inequality tight for $\bar{x}$, tilting can be viewed as a strengthening (within S) of $h^\mathsf{T} x \geq b_h$ using the split disjunction. We say an inequality *can be tilted* when tilting yields an inequality stronger (within S) than the original one, which happens if (and only if) $b_h < \max\{\beta_1, \beta_2\}$.

Being stronger within S does not mean that the tilted inequality strictly *dominates* the base inequality with respect to the given split, as defined below.[2]

Definition 3.6. *Let $\ell := \inf_x\{\pi^\mathsf{T} x : x \in P\}$ and $u := \sup_x\{\pi^\mathsf{T} x : x \in P\}$. Then $\alpha^\mathsf{T} x \geq \beta$* dominates *$h^\mathsf{T} x \geq b_h$ with respect to S if every point $p \in R := \{x \in \mathbb{R}^n : \ell \leq x \leq u\}$ that satisfies $\alpha^\mathsf{T} p \geq \beta$ also satisfies $h^\mathsf{T} x \geq b_h$.* Strict dominance *occurs when there exists a point $p \in R$ such that $\alpha^\mathsf{T} p \geq \beta$ but $h^\mathsf{T} p > b_h$.*

[2]When we say one inequality dominates another on a region $R \subseteq \mathbb{R}^n$, we mean that any point from R that satisfies the first necessarily satisfies the second. See, for example, Definition 2.17. In the setting of this chapter, using $R = P$ is not meaningful, because we are not comparing two cuts: when the base inequality is valid for P, and the tilted cut is valid for P_I, then clearly every point valid for the split hull satisfies both inequalities. The definition we use instead is adapted from [3, Definition 2].

Our notion of dominance is equivalent to redundancy: essentially, we are checking whether the tilted inequality can replace the base inequality without introducing new feasible points. In the case of *facial disjunctive programs* [17], such as binary integer programs, tilting can achieve strict dominance, which follows from Proposition 3.7.

Proposition 3.7. *Let $h^\intercal x \geq b_h$ be a valid inequality for P such that $h^\intercal p = b_h$ for some $p \in P \cap \operatorname{int} S$. and let $\beta_t \geq b_h$ such that $h^\intercal x \geq \beta_t$ is valid for P^t, $t \in \{1,2\}$. Let $\alpha^\intercal x \geq \beta$ be the tilted inequality as defined in* (3.1)*. Suppose that the following two conditions hold: (1) $\beta_1 = b_h$ or $P^2 \setminus S^{2=} = \emptyset$, and (2) $\beta_2 = b_h$ or $P^1 \setminus S^{1=} = \emptyset$. The tilted inequality dominates $h^\intercal x \geq b_h$ with respect to S, and it is strictly dominant if and only if $b_h < \max\{\beta_1, \beta_2\}$. If $P \subseteq S$, we do not need to require the existence of $p \in P \cap \operatorname{int} S$.*

Proof. We assume that $b_h < \max\{\beta_1, \beta_2\}$, as otherwise the base inequality remains unchanged. The tilted cut is valid by Theorem 3.3, trivially when $P \subseteq S$, and using the same reasoning as in Proposition 3.4 to show that $\alpha^\intercal p < \beta$. Moreover, it clearly dominates $h^\intercal x \geq b_h$ for points in $P \cap S^{t=}$, with strict domination for either side for which $\beta_t > b_h$, which completes the proof when $P \subseteq S$. Now suppose $P^2 \setminus S^{2=}$ is nonempty, which implies that $\beta_1 = b_h$, by assumption. Let $q \in S^{2=} \setminus P$ (otherwise $h^\intercal q \geq b_h$ by definition) such that $\alpha^\intercal q \geq \beta$. We show that $h^\intercal q > b_h$. Take any $q^1 \in P \cap S^{1=}$ such that $\alpha^\intercal q^1 = \beta$. Let q^2 be the point on the line segment between q and q^1 that intersects $S^{2=}$. Since $h^\intercal q^1 = b_h$ and $h^\intercal q^2 > b_h$, it follows that $h^\intercal q > b_h$. □

3.2.3 Example of tilting a valid inequality for the LP relaxation

We show an example of tilting applied to the mixed-integer program below. The example is shown in maximization form for ease of illustration.

$$\begin{aligned}
\max_{x_1,x_2,x_3} \quad & x_3 \\
& -(1/4)x_1 + x_3 \leq 7/8 \\
& (3/2)x_1 + x_3 \leq 7/4 \\
& -x_2 + x_3 \leq 1/2 \\
& (1/2)x_2 + x_3 \leq 5/4 \\
& (9/4)x_1 + (3/4)x_2 + x_3 \leq 25/8 \\
& x_1, x_2, x_3 \in [0,1] \\
& x_1 \in \mathbb{Z}
\end{aligned}$$

We will tilt the inequality $h^\intercal x = -(1/4)x_1 + x_3 \leq 7/8$ using the elementary split disjunction on variable x_1. For the $x_1 \leq 0$ side, the point $(0, 3/8, 7/8)$ is feasible to P, so $\beta_1 = 7/8$. For $x_1 \geq 1$, $\max_x\{h^\intercal x : x \in P, x_1 \geq 1\} = 0 = \beta_2$, achieved at $(1, 0, 1/4)$. Applying Theorem 3.3, the tilted cut only differs in the coefficient on x_1, which changes to $(-1/4) - (0 - 7/8) = 5/8$. The resulting cut is $(5/8)x_1 + x_3 \leq 7/8$. Note that the conditions in Proposition 3.4 are satisfied; hence, the tilted cut is facet-defining for the split hull. Figure 3.1 depicts this example.

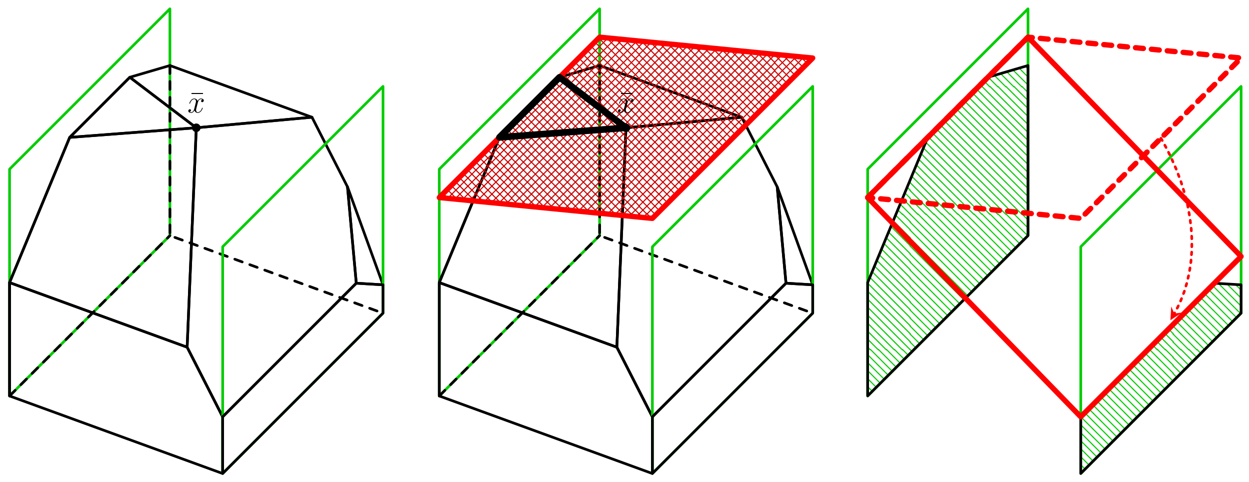


Figure 3.1: Example of tilting an inequality valid for P using an elementary split disjunction.

3.2.4 Existence of tilted cuts

If tilting is applied recursively, i.e., inequalities defining P are recursively tilted via split disjunctions, then it is natural to consider whether this procedure will converge to the convex hull of P_I or stall somewhere in the process. Proposition 3.8 implies that, for $n = 3$, the process converges to a point in P_I even when restricted to elementary split disjunctions.

Proposition 3.8. *When $n = 3$, for any $k \in \mathcal{I}$ such that $\bar{x}_k \notin \mathbb{Z}$, there exists an inequality defining P that can be tilted.*

Proof. We show that, specifically, one of three inequalities that correspond to the nonbasic variables at $\bar{x}$ can be tilted. These three inequalities define a polyhedral cone, C. Let $C^1 := \{x \in C : x_k \leq \lfloor \bar{x}_k \rfloor\}$ and $C^2 := \{x \in C : x_k \geq \lceil \bar{x}_k \rceil\}$. The extreme points of C^1 and C^2 are in one-to-one correspondence with the extreme rays of C. As C has three extreme rays, it follows that either C^1 or C^2 will have exactly only extreme point; say C^1 without loss of generality. In this case, C^1 will exactly be the intersection of two inequalities of C and the disjunctive term inequality, $x_k \leq \lfloor \bar{x}_k \rfloor$. This implies the desired result, as the third inequality defining C will then not be supporting for C^1, meaning it can be tilted. $\square$

In contrast, for $n \geq 4$, recursively tilting only the inequalities defining P via elementary split disjunctions does not, in general, converge to a point in P_I.

Proposition 3.9. *There exists a polyhedron $P \neq P_I$ for which no inequality defining P can be tilted using any elementary split disjunction.*

Proof. The statement is proved via the integer program below.

$$\begin{aligned} \max_{x_1,\ldots,x_4} \quad & x_4 \\ & -x_1 + x_3 + x_4 \leq 0 \\ & \ \ x_1 + x_2 + x_4 \leq 1 \\ & -x_2 \leq 0 \\ & -x_3 \leq 0 \\ & \ \ x_1 \in \mathbb{Z} \end{aligned}$$

An optimal solution to the LP relaxation is $(1/2, 0, 0, 1/2)$. The only split disjunction available is on x_1. However, the points $(0,0,0,0)$, $(0,1,0,0)$, $(1,0,0,0)$, and $(1,0,1,0)$ are feasible to P and demonstrate that none of the right-hand sides of the inequalities defining the problem can be improved. □

Though pathological examples exist for which no inequality defining P can be tilted, such as in Proposition 3.9, it is intuitively very unlikely to encounter such a case (and, indeed, large numbers of inequalities defining P can be tilted for every instance in our computational experiments). Next, in Proposition 3.10, we state a sufficient condition for when a SIC can be tilted. Moreover, under this condition, the tilted cut will dominate the SIC generated from a basis of P. Essentially, the proposition is that a SIC can be strengthened by tilting when it is not supporting for the split hull.

Proposition 3.10. *Let $\gamma^\intercal x \geq \gamma_0$ denote the SIC from a given basis of P. If $\{x \in P^t : \gamma^\intercal x = \gamma_0\} = \emptyset$ for $t \in \{1,2\}$, then there exists an inequality $h^\intercal x \geq b_h$ valid for P that can be tilted such that the resulting tilted cut will strictly dominate $\gamma^\intercal x \geq \gamma_0$ when $P \subseteq S$.*

Proof. The fact that an inequality that can be tilted exists is trivial under the assumption, by letting $h := \gamma$ and $b_h := \gamma^\intercal \bar{x} < \gamma_0$, as it implies $\min_x\{\gamma^\intercal x : x \in P^t\} > \gamma_0$ for $t \in \{1,2\}$; this results in a tilted SIC that dominates the original SIC when $P \subseteq S$, by Proposition 3.7.□

A different way to circumvent the lack of convergence to the optimal solution for the integer hull is to tilt in different directions when using the direction defined by the split is not fruitful. We discuss a method for doing this in Section 3.2.7.

3.2.5 Selecting base inequalities via strong branching

We have shown that tilting, when done appropriately, leads to inequalities that are supporting (but perhaps not facet-defining) for $\operatorname{conv}(P^1 \cup P^2)$. This is generally a desirable property that is not satisfied by other classes of split cuts, such as GMICs. Even lift-and-project (Section 1.4), traditionally thought of as a way of producing facet-defining inequalities, generates cuts that are not supporting the split hull under commonly-used normalizations [84]. Moreover, generating a L&PC involves solving a higher-dimensional linear program, whereas tilting uses the solution of two linear programs in the original dimension of the problem, which will generally be faster.[3] Nevertheless, a naïve implementation of tilting may still seem expensive, as we ostensibly optimize over P^1 and P^2 for each inequality being tilted. In this section, we outline a more computationally efficient tilting procedure utilizing readily-available information from solvers when only elementary splits are used.

The key insight is in the following easy lemma.

Lemma 3.11. *Every solution to* $\min_x\{h^\mathsf{T} x : x \in P^t\}$*, for any* $h \in \mathbb{R}^n$ *and* $t \in \{1,2\}$*, yields the optimal tilting right-hand side* β_t *for at least* $n-1$ *inequalities.*

Proof. Let $p \in \arg\min_x\{h^\mathsf{T} x : x \in P^t\}$. There are at least $n-1$ inequalities defining p other than $\pi^\mathsf{T} x = \pi_{0t}$ (if p lies on $\operatorname{bd} S$), as p is a vertex. The right-hand side of every such inequality cannot be improved (from whatever the original value is in the description of P) using that side of the split disjunction. □

Therefore, in our experiments, after each time solving $\min_x\{h^\mathsf{T} x : x \in P^t\}$, we record all of the inequalities of P tight at that vertex. Hence, when any of those inequalities other than h is used as a base inequality, only one linear program needs to be solved to perform tilting.

Lemma 3.11 is especially useful in light of the fact that, during branch-and-bound, solvers perform strong branching, which involves choosing several potential branching variables and computing an optimal point (with respect to the objective direction c) for each side of the corresponding elementary split disjunctions. Consequently, solvers already have much of the requisite information needed to employ tilting, but that information is not being used.[4]

Strong branching can be used for a further computational improvement that applies to any base inequality, not only one defining P (a generalization we consider later in this chapter). For each split disjunction considered in strong branching, we are given not only an optimal solution $p^t \in \arg\min_x\{c^\mathsf{T} x : x \in P^t\}$ for $t \in \{1,2\}$, but also an associated *cobasis* for p^t. The cobasis contains indices corresponding to n linearly independent inequalities of P^t tight at p^t, including the indices for $n-1$ tight inequalities from P, as in Lemma 3.11. Let C^t, $t \in \{1,2\}$, denote the basis cone, with apex at p^t, defined by the n inequalities corresponding to the cobasis of p^t. As we show next in Corollary 3.12, we can use this cobasis to deduce the optimal right-hand side over P^t for additional base inequalities, namely those that are valid for both C^1 and C^2.

Corollary 3.12. *Suppose that* $h \in \mathbb{R}^n$ *such that* $h^\mathsf{T} r \geq 0$ *for each extreme ray* r *of* C^1 *and* C^2*. Then the following inequality is valid for* $\operatorname{conv}(P^1 \cup P^2)$*:*

$$h^\mathsf{T} x \geq h^\mathsf{T} p^1 + \left(\frac{h^\mathsf{T} p^2 - h^\mathsf{T} p^1}{\pi_{02} - \pi_{01}}\right)(\pi^\mathsf{T} x - \pi^\mathsf{T} \bar{x}).$$

Proof. If $h^\mathsf{T} r \geq 0$ for each extreme ray of C^t, then $h^\mathsf{T} x \geq h^\mathsf{T} p^t$ is valid for C^t, $t \in \{1,2\}$. The result then directly follows from Theorem 3.3 with $\beta_1 = h^\mathsf{T} p^1$ and $\beta_2 = h^\mathsf{T} p^2$. □

[3]There exists an alternative way to produce L&PCs involving pivoting to different basic solutions in the original space [32]; however, one can employ tricks to reduce the computational overhead for tilting as well, such as only using few dual pivots to calculate β_1 and β_2, so we do not pursue this comparison further.

[4]What we call strong branching information can also be gathered during the equivalent presolve step called *probing*. We discuss this more in Section 3.5.

To reiterate, the utility of Lemma 3.11 and Corollary 3.12 is that many inequalities can be tilted simultaneously by solving only one linear program for each side of the split disjunction, and, in fact, such a linear program is already being solved to obtain strong branching information, but it is not being used for generating cuts. This observation drastically reduces the computational overhead of tilting; conversely, tilting could justify the consideration of more strong branching candidates.

We also point to another important reason to focus on p^1 and p^2 when generating cuts. Intuitively, the cuts tight on these two points are likely to be the most helpful ones for closing the integrality gap. We provide computational evidence for this idea later in this chapter; in Chapter 4, we also give some theoretical support for this intuition.

3.2.6 Farkas certificate for tilted cuts

Tilted inequalities from a split disjunction are in the class of split cuts [61]. One can generate all cuts that are valid for a split disjunction via the CGLP as defined in Section 1.4, whose basic feasible solutions are L&PCs. The feasible region of the CGLP, specialized to a split disjunction and without the normalization constraint, is given in the linear system (3.2).

$$\begin{aligned} u^1 A - u_0^1 \pi^\mathsf{T} &= \alpha^\mathsf{T} \\ u^2 A + u_0^2 \pi^\mathsf{T} &= \alpha^\mathsf{T} \\ u^1 b - u_0^1 \pi_{01} &\geq \beta \\ u^2 b + u_0^2 \pi_{02} &\geq \beta \\ u^1, u_0^1, u^2, u_0^2 &\geq 0 \end{aligned} \tag{3.2}$$

Kılınç et al. [108] make the connection between tilted objective cuts and the CGLP; we now extend this to the general case of tilted inequalities. The utility of making this connection is not purely theoretical; in fact, it enables us to apply classical cut strengthening techniques to tilted cuts, as we discuss in Section 3.2.7.

Since the tilted inequality is valid for P^t for $t \in \{1,2\}$, there exist nonnegative Farkas multipliers (w^1, w_0^1) and (w^2, w_0^2) that certify the validity of $\alpha^\mathsf{T} x \geq \beta$, equivalently of $h^\mathsf{T} x \geq \beta_t$, for each of those sets, meaning they satisfy:

$$\begin{aligned} w^1 A - w_0^1 \pi^\mathsf{T} &= h^\mathsf{T} & \qquad w^2 A + w_0^2 \pi^\mathsf{T} &= h^\mathsf{T} \\ w^1 b - w_0^1 \pi_{01} &\geq \beta_1 & \qquad w^2 b + w_0^2 \pi_{02} &\geq \beta_2. \end{aligned}$$

These multipliers are provided to us as dual solutions after solving $\min_x \{h^\mathsf{T} x : x \in P^t\}$.

Theorem 3.13. *Let $h \in \mathbb{R}^n$, $\beta_1 \in \mathbb{R}$, and $\beta_2 \in \mathbb{R}$ such that $h^\mathsf{T} x \geq \beta_t$ is a valid inequality for P^t for each $t \in \{1,2\}$. Define α and β as in Theorem 3.3. Suppose that $\alpha^\mathsf{T} p < \beta$ for some point $p \in P \cap \operatorname{int} S$. Then $(\alpha, \beta, u^1, u_0^1, u^2, u_0^2)$ is feasible to (3.2), where*

$$\begin{aligned} u^1 &= w^1 & \qquad u_0^1 &= w_0^1 + (\beta_2 - \beta_1)/(\pi_{02} - \pi_{01}) \\ u^2 &= w^2 & \qquad u_0^2 &= w_0^2 - (\beta_2 - \beta_1)/(\pi_{02} - \pi_{01}). \end{aligned}$$

Proof. We first verify that the constraints of (3.2) hold:

$$\begin{aligned}
u^1A - u_0^1\pi^\mathsf{T} &= w^1A - \left(w_0^1 + \frac{\beta_2-\beta_1}{\pi_{02}-\pi_{01}}\right)\pi^\mathsf{T} = h^\mathsf{T} - \left(\frac{\beta_2-\beta_1}{\pi_{02}-\pi_{01}}\right)\pi^\mathsf{T} = \alpha^\mathsf{T} \\
u^2A + u_0^2\pi^\mathsf{T} &= w^2A + \left(w_0^2 - \frac{\beta_2-\beta_1}{\pi_{02}-\pi_{01}}\right)\pi^\mathsf{T} = h^\mathsf{T} - \left(\frac{\beta_2-\beta_1}{\pi_{02}-\pi_{01}}\right)\pi^\mathsf{T} = \alpha^\mathsf{T} \\
u^1b - u_0^1\pi_{01} &= w^1b - \left(w_0^1 + \frac{\beta_2-\beta_1}{\pi_{02}-\pi_{01}}\right)\pi_{01} \geq \beta_1 - \left(\frac{\beta_2+\beta_1}{\pi_{02}-\pi_{01}}\right)\pi_{01} = \beta \\
u^2b + u_0^2\pi_{02} &= w^2b + \left(w_0^2 - \frac{\beta_2-\beta_1}{\pi_{02}-\pi_{01}}\right)\pi_{02} \geq \beta_2 - \left(\frac{\beta_2-\beta_1}{\pi_{02}-\pi_{01}}\right)\pi_{02} = \beta.
\end{aligned}$$

It remains to prove that u_0^1 and u_0^2 are nonnegative. Without loss of generality, assume that $\beta_2 \geq \beta_1$; this implies $u_0^1 \geq 0$. Next, assume for the sake of contradiction that u_0^2, as defined, is negative. We show that this would imply that $\alpha^\mathsf{T}p \geq \beta$. Geometrically, as we are only changing u_0^1 and u_0^2, the interpretation is that we obtain $\alpha^\mathsf{T}x \geq \beta$ by tilting the base inequality using the disjunctive term inequalities $\pi^\mathsf{T}x \leq \pi_{01}$ and $\pi^\mathsf{T}x \geq \pi_{02}$; when $u_0^2 < 0$, the implication is that we have "tilted too much" on P^2. Using this intuition, we proceed algebraically. Let $\bar{\alpha}^\mathsf{T} = u^2A$ and $\gamma = u^2b$; $\bar{\alpha}^\mathsf{T}x \geq \gamma$ is a consequence of $Ax \geq b$, so it follows that it is valid for P, i.e., that $\bar{\alpha}^\mathsf{T}p \geq \gamma$. Observe also that $\alpha = \bar{\alpha} + u_0^2\pi = \bar{\alpha} - |u_0^2|\pi$. Hence,

$$\alpha^\mathsf{T}p = \bar{\alpha}^\mathsf{T}p - |u_0^2|\pi^\mathsf{T}p \geq \gamma - |u_0^2|\pi_{02} = u^2b - |u_0^2|\pi_{02} = u^2b + u_0^2\pi_{02} \geq \beta,$$

which is the desired contradiction. □

3.2.7 Generating stronger tilted cuts

This section outlines ideas for generating stronger tilted cuts. One straightforward direction is using tilting to strengthen any cut valid for the split hull. For example, one can generate a GMIC from a basis of P and a split disjunction, then apply tilting to the GMIC as a base inequality, using either the same or a different split disjunction. The resulting cut can then be further strengthened by modularization, which we discuss next.

A posteriori strengthening by modularization. There are several generic ways to use integrality information to strengthen a given cut, which are more or less equivalent when applied to a cut derived from a split disjunction. These include *monoidal cut strengthening* [28], *standard modularization* [34], and the group-theoretic perspective, such as *trivial lifting* [75, 100]. Applying these techniques (see Section 1.5) simply requires a set of Farkas multipliers certifying the validity of each cut,[5] which is precisely a solution to the system (3.2).

Given a feasible solution $(\alpha, \beta, u^1, u_0^1, u^2, u_0^2)$ to (3.2), one can define a strengthened cut $\widetilde{\alpha}^\mathsf{T}x \geq \beta$ using Corollary 1.11 from Section 1.5. For all $j \in [n] \setminus \mathcal{I}$, define $\widetilde{\alpha}_j := \alpha_j$. The

[5]Actually, monoidal strengthening also requires the split variable to be bounded over P. Standard modularization works without this assumption.

coefficient $\widetilde{\alpha}_j$, for $j \in \mathcal{I}$ is set as follows. Let $z^* = (u^1 - u^2)A_{\cdot j}/(u_0^1 + u_0^2)$ for any $j \in \mathcal{I}$, where $A_{\cdot j}$ denotes column j of A. Replace α_k by

$$\widetilde{\alpha}_j := \max\{u^1 A_{\cdot j} + u_0^1 \lceil z^* \rceil\ ,\ u^2 A_{\cdot j} - u_0^2 \lfloor z^* \rfloor\}.$$

The cut $\widetilde{\alpha}^\mathsf{T} x \geq \beta$ is valid for $\operatorname{conv}(P^1 \cup P^2)$ and is stronger than $\alpha^\mathsf{T} x \geq \beta$. Theorem 3.13 gives a feasible solution to (3.2) for the tilted cuts we have defined so far, enabling tilted cuts to be modularized efficiently.

Strengthening the disjunction. Modularizing the cut coefficients is applied after a cut (and corresponding Farkas multipliers) has been generated. Conceptually, the proof of the strengthening comes from showing that the new cut is valid for a strengthened version of the original disjunction, though we never explicitly optimize over the disjunction. An alternative is to first strengthen the disjunction and then produce a cut. This may, in general, produce a different cut than modularization of an inequality from the original split disjunction.

Further tilting. One perspective for tilting that we mentioned is that of taking the cut restricted to one side of the split disjunction and using this as a hinge to tilt until the inequality is supporting for the other side of the split disjunction. This corresponds to taking a face-defining inequality for, say, $S^{1=}$ and appending one point from P^2. It is natural to consider incorporating a second point from P^2, leading to tilting along a different hinge and direction, which can further increase the dimension of the face of $\operatorname{conv}(P^1 \cup P^2)$ defined by the cut. (Alternatively, if the initial tilted cut already defines a facet of the split hull, it is worthwhile to pursue a second facet-defining inequality.) This has strong connections to the tilting technique studied by Chvátal et al. [54] (which itself is related to lifting, as discussed in [80]), and also to the tilting idea outlined in the dissertation of Perregaard [126, Section 4]. We elaborate on these connections in Section 3.5.

Let $\bar{\alpha}^\mathsf{T} x \geq \gamma$ be a given tilted cut from a base inequality $h^\mathsf{T} x \geq b_h$. To perform additional tilting, first we find, for $t \in \{1, 2\}$, a point $q^t \in \arg\min_x\{\bar{\alpha}^\mathsf{T} x : x \in P^t\}$ and a corresponding cobasis, which in most cases is exactly the optimal solution to $\min_x\{h^\mathsf{T} x : x \in P^t\}$. This point will satisfy $\bar{\alpha}^\mathsf{T} q^t = \gamma$ (if not, we can tilt until the cut is supporting). Along with the point, we will be given the dual values which allow us to apply Theorem 3.13 to calculate the Farkas multipliers, say $\{\bar{u}^1, \bar{u}_0^1, \bar{u}^2, \bar{u}_0^2\}$, certifying the validity of $\bar{\alpha}^\mathsf{T} x \geq \gamma$, which we will use to further tilt the inequality.

First, we state a simple but useful complementary slackness condition that must be satisfied by the Farkas multipliers. For convenience, define $d^1 := -\pi^\mathsf{T}$, $d_0^1 := -\pi_{01}$, $d^2 := \pi^\mathsf{T}$, and $d_0^2 := \pi_{02}$.

Lemma 3.14. *Let $(\alpha, \beta, \{u^1, u_0^1, u^2, u_0^2\})$ denote a feasible solution to (3.2) in which u_0^1 and u_0^2 are both positive. Fix $t \in \{1, 2\}$, define $N^+ := \{i \in [m] : u_i^t > 0\}$, and let $q \in P^t$. It holds that $\alpha^\mathsf{T} q = \beta$ if and only if $A_{i\cdot} q = b_i$ for all $i \in N^+$ and $d^t q = d_0^t$.*

Proof. If $\alpha^\mathsf{T} q = \beta$, then

$$\beta = \alpha^\mathsf{T} q = u^t A q + u_0^t d^t q \geq u^t b + u_0^t d_0^t \geq \beta.$$

Then $A_{i\cdot}q = b_i$ for all $i \in N^+$ and $d^t q = d_0^t$. In the converse direction, under the assumption,

$$\alpha^\mathsf{T} q = \sum_{i\in N^+} u_i^t A_{i\cdot} q + u_0^t d^t q = \sum_{i\in N^+} u_i^t b_i + u_0^t d_0^t = \beta. \qquad \square$$

Let C^t be the basis cone with respect to the cobasis of q^t, defined by n linearly independent inequalities of P^t that are tight at q^t (say, indexed by set N^t). Let $\mathcal{R}^1$ and $\mathcal{R}^2$ denote the extreme rays of C^1 and C^2.[6] Lemma 3.15 summarizes a few key properties of the basis cones.

Lemma 3.15. *Suppose $\alpha^\mathsf{T} x \geq \beta$ is a valid cut for $\operatorname{conv}(P^1 \cup P^2)$. Let $t \in \{1,2\}$, $q^t \in \arg\min_x\{\alpha^\mathsf{T} x : x \in P^t\}$, and C^t be the basis cone defined by the cobasis associated with q^t. It holds that (1) each ray $r \in \mathcal{R}^t$ satisfies $\alpha^\mathsf{T} r \geq 0$, and (2) $\alpha^\mathsf{T} x \geq \beta$ is valid for C^t. If $\alpha^\mathsf{T} x \geq \beta$ defines a d-dimensional face of P^t, then (3) it also defines a d-dimensional face of C^t, and (4) $\alpha^\mathsf{T} r = 0$ for d extreme rays of C^t.*

Proof. Statement (1) is a consequence of the optimality of q^t over P^t with respect to objective direction α. Namely, the rays of C^t correspond to nonbasic variables and $\alpha^\mathsf{T} r$ is the reduced cost on the related nonbasic variable, which will be nonnegative at optimality. This then implies statement (2), that $\alpha^\mathsf{T} x \geq \beta$ is valid for C^t.

Now suppose $\alpha^\mathsf{T} x \geq \beta$ defines a face of P^t of dimension d. Take $d+1$ affinely independent points, $\{q^t, x^1, \ldots, x^d\}$ from $\{x \in P^t : \alpha^\mathsf{T} x = \beta\}$. Each of these points belongs to C^t because $C^t \supseteq P^t$, certifying (3), that $\alpha^\mathsf{T} x \geq \beta$ defines (at least) a d-dimensional face of C^t.

To prove (4), since C^t is a simplicial cone, all of its proper faces are simplicial cones themselves, so the fact follows from taking the d (or more) rays of the simplicial cone corresponding to the proper face defined by $\alpha^\mathsf{T} x \geq \beta$. $\square$

In the following discussion, we look at ways to further tilt the initial cut. By Lemma 3.15, $\bar{\alpha}^\mathsf{T} x \geq \gamma$ is a valid, though perhaps not facet-defining, inequality for $\operatorname{conv}(C^1 \cup C^2)$. We first propose, in Lemma 3.16, a tilting that, when available, immediately increases the dimension of the face of $\operatorname{conv}(C^1 \cup C^2)$ defined by the tilted cut. This tilting utilizes a well-known complementarity property for any basic feasible solution to (3.2), that $u_i^1 \cdot u_i^2 = 0$ for all $i \in [m]$; it corresponds to finding a ray of C^1 or C^2 to be tight for the new tilted cut.

Lemma 3.16. *Suppose there exists $i^* \in N^1 \cap N^2$ such that $\delta_{i^*} := \min\{\bar{u}_{i^*}^1, \bar{u}_{i^*}^2\} > 0$. Let*

$$\widetilde{\alpha}^\mathsf{T} := \bar{\alpha}^\mathsf{T} - \delta_{i^*} A_{i^*\cdot}, \text{ and}$$
$$\widetilde{\beta} := \gamma - \delta_{i^*} b_{i^*}.$$

Then $\widetilde{\alpha}^\mathsf{T} x \geq \widetilde{\beta}$ is a valid cut for and defines a higher-dimensional face of $\operatorname{conv}(C^1 \cup C^2)$.

Proof. Define corresponding multipliers, for each $t \in \{1,2\}$: $\widetilde{u}_{i^*}^t := \bar{u}_{i^*}^t - \delta_{i^*}$ and $\widetilde{u}_i^t := \bar{u}_i^t$ for all $i \in [0, 1, \ldots, m] \setminus \{i^*\}$. These certify the validity of the new cut for $C^1 \cup C^2$.

[6]These can be read directly from the optimal tableau to $\min_x\{\bar{\alpha}^\mathsf{T} x : x \in P^t\}$, which is already being solved when we optimize over P^t for the best right-hand side for the (original) base inequality. It could be beneficial to instead work with the tangent cone at q^t, but this is not always easy to obtain.

By Lemma 3.14, for any $q \in P^t$ such that $\bar{\alpha}^\mathsf{T} q = \gamma$, it holds that $A_{i\cdot} q = b_i$ for all $i \in [m]$ such that $u_i^t > 0$. Hence,

$$\widetilde{\alpha}^\mathsf{T} q = \bar{\alpha}^\mathsf{T} q - \delta_{i^*} A_{i^*\cdot} q = \gamma - \delta_{i^*} b_{i^*} = \widetilde{\beta}.$$

Thus, the new tilted inequality is tight on all the points of P^t that the original inequality was tight on. In addition, at least one of $\widetilde{u}_{i^*}^1$ and $\widetilde{u}_{i^*}^2$ has value zero. Without loss of generality, say $\widetilde{u}_{i^*}^1 = 0$. Let r denote the (unique) ray of C^1 such that $A_{i^*\cdot} r > 0$. It follows from Lemma 3.14 that $\bar{\alpha}^\mathsf{T} r > 0$ while $\widetilde{\alpha}^\mathsf{T} r = 0$. Moreover, the point $q^t + r$ does not belong to $\{x \in P^t : \alpha^\mathsf{T} x = \beta\}$, so it is an additional affinely independent point satisfying the new cut at equality, proving that the face of $\operatorname{conv}(C^1 \cup C^2)$ defined by $\widetilde{\alpha}^\mathsf{T} x \geq \widetilde{\beta}$ has dimension one higher than the face defined by $\bar{\alpha}^\mathsf{T} x \geq \gamma$. $\square$

We can apply Lemma 3.16 repeatedly, until $u_i^1 \cdot u_i^2 = 0$ for all $i \in [m]$. Each application of this simple tilting increases the number of rays of $C^1 \cup C^2$ that are tight for the tilted cut. Whenever one of these rays contains a point of P^t, the new tilted cut actually defines a higher-dimensional face of the split hull (not just $\operatorname{conv}(C^1 \cup C^2)$). In general, this procedure may stop short of defining a facet of $\operatorname{conv}(C^1 \cup C^2)$, such as when there are more than n variables from u^1 and u^2 that take positive values.

We now consider an alternative way to further tilt. Let $\mathcal{R}^* \subseteq \mathcal{R}^1 \cup \mathcal{R}^2$ denote the rays that are tight for the cut, i.e., $\mathcal{R}^* := \{r \in \mathcal{R}^1 \cup \mathcal{R}^2 : \bar{\alpha}^\mathsf{T} r = 0\}$. The tilting approach of Chvátal et al. [54] involves finding a feasible solution (a *tilting direction*) $(v, w) \in \mathbb{R}^n \times \mathbb{R}$ to

$$\begin{aligned}
(\bar{\alpha} + v)^\mathsf{T} q^1 &= \gamma + w \\
(\bar{\alpha} + v)^\mathsf{T} q^2 &= \gamma + w \\
(\bar{\alpha} + v)^\mathsf{T} r &= 0 \qquad\qquad \text{for all } r \in \mathcal{R}^* \\
(\bar{\alpha} + v)^\mathsf{T} \bar{r} &= 0
\end{aligned}$$

and then tilting in the direction defined by (v, w). This leads to an inequality that is tight on q^1, q^2, all rays in $\mathcal{R}^*$, and on $\bar{r}$. Equivalently, we can find a solution (v, w) to

$$\begin{aligned}
v^\mathsf{T} q^1 &= w \\
v^\mathsf{T} q^2 &= w \\
v^\mathsf{T} r &= 0 \qquad \text{for all } r \in \mathcal{R}^* \\
v^\mathsf{T} \bar{r} &= 0
\end{aligned}$$

A solution to this system can be found (if one exists) via the simplex method.

3.3 Tilting using multiple split disjunctions

In practice, multiple integer-restricted variables take fractional values in $\bar{x}$, leading to multiple possible split disjunctions that can be considered. In this section, we describe ways to apply tilting to multiple split disjunctions. First, we cover a method for generating a cut from multiple disjunctions applied in sequence, which is equivalent to generating a cut from a subspace of the original problem, or lifting a cut valid for a node of a branch-and-bound tree to be valid for the root node. Afterwards, we look at a way to tilt from information from several split disjunctions applied in parallel (as opposed to in sequence).

3.3.1 Tilting cuts from a subspace (sequential tilting)

In the statement of Theorem 3.3, we require $h^\intercal x \geq \beta_t$ to be valid for P^t, $t \in \{1,2\}$. However, with minor modifications of the proof of that result, one can see that the same tilted inequality is actually valid for P_I as long as β_1 and β_2 are such that $h^\intercal x \geq \beta_t$ is valid for $P^t \cap P_I$ (rather than P^t). The resulting tilted cut may no longer be valid for the split disjunction. One way to use this observation is to generate a SIC (or GMIC) in the subspace P^t. The coefficients of the cut can simply be read from the optimal tableau for a solution $p^t \in P^t$, e.g., as the SIC (or GMIC) from any binary variable fractional at the vertex p^t of P^t, making this a low-cost operation.[7]

The more generic extension to the above concept is deriving globally valid cuts from a branch-and-bound tree by a sequence of tilting operations applied recursively. This *sequential tilting* is an analogue to what is called *approximate lifting* in the literature, and it has previously been used in the contexts of binary knapsack sets [65, 92] and primal cutting planes [74] (see also Richard [130] and the references therein). In our setting, we are considering an arbitrary mixed-integer program, rather than a class of problems with special substructure, but the underlying mathematical technique is exactly the same.

Specifically, consider any leaf node of a given branch-and-bound tree in which branching is performed on elementary split disjunctions, and take the path from the root to that leaf node. Say that the leaf node is on level K of the tree, and the optimal solutions to the LP relaxations for the nodes along the path are $x^1, \dots, x^{K+1}$, where $x^1 := \bar{x}$. For ease of exposition but without loss of generality, assume that the leaf node is arrived at by taking the right branch at each level of the tree, and the split variable to get from level $k-1$ to k is x_k. Thus, the LP relaxation at the leaf node is $\{x \in P : x_k \geq \lceil x^k_k \rceil \text{ for all } k \in [K]\}$.

We can derive a valid inequality for the root-level split disjunction $(x_1 \leq \lfloor \bar{x}_1 \rfloor) \vee (x_1 \geq \lceil \bar{x}_1 \rceil)$ as follows. Let $(\alpha^K)^\intercal x \geq \beta^K_2$ be a valid inequality for the the leaf node. To get a cut valid for the parent of the leaf node, we solve the tilting (lifting) problem, by solving for

$$\beta^K_1 := \min_x \{(\alpha^K)^\intercal x : x \in P,\ x_K \leq \lfloor x^K_K \rfloor,\ x_k \geq \lceil x^k_k \rceil \text{ for all } k \in [K-1]\}.$$

Thus, $(\alpha^K)^\intercal x \geq \beta^K_1 + (\beta^K_2 - \beta^K_1)(x_K - \lfloor x^K_K \rfloor)$ is valid for the parent of the leaf node (by Theorem 3.3). We can define this process recursively. For $\ell \in [K-1]$, let

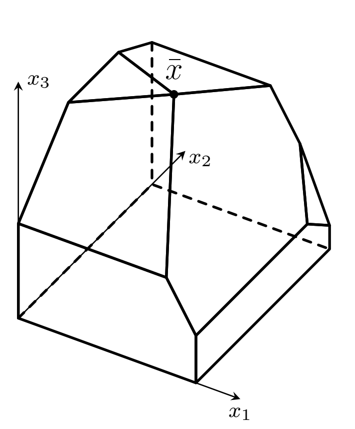
$$\begin{aligned}
\alpha^\ell &:= \alpha^{\ell+1} - (\beta^{\ell+1}_2 - \beta^{\ell+1}_1) e_{\ell+1} \\
\beta^\ell_2 &:= \beta^{\ell+1}_1 - (\beta^{\ell+1}_2 - \beta^{\ell+1}_1) \left\lfloor x^{\ell+1}_{\ell+1} \right\rfloor \\
\beta^\ell_1 &:= \min_x \{(\alpha^\ell)^\intercal x : x \in P,\ x_\ell \leq \lfloor x^\ell_\ell \rfloor,\ x_k \geq \lceil x^k_k \rceil \text{ for all } k \in [\ell-1]\}.
\end{aligned}$$

Then, by induction, the inequality $(\alpha^1)^\intercal x \geq \beta^1_1 + (\beta^1_2 - \beta^1_1)(x_1 - \lfloor \bar{x}_1 \rfloor)$ is valid for the root-level split disjunction.[8] The inequality in expanded form is

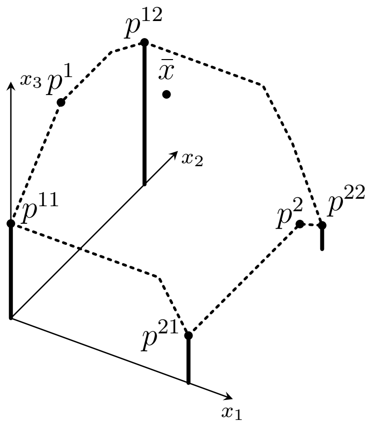
$$(\alpha^K)^\intercal x - \sum_{\ell=1}^{K} (\beta^\ell_2 - \beta^\ell_1) x_\ell \geq \beta^1_1 - (\beta^1_2 - \beta^1_1) \lfloor \bar{x}_1 \rfloor .$$

[7]Generating Gomory cuts merely requires the basis inverse for the optimal solution on each side of the split disjunction, which, as we have already pointed out, is readily available from strong branching.

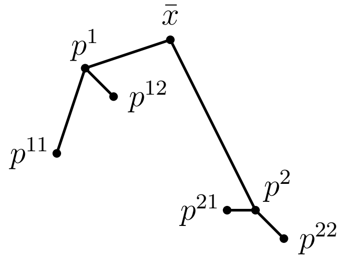
[8]Of course, the inequality that is valid for the root-level split disjunction can be used in strengthened form for internal nodes of the branch-and-bound tree.



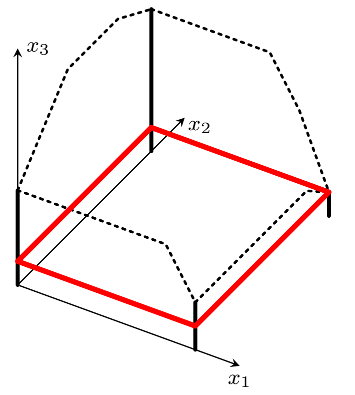
(a) Feasible region of the original linear relaxation.



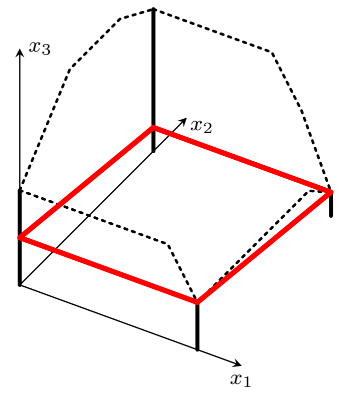
(b) Feasible regions for four leaf nodes of a B&B tree.



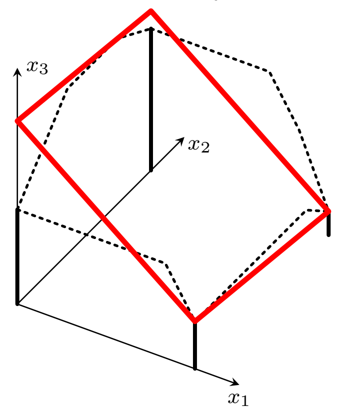
(c) B&B tree, with a node's depth proportional to its objective value.



(d) Cut valid only for the node associated with p^{22}.



(e) Cut valid for the node associated with p^2.



(f) Cut valid for the root node; cuts a ray of basis cone at p^1.

Figure 3.2: Example of sequentially tilting an objective cut using a branch-and-bound tree. The base inequality is $x_3 \leq 5/8$, shown in (d), and it is iteratively tilted, starting at an arbitrary leaf node along the path to the root, until it is valid for the root.

We illustrate this technique for a small example in Figure 3.2.

Returning to the simplest version of this procedure, of tilting valid cuts derived for P^1 or P^2, one issue that becomes readily apparent is that the number of cuts starts growing unmanageably large. There could be n fractional integer variables at $\bar{x}$, and $n-1$ at p^1 and p^2, yielding about $2n^2$ possible tilted cuts, far more than the number of Gomory cuts from elementary split disjunctions based on $\bar{x}$ (at most n cuts). This points to the need for future work in designing intelligent rules for the number and characteristics of the cuts to generate from each elementary split.

A cut produced in the subspace P^t that cuts away p^t is clearly infeasible for $\text{conv}(P^1 \cup P^2)$. This implies that tilted subspace cuts would not be obtainable as L&PCs from the original split disjunction, but rather as cuts that are valid for a disjunction formed from two split disjunctions. We next discuss another way to use multiple split disjunctions.

3.3.2 Mixing inequalities (simultaneous tilting)

In the previous section, we considered a sequential tilting approach as a way of generating a globally valid cut from a valid inequality for a node of a branch-and-bound tree. In this section, we look at split disjunctions that are applied in parallel and how we can generate one stronger tilted cut by sharing information across the splits.

For ease of exposition, we consider only elementary split disjunctions. Let $\sigma := \{j \in \mathcal{I} : \bar{x}_j \notin \mathbb{Z}\}$ and $k \in \sigma$. Define $P_k^1 := \{x \in P : x_k \le \lfloor \bar{x}_k \rfloor\}$ and $P_k^2 := \{x \in P : x_k \ge \lceil \bar{x}_k \rceil\}$, which we assume are nonempty, and set $S_k := \{x \in \mathbb{R}^n : \lfloor \bar{x}_k \rfloor \le x_k \le \lceil \bar{x}_k \rceil\}$. For $t \in \{1, 2\}$, let z_{kt} denote the optimal value of $\min_x\{c^\mathsf{T} x : x \in P_k^t\}$ and set

$$z_0 := \max_{k \in \sigma} \min\{z_{k1}, z_{k2}\}.$$

As a direct corollary of Proposition 3.2, we can state a better bound on the objective value of (LP) after adding cuts from multiple split disjunctions simultaneously.

Corollary 3.17. *The optimal value of* (LP) *after adding the tilted objective cut for each $k \in \sigma$ will be at least z_0.*

Proof. Adding to P the tilted objective cut from any S_k, $k \in \sigma$, results in an objective value of $\min\{z_{k1}, z_{k2}\}$ by Proposition 3.2, so we can guarantee objective value z_0 by adding all the tilted objective cuts. $\square$

Kılınç et al. [108] use Corollary 3.17 to strengthen the cuts derived from Proposition 3.1 via the following valid inequality for P_I, in which $\bar{z}_{kt} := \max\{z_0, z_{kt}\}$:

$$c^\mathsf{T} x \ge \bar{z}_{k1} + (\bar{z}_{k2} - \bar{z}_{k1})(x_k - \lfloor \bar{x}_k \rfloor).$$

This inequality cuts points of $\operatorname{conv}(P_k^1 \cup P_k^2)$ for any $k \in \sigma$ for which $z_0 > \min\{z_{k1}, z_{k2}\}$.

Though the tilted objective inequalities are not parallel to the objective function, they each differ from the objective function in only one coefficient, albeit a different one for each $k \in \sigma$. Using the idea of *mixing* [95, 128] (or equivalently *star inequalities* in the context of mixed vertex packing problems [14]), Kılınç et al. [108] further strengthen the tilted objective cuts for binary problems, and simultaneously derive an inequality that differs in more coefficients from the original objective vector.

In Theorem 3.18, following Günlük and Pochet [95, Theorem 2], we apply mixing to any base inequality, though we give a different proof based on tilting. Note that the method only applies at the bounds of P corresponding to variables in σ. This is automatically satisfied for binary problems.

Theorem 3.18. *Let $h \in \mathbb{R}^n$. For $k \in \sigma$ and $t \in \{1, 2\}$, define*

$$\beta_{kt} := \min_x\{h^\mathsf{T} x : x \in P_k^t\}$$
$$\gamma_0 := \max_{k \in \sigma} \min\{\beta_{k1}, \beta_{k2}\}$$
$$\gamma_k := \max\{\beta_{k1}, \beta_{k2}\}.$$

Suppose there exists $p \in P$ such that $p \in \operatorname{int} S_k$ for all $k \in \sigma$ and $h^\intercal p < \gamma_0$. Assume, without loss of generality, that $\sigma^* \subseteq \sigma$ is numbered sequentially (from 1) such that $\gamma_{k-1} < \gamma_k$ for $1 \le k \le |\sigma^*|$. Let $\sigma_1^* := \{k \in \sigma^* : \beta_{k1} > \beta_{k2}\}$ and $\sigma_2^* := \sigma^* \setminus \sigma_1^*$. Then the following mixing *inequality is valid for P_I*

$$h^\intercal x \ge \gamma_0 + \sum_{k\in\sigma_1^*}(\gamma_k - \gamma_{k-1})(\lceil \bar{x}_k \rceil - x_k) + \sum_{k\in\sigma_2^*}(\gamma_k - \gamma_{k-1})(x_k - \lfloor \bar{x}_k \rfloor). \tag{3.3}$$

if $\{x \in P : x_k < \lfloor \bar{x}_k \rfloor\} = \emptyset$ for all $k \in \sigma_1^*$ and $\{x \in P : x_k > \lceil \bar{x}_k \rceil\} = \emptyset$ for all $k \in \sigma_2^*$.

Proof. For ease of exposition, suppose $\beta_{k1} \le \beta_{k2}$ for all $k \in \sigma^*$. The inequality $h^\intercal x \ge \gamma_0$ is valid for P_I, and specifically $P_1^1 \cap P_I$, by the assumption that $h^\intercal p < \gamma_0$. By Theorem 3.3, the inequality $h^\intercal x \ge \gamma_0 + (\gamma_1 - \gamma_0)(x_1 - \lfloor \bar{x}_1 \rfloor)$ is valid for P_I. Now let $k' := |\sigma|$ and assume, inductively, that the mixing inequality holds for the first $k'-1$ variables from σ^*, so that

$$h^\intercal x \ge \gamma_0 + \sum_{k\in[k'-1]}(\gamma_k - \gamma_{k-1})(x_k - \lfloor \bar{x}_k \rfloor)$$

is valid for P_I. Let

$$\tilde{h} := h - \sum_{k\in[k'-1]}(\gamma_k - \gamma_{k-1})e_k$$

and

$$\tilde{\beta}_1 := \gamma_0 - \sum_{k\in[k'-1]}(\gamma_k - \gamma_{k-1})\lfloor \bar{x}_k \rfloor.$$

Thus, $\tilde{h}^\intercal x \ge \tilde{\beta}_1$ is valid for $P_{k'}^1 \cap P_I$. Moreover, we know that $h^\intercal x \ge \gamma_{k'}$ is valid for $P_{k'}^2$. We assumed that $\{x \in P : x_k > \lceil \bar{x}_k \rceil\} = \emptyset$ for all $k \in \sigma^*$, which implies that

$$\begin{aligned}
\min_{x\in P_{k'}^2} \tilde{h}^\intercal x &= \min_{x\in P_{k'}^2}\left\{h^\intercal x - \sum_{k\in[k'-1]}(\gamma_k - \gamma_{k-1})x_k\right\} \\
&\ge \gamma_{k'} - \max_{x\in P_{k'}^2}\left\{\sum_{k\in[k'-1]}(\gamma_k - \gamma_{k-1})x_k\right\} \\
&\ge \gamma_{k'} - \sum_{k\in[k'-1]}(\gamma_k - \gamma_{k-1})\lceil \bar{x}_k \rceil \\
&=: \tilde{\beta}_2.
\end{aligned}$$

Note that

$$\tilde{\beta}_2 - \tilde{\beta}_1 = \gamma_{k'} - \sum_{k\in[k'-1]}(\gamma_k - \gamma_{k-1})\lceil \bar{x}_k \rceil - \gamma_0 + \sum_{k\in[k'-1]}(\gamma_k - \gamma_{k-1})\lfloor \bar{x}_k \rfloor = \gamma_{k'} - \gamma_{k'-1}.$$

Therefore, by Theorem 3.3, $\tilde{h}^\intercal x \ge \tilde{\beta}_1 + (\tilde{\beta}_2 - \tilde{\beta}_1)(x_{k'} - \lfloor \bar{x}_{k'} \rfloor)$ is valid for P_I (because it is violated by p), which is exactly the mixing inequality (3.3). □

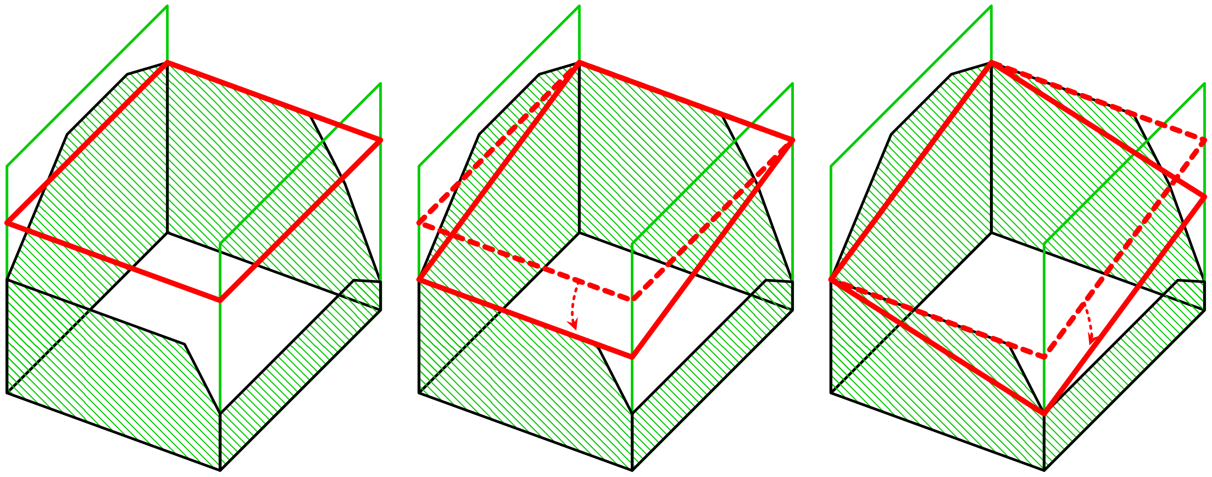


Figure 3.3: Example of tilting an inequality using multiple split disjunctions simultaneously (i.e., without reoptimization, by using only the optimal values on the two sides of each split).

The proof of Theorem 3.18 demonstrates that a mixing inequality (3.3) corresponds to performing a sequence of tiltings, without having to reoptimize to find the new right-hand side every time. This is why we call this method *simultaneous tilting*, as an analogue to simultaneous lifting. An example illustrating the generation of a mixing inequality, viewed as a sequence of tilting operations, is given in Figure 3.3.

One advantage of using mixing is that one base inequality need not be replaced by $|\sigma|$ new tilted cuts, but rather by just one new inequality that aggregates information across all the possible splits. For the binary case, the mixing inequality dominates the tilted inequality that would be derived from the first split disjunction (in the order in σ^*); however, it will not, in general, dominate for the subsequent split disjunctions. We can close some of this gap by adding the last (or last few) unmixed tilted cuts.

The mixing inequality can also be derived for arbitrary split disjunctions. On the other hand, it is not clear whether it can easily be applied to splits not occurring at the variable lower or upper bounds.

3.4 Computational results

In this section, we report preliminary experiments with variants of tilted cuts for different classes of base inequalities. Algorithm 3.5 summarizes the theoretical results contained in Sections 3.2 and 3.3 in an implementable format. We emphasize that our investigation is preliminary; our objective is only to ascertain some limited sense of the strength of tilted cuts that use strong branching information. There are several steps of the algorithm that have not been implemented in our code. First, in steps 12, 22, and 29, we only check if $A_{i\cdot}x \geq b_i$ is tight at the corresponding point (without doing the ray test, which we perceive as too expensive). From the strengthening procedures of Section 3.2.7, we only implement the idea of tilting Gomory cuts generated in the subspace. The steps to perform these missing procedures are still included in the algorithm description for completeness.

With that caveat, our goal is to evaluate the percent integrality gap that can be closed

Algorithm 3.5 Tilted Cuts

```
Input: Polyhedron $P$; objective direction $c$; optimal solution to (LP) $\bar{x}$; integer-restricted
    variables fractional at $\bar{x}$, $\sigma \subseteq \mathcal{I}$.
 1: function TiltedCuts($P, c, \bar{x}, \sigma$)
 2:     $\mathcal{C} \leftarrow \emptyset$.                                  ▷ Initialize pool of cuts.
 3:     for $i \in [m]$ do
 4:         $\beta^i_{kt} \leftarrow b_i$ for $k \in \sigma$ and $t \in \{1,2\}$.          ▷ Initialize tilting coefficients.
 5:     for $k \in \sigma$ do
 6:         For $t \in \{1,2\}$, let $p^{kt} \in \arg\min_x\{c^\intercal x : x \in P^t_k\}$ and $C^{kt}$ the associated basis cone.
 7:         $\beta^0_{kt} \leftarrow c^\intercal p^{kt}$ for $t \in \{1,2\}$.
 8:         $\mathcal{C} \leftarrow \mathcal{C} \cup \{c^\intercal x \geq \beta^0_{k1} + (\beta^0_{k2} - \beta^0_{k1})(x_k - \lfloor \bar{x}_k \rfloor)\}$.       ▷ Tilted objective cut.
 9:         Choose $M^k \subseteq [m]$.               ▷ E.g., only those tight at $p^{k1}$ or $p^{k2}$, or all $[m]$.
10:         $M^{kt} \leftarrow M^k$ for $t \in \{1,2\}$.            ▷ Track which indices remain for each side.
11:         for $t \in \{1,2\}$ and $i \in M^{kt}$ do
12:             if $A_{i\cdot}p^{kt} = b_i$ (or $A_{i\cdot}r \geq 0$ for all rays $r$ of $C^{kt}$) then
13:                 $\beta^i_{kt} \leftarrow A_{i\cdot}p^{kt}$ and remove $i$ from $M^{kt}$.
14:         for $i \in M^k$ do
15:             if $A_{i\cdot}p^{k1} = b_i$ and $A_{i\cdot}p^{k2} = b_i$ then ▷ Inequality cannot be tilted for this split.
16:                 Remove $i$ from $M^{k1}$ and $M^{k2}$ and return to step 14.
17:             if $i \in M^{k1}$ then
18:                 Remove $i$ from $M^{k1}$.
19:                 if $A_{i\cdot}p^{k1} > b_i$ then
20:                     Let $q \in \arg\min_x\{A_{i\cdot}x : x \in P^1_k\}$ and let $C$ be the associated basis cone.
21:                     $\beta^i_{k1} \leftarrow A_{i\cdot}q$.
22:                     for $i' \in M^{k1}$ such that $A_{i'\cdot}q = b_{i'}$ (or $A_{i'\cdot}r \geq 0$ for all rays $r$ of $C$) do
23:                         $\beta^{i'}_{k1} \leftarrow A_{i'\cdot}q$ and remove $i'$ from $M^{k1}$.
24:             if $i \in M^{k2}$ then
25:                 Remove $i$ from $M^{k2}$.
26:                 if $A_{i\cdot}p^{k2} > b_i$ then
27:                     Let $q \in \arg\min_x\{A_{i\cdot}x : x \in P^2_k\}$ and let $C$ be the associated basis cone.
28:                     $\beta^i_{k2} \leftarrow A_{i\cdot}q$.
29:                     for $i' \in M^{k2}$ such that $A_{i'\cdot}q = b_{i'}$ (or $A_{i'\cdot}r \geq 0$ for all rays $r$ of $C$) do
30:                         $\beta^{i'}_{k2} \leftarrow A_{i'\cdot}q$ and remove $i'$ from $M^{k2}$.
31:             $\mathcal{C} \leftarrow \mathcal{C} \cup \{A_{i\cdot}x \geq \beta^i_{k1} + (\beta^i_{k2} - \beta^i_{k1})(x_k - \lfloor \bar{x}_k \rfloor)\}$.
32:         for $t \in \{1,2\}$, $k' \in \mathcal{I}$ such that $p^{kt}_{k'} \notin \mathbb{Z}$ do                    ▷ Optional.
33:             Generate subspace cut from $\{x \in P^t_k : x_{k'} \leq \lfloor p^{kt}_{k'} \rfloor\} \vee \{x \in P^t_k : x_{k'} \geq \lceil p^{kt}_{k'} \rceil\}$.
34:             Tilt subspace cut then add resulting tilted inequality to $\mathcal{C}$.
35:     Generate mixing inequality for each $i \in [m]$ using $\{\beta^i_{kt}\}_{k\in\sigma, t\in\{1,2\}}$ and replace the
            corresponding cuts in $\mathcal{C}$. Optionally, also replace constraint $i$ of $P$.   ▷ Optional.
36:     Apply monoidal strengthening to all tilted cuts in $\mathcal{C}$.                  ▷ Optional.
37:     Perform further tilting as in Section 3.2.7.                              ▷ Optional.
38:     return Set of tilted cuts $\mathcal{C}$ and updated formulation for $P$.
39: end function
```

Table 3.1: Results for experiments with tilted cuts as in Algorithm 3.5.

	GMIC	T0	T1	T1M	T1*	T1*M	T*1K	T*10K
Avg % gap closed	22.9	31.9	35.3	35.4	35.1	35.2	35.8	37.7
Avg time (s)		0.3	75.0	75.1	10.0	10.0	1.1	7.4
Avg # cuts / $\lvert\sigma\rvert$	1.0	1.0	14.3	4.1	1.9	0.7	16.3	26.0

by tilted cuts, when compared to a baseline of GMICs. We select 42 instances from MIPLIB (all versions) based on the criteria that they contain no more than 500 rows or 500 columns, have a known IP optimal value, and have nonzero gap closed by split closure experiments [35]. All code is implemented in C++, compiled with `g++ (GCC) 4.8.3 20140911 (Red Hat 4.8.3-7)`, and uses `Clp version 1.16.10` as the linear programming solver.[9]

We first compare three classes of base inequalities: the objective vector (we call these type 0, or T0, tilted cuts), all hyperplanes defining P (type 1, or T1), and only those hyperplanes defining P that are tight at an optimal solution on one or the other side of the elementary split disjunction (type 1*, or T1*). We also employ mixing to reduce the number of generated cuts; note that mixing is essentially free computationally, once bounds have been computed—the only requirement is sorting the bounds in increasing order. The variants of T1 and T1* with mixing are denoted T1M and T1*M. Lastly, we tilt unstrengthened Gomory cuts in the subspace, as discussed in Section 3.3. As discussed previously, there are unmanageably many such possible cuts, so we set a limit of either $1,000$ or $10,000$ tilted subspace cuts; we call these procedures T*1K and T*10K, respectively.

In Table 3.1, we report the average percent integrality gap closed by these tilted cuts, the average time to generate all of the cuts per instance, and average number of cuts generated as a fraction of the number of fractional variables at $\bar{x}$ (which is the number of GMICs that can be produced from elementary split disjunctions). The baseline for comparison will be the percent of the integrality gap closed by GMICs. All percent gap closed numbers reflect tilted cuts added together with GMICs (in order to measure the *additional* effect that using tilted cuts would have). The column headings reflect the type of tilted cuts being used, with labels as previously defined. The full set of results for percent gap closed are given in Table 3.2, where we also include the sizes of the instances tested. For the `stein` instances, we make two modifications with respect to their online version, to make them less symmetric. First, we replace the objective function $\sum_{i\in[n]} x_i$ with $\sum_{i\in[n]} ix_i$, and second, we remove an additional cardinality constraint (which is also a lower bound on the original objective value), as this is not present in the original formulation of these instances [86].

The results show that tilted objective cuts (T0) are fast to generate (0.3 seconds in total, on average across the instances) and close roughly 9% additional gap, on average, over GMICs on their own. Tilting the hyperplanes defining P (procedure T1) yields an additional 3.4% gap closed, on average. However, tilting all the hyperplanes defining P generates a very large number of cuts. At this point, we employ mixing (as procedure T1M) to reduce the number of cuts: after applying mixing, we have a single tilted inequality from each base

[9]Note that an earlier version of `Clp` results in a different optimal basis being selected for some problems (for example, with instance `misc02`), which would lead to different results.

Table 3.2: Instance size and percent gap closed for tilting experiments.

			% gap closed							
	Rows	Cols	GMIC	T0	T1	T1M	T1*	T1*M	T*1K	T*10K
bell3a	123	133	45.1	61.5	64.0	62.1	64.0	61.6	64.6	64.6
bell3b	123	133	44.6	90.7	90.7	96.6	90.7	90.7	91.8	91.8
bell4	105	117	23.4	91.6	91.6	91.6	91.6	91.6	92.2	92.2
bell5	91	104	14.5	85.4	85.4	85.4	85.4	85.4	86.4	86.4
blend2	274	353	16.3	20.9	20.9	20.9	20.9	20.9	20.1	20.1
bm23	20	27	16.8	17.2	17.2	16.8	17.2	16.8	26.2	26.2
egout	98	141	40.3	51.6	56.3	56.3	56.3	56.3	76.2	76.2
flugpl	18	18	11.7	11.7	11.7	11.7	11.7	11.7	11.7	11.7
gt2	29	188	91.9	91.9	91.9	91.9	91.9	91.9	91.9	91.9
go19	441	441	2.0	3.1	3.1	3.2	3.1	3.2	3.0	4.2
k16x240	256	480	11.4	11.4	11.4	11.4	11.4	11.4	25.5	25.5
lseu	28	89	55.2	55.2	55.2	55.2	55.2	55.2	55.8	55.8
mas284	68	151	0.9	4.2	4.2	5.2	4.2	5.2	3.1	3.1
mas74	13	151	6.7	6.7	6.7	6.7	6.7	6.7	8.4	8.4
mas76	12	151	6.4	6.4	6.4	6.4	6.4	6.4	8.1	8.1
misc02	39	59	3.7	4.9	5.1	5.1	5.1	5.1	5.1	5.1
misc03	96	160	8.6	8.6	9.0	8.6	9.0	8.6	8.6	8.6
misc05	300	136	23.7	23.7	23.7	23.7	23.7	23.7	26.9	26.9
misc07	212	260	0.7	1.2	1.2	0.7	1.2	0.7	0.7	0.7
mod008	6	319	20.1	20.1	20.1	20.1	20.1	20.1	20.8	20.8
mod013	62	96	4.4	7.4	7.4	5.7	7.4	5.7	26.2	26.2
modglob	291	422	17.4	17.6	17.6	18.0	17.6	18.0	29.3	29.3
p0033	16	33	56.8	56.8	56.8	56.8	56.8	56.8	57.1	57.1
p0040	23	40	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
p0201	133	201	25.2	25.2	25.2	25.2	25.2	25.2	46.4	46.4
p0282	241	282	3.7	40.3	85.7	85.7	81.3	81.3	17.2	17.3
p0291	252	291	27.8	29.6	69.5	69.5	69.4	69.4	47.9	47.9
pipex	25	48	28.4	28.4	28.4	28.4	28.4	28.4	30.3	30.3
pp08a	136	240	51.4	51.4	53.0	53.0	53.0	53.0	57.1	78.9
pp08aCUTS	246	240	28.5	28.5	28.5	28.5	28.5	28.5	46.3	52.2
probportfolio	302	320	25.1	25.1	25.6	27.4	25.6	27.4	26.8	34.1
prod1	208	250	0.0	40.9	64.3	64.3	63.3	63.5	9.1	40.9
rgn	24	180	9.7	9.7	9.7	9.7	9.7	9.7	17.9	17.9
rlp2	68	451	0.6	1.9	3.2	3.1	3.2	3.1	2.7	2.7
sample2	45	67	5.9	13.0	13.0	16.0	13.0	16.0	38.2	38.2
sentoy	30	60	19.3	19.3	19.3	19.3	19.3	19.3	21.9	21.9
stein15*	35	15	50.0	64.6	64.6	65.6	64.6	65.6	63.8	63.8
stein27*	117	27	7.4	32.0	32.0	35.4	32.0	35.4	33.7	33.9
stein45*	330	45	7.1	27.7	27.7	23.7	27.7	23.7	27.8	31.4
timtab1	171	397	23.7	23.7	31.4	28.5	31.1	31.8	26.4	34.2
vpm1	234	378	10.0	10.0	11.1	11.1	11.1	11.1	14.8	14.8
vpm2	234	378	15.4	17.4	32.3	33.4	32.2	33.3	34.0	37.3
Average			22.9	31.9	35.3	35.4	35.1	35.2	35.8	37.7

inequality. The result is drastic: we decrease the number of generated tilted cuts from 14 times the number of possible GMICs to merely 4. Interestingly, the gap closed by the mixed inequalities actually increases on average (due to increases for 9 of the 42 instances tested). However, generating the T1 and T1M cuts takes far too much time (over a minute, on average, per instance, which is slower than many of these instances would take to solve to optimality using a commercial code).

The next improvement we consider is only tilting a subset of the inequalities defining P, procedure T1*. Intuitively, inequalities that are tight on one or the other side of the split disjunction may be the most important ones in terms of affecting the objective value. The results reflect this intuition: the average percent gap closed by using only these cuts is 35.1%, which is only 0.2% smaller than the gap closed using all of the hyperplanes defining P. Moreover, the number of cuts generated is drastically reduced: only 1.9 times as many cuts, on average, as there are fractional variables at $\bar{x}$. We can further reduce this number by mixing, which leads to *fewer* cuts than the number of GMICs. Though we do also make progress on time (taking ten seconds on average per instance for cut generation), given the small size of these instances, the procedure is still far from practical.

Lastly, we test only one of the strengthening procedures from Section 3.2.7 in these early experiments: tilting subspace SICs, i.e. unstrengthened Gomory cuts generated from the optimal solution to one side of a split disjunction, which are then tilted to be valid for the other side of the split disjunction. The other strengthening approaches (tilting based on the GMIC disjunction, applying modularization, and further tilting based on Farkas multipliers) have not yet been implemented in our code. Note that tilting is also an easy way to strengthen other cutting planes when they are not supporting for the split hull, but we do not test them in this context. The results are peculiar, in the sense that T*1K and T*10K lead to stronger cuts (which is understandable, as the base inequality here is stronger, and also more cuts are produced), but they are also faster. For example, on average, the cuts from procedure T*1K close 35.8% of the integrality gap while taking only 1.1 seconds per instance. We can close about 2% more gap using a cut limit of $10,000$ with T*10K, but the result is an enormous number of cuts, which is impractical; on the other hand, we do not employ intelligent rules for choosing which Gomory cuts to tilt, leaving open the question of developing effective heuristics for selecting subspace Gomory cuts.

Our computational experiments are preliminary, but they suggest that using strong branching (probing) information may be extremely worthwhile, either to generate cuts or replace (strengthen) rows of the original formulation. As future work, we hope to integrate Algorithm 3.5 into the code of an open-source solver, which will facilitate more extensive computational testing in a practical setting. The motivation for making the next step a solver implementation is the ubiquity of tilting-based operations—it is not immediately clear how many solvers already have similar techniques in their cut generation or presolve codes, without perceiving the techniques from a unified mathematical viewpoint, which we contribute. To this end, we next discuss some of the related literature to the tilting idea of this chapter.

3.5 Existing literature related to tilting

We end this chapter with a discussion of the existing approaches, many of which have already been mentioned, that are akin to tilting. What we define as tilting is not formally specified, but rather corresponds to an informal operation that takes as an input one inequality and outputs a different inequality that depends somehow on the first. Being conceptually simple, the operation of tilting a hyperplane has been applied in a variety of contexts and under a myriad of different names (in fact, we have already used the concept in this dissertation, in Chapter 2). Hence, it is useful to put the tilting technique, as it is specifically used in this chapter, in context. One difference we state outright is that most of the methods in the literature focus on the special case of binary problems and elementary split disjunctions, whereas we present tilting as a tool that can be applied for mixed-integer programs and arbitrary split disjunctions. The typical application is likely to be to elementary split disjunctions, but our presentation applies in the more general context, which has been a topic of interest in recent years (e.g., when branching using strengthened disjunctions, as in [63, 104]).

Coefficient strengthening and lifting. Tilting on elementary split disjunctions, without using strengthening or mixing, is (nearly) equivalent to research streams on a presolve technique called *coefficient strengthening* and on cutting planes obtained by *lifting*.[10] Both of these techniques take a base inequality $h^\intercal x \geq b_h$ and replace h_k, $k \in \mathcal{I}$, by a "stronger" coefficient such that the resulting cut is valid for P_I. The primary difference between the two methods is that coefficient strengthening is typically implemented during presolve to strengthen a given formulation, while lifting is applied during cut generation to increase the dimension of a valid inequality generated from a subspace. Thus, the distinction between the two methods is the origin of the base inequality, but both can be interpreted as applying *probing*, in which implications are derived from tentatively fixing variables to some value [101, 131].[11] There is one more practical difference, which we ignore, that coefficient strengthening requires the new inequality to dominate the base inequality, which is why the technique is only applied at the bounds of the variable whose coefficient is being strengthened (which satisfies the conditions for dominance in Proposition 3.7).

In contrast to our setting, in most of the literature, coefficient strengthening and lifting are investigated for the specific case of pure or mixed binary problems, and even more so for specially-structured constraints. For example, coefficient strengthening was first applied to knapsack constraints [50, 76, 107], though it was later extended to other constraint types [77, 78]. The history is similar for lifting, where the first applications were to set packing [119, 123, 124] and knapsack problems [20], before Wolsey [137] outlined lifting more generically, though still in the context of a one-row relaxation.

More recently, coefficient strengthening was presented in a more general setting and evaluated for mixed binary problems by Andersen and Pochet [10], who test strengthening the coefficients on binary variables in constraints defining P (though in their tests, this strength-

[10]The equivalence of lifting and tilting is also highlighted by Espinoza et al. [80].

[11]Presolve refers to improving the representation of an instance. There are two types of presolve: preprocessing methods that derive implications prior to solving the first LP relaxation, and optimization-based approaches. The latter category, which includes probing, is significantly more expensive and more naturally pairs with cutting plane generation when making decisions about allocating computational resources.

ening is applied recursively) as well as for Gomory cuts derived from $\bar{x}$. In parallel, Dey and Richard [74] explored lifting as a way of preprocessing cuts for mixed-integer programs. The technique in this latter paper is tested only for a modified primal cutting plane method, but it is conceptually identical to tilting as in this chapter. Notably, unlike much of the lifting literature, they consider the full set of constraints of P when computing lifting coefficients, as opposed to a one-row relaxation. They do not consider improvements such as strengthening or mixing, and they do not evaluate the method for generic coefficient strengthening. One way to view our work is as a first step to investigating the open questions discussed in the conclusion of [74] via a more computationally efficient implementation of the idea tested in [10]. To the best of our knowledge, the tilting we test in this chapter is not applied in its generic form in commercial codes. See, for instance, the sections on coefficient reduction and lifting in the recent survey of presolve methods in `Gurobi` by Achterberg et al. [3].

Applying tilting to Gomory cuts derived from $\bar{x}$ has previously been investigated [10, 74, 107], but we have not found examples testing the generation of a Gomory cut from a subspace and then tilting it to be globally valid, as we do in our computational experiments. We also have not found the ideas of modularization, mixing, or selecting base inequalities by probing (strong branching) in either the lifting or coefficient strengthening literature. For more references for coefficient strengthening and lifting, the interested reader can consult modern surveys of presolve and lifting techniques [3, 130]. See also the paper of Gu et al. [93] for a review of sequential lifting methods.

Other tilting methods. We next mention two papers concerning more general tilting procedures that deal with generating valid inequalities from arbitrary disjunctions. In contrast to these two papers, the tilting procedure we define is simpler because its scope is less broad: we focus on split disjunctions in order to strengthen a given valid inequality (such as a constraint defining P). As discussed in Section 3.2.7, our tilted cuts can be viewed as a starting point for the more expensive tilting procedures of [54, 126].

In Chapter 4 of his Ph.D. dissertation, Perregaard [126] outlines a recursive procedure for producing a collection of tilted cuts from an arbitrary valid disjunction, with the goal of capturing some of the information from a branch-and-bound tree via a pool of cuts that implies the *disjunctive lower bound*, the best objective value that could be obtained from the disjunction (by optimizing over each term). At each step of that procedure, there is a collection of inequalities that are valid for the disjunctive hull, and a collection of affinely independent points from the disjunctive hull are kept per each inequality (certifying a lower bound on the dimension of the face defined by that inequality). One base inequality (that is not facet-defining for the disjunctive hull) is selected from the collection. A second, invalid, inequality is then identified that is tight for the same points of the disjunctive hull as the base inequality. Subsequently, these two inequalities are combined (by tilting) to yield two new cuts that together dominate the base inequality and each define a higher-dimensional face of the disjunctive hull. To achieve the goal of capturing the dual bound from a partial branch-and-bound tree, the procedure initializes the cut collection with the objective cut implied by the disjunctive lower bound.

A similar tilting method was explored, more formally and with accompanying computational experiments, by Chvátal et al. [54], inspired by local cuts for the traveling salesman problem [12]. The goal of this paper is to generate strong facet-defining inequalities (without necessarily capturing the disjunctive lower bound). One modification, with respect to the just-mentioned algorithm of Perregaard, is that the local cut tilting works in a subspace of the original problem, generating facet-defining inequalities for some restriction of P_I, which are then lifted to be valid for the original space.

Although neither paper explicitly makes a connection to strong branching, in Section 4.7 of his dissertation, Perregaard discusses a "fast approximation" that uses only the constraints from each disjunctive term that are nonbasic at the optimal solution for that term. Our tilting algorithm in essence uses this same relaxation, but for a different purpose, generating weaker cuts but doing so more computationally efficiently.

There are other methods in the literature that are essentially tilting by another name, but a comprehensive review is outside the scope of this chapter. Nevertheless, we mention the relevant techniques of sequential pairing of inequalities [94] and the tilted cuts of Balas [17, Section 9].

Mixing. The last connection we make is to the literature on mixing. As we previously stated, and is readily apparent from the proof of the mixing inequality in Theorem 3.18, mixing *is* tilting. The idea of mixing arose independently as a way of combining valid inequalities (in [95]) and from extending conflict graphs for mixed-binary programming [13, 14]. This latter connection is notable in that conflict graphs are themselves used for preprocessing, including for strengthening coefficients. However, while the conflict graph is only used for binary programs, we apply tilting to general mixed-integer problems.

3.6 Conclusion & open problems

The main contributions of this chapter are (1) introducing a generic framework for tilting-based approaches to generating and strengthening valid inequalities for mixed-integer programs, (2) proposing an efficient implementation of tilting that uses strong branching information readily available in solvers, (3) incorporating the concept of tilting on several split disjunctions at once when generating a tilted inequality, (4) describing preliminary computational experiments within our tilting framework that allude to its potential for practical impact, and (5) surveying existing literature on methods that involve tilting, such as lifting and coefficient strengthening.

Our theoretical treatment of tilting is self-contained and aggregates several techniques that previously existed independently in the literature. The review of the literature in this chapter is surely not comprehensive, and other preprocessing, strengthening, and cut generation methods are likely to also be geometrically equivalent to tilting on split disjunctions. Nevertheless, the takeaway is that it is important to understand when a technique reduces to tilting, in order to leverage the various theoretical and practical insights that have been derived from studying related concepts under different names.

While we stress that the computational results are preliminary, they indicate that there is substantial strength to be gained from tilting constraints defining a given relaxation, as well as tilting Gomory cuts derived from one side of a split disjunction. Both of these base inequalities make use of strong branching, which potentially provides new ways to offset the cost of that expensive operation. In addition, the idea of mixing dramatically reduces the number of cuts that are generated, which is helpful for further decision-making during branch-and-bound. Finally, we outline, but have not yet evaluated, strengthening approaches, including using Farkas multipliers (again, readily available from strong branching) for modularizing the coefficients of the tilted cut and for further tilting.

There are several questions that merit further computational investigation. First, when tilting is applied to a base inequality coming from the constraints of P, we can either replace the original inequality in the formulation, or we can add the tilted cut with the rest of the pool of cuts, and it is not clear when to do which. The latter option could make sense if we wish to avoid making the constraint matrix more dense (see also the related remarks in [3, Section 7.2.2]), but this necessitates extensive empirical evaluation. A second generic question is when should the solver expend effort on producing shallow, but computationally inexpensive, tilted cuts as compared to deeper cuts, such as general disjunctive cuts? This second question closely ties in with recurring comments about the practical limitations of strong branching and probing, that the information from these techniques is often useful but too expensive to gather; see, e.g., the discussion in [101]. One way to reduce that computational burden is utilizing solver information in as many ways as possible, which is one reason for our focus in this chapter on strong branching. Lastly, we mention that it would be natural to attempt extending the tilted cuts in this chapter to nonlinear problems, similarly to how Kılınç et al. [108] evaluated the tilted objective cuts in a nonlinear context.

Chapter 4

$\mathcal{V}$-Polyhedral Cuts

This chapter is based on joint work with Egon Balas.

4.1 Introduction

As a reminder of notation, P_I is the feasible region of the integer programming problem we are solving, (IP); P is the feasible region of (LP), the linear programming relaxation of (IP); $\bar{x}$ is an optimal solution to (LP); and $\mathcal{N}$ is the index set of nonbasic variables at $\bar{x}$. The *basis cone* $C(\mathcal{N})$ is the (simple and simplicial) cone with apex at $\bar{x}$ defined by the n inequalities indexed by $\mathcal{N}$. Given a valid disjunction, $\vee_{t \in \mathcal{T}}(D^t x \geq D^t_0)$, $P^t := \{x \in P : D^t x \geq D^t_0\}$ is the feasible region for *disjunctive term* $t \in \mathcal{T}$ and $P_D := \operatorname{conv}(\cup_{t \in \mathcal{T}} P^t)$ is the *disjunctive hull.*

This chapter presents theoretical and experimental results for a framework, which we call *$\mathcal{V}$-polyhedral cuts* (VPCs), for generating cuts from general valid disjunctions. Our approach is motivated by and can be viewed as a natural extension of *generalized intersection cuts* (GICs) [31], for which we presented computational results in Chapter 2 but required simple disjunctions (equivalently, convex P_I-free cut-generating sets). However, VPCs can also be viewed independently, as an efficient alternative to *lift-and-project cuts* (L&PCs) [24, 25], which were discussed in Section 1.4. The advantage of developing this methodology is that it enables practical testing of cut generation from much stronger disjunctions than those typically used in practice, which is the focus of this chapter. We evaluate our cuts both for strength, by measuring the gap closed when they are applied to the LP relaxation feasible region P, and effectiveness as part of branch-and-cut. The results show that VPCs substantially improve the root gap closed as compared to *Gomory mixed-integer cuts* (GMICs), while the branch-and-cut experiments merit further research and refinement of the implementation but already suggest the procedure can have significant practical relevance.

The VPC framework allows us to successfully generate a large number of cuts without resorting to recursion. As mentioned in Section 1, the motivation for pursuing such a procedure is avoiding numerical issues and "tailing off" that occurs when Gomory cuts are applied based on rounds of previously-generated cuts, a well-known phenomenon in the integer programming community (for example, see the discussion of [40, Table 1]). The cuts in this chapter are all produced from the initial relaxation P, not derived from other cutting planes, reducing the potential for compounding computational inaccuracies, and indeed the VPCs in our experiments appear numerically stable. Moreover, our procedure is able to generate arbitrarily strong valid inequalities (up to the limit of reaching the facets defining P_I) by using increasingly strong disjunctions (again, we emphasize that this is without recursion).

In practice, non-simple disjunctions are derived from the integrality constraints on several variables and are closely related to the process of branching on variable bounds. For example, cross cuts come from the union of two elementary split disjunctions, which is a special case of t-branch split cuts studied by Li and Richard [110]; each cross disjunction is equivalent to taking the four-term disjunction formed from the leaf nodes of a two-level branch-and-bound tree. In general, the set of leaf nodes at an arbitrary stage of the branch-and-bound process defines a valid disjunction. The disjunctions we test in this chapter precisely arise from the leaf nodes of a partial branch-and-bound tree. One reason we focus on this experimental setup is to develop a better understanding of the interaction between cuts and branch-and-bound. We echo the following sentiment by Lodi [111]:

> Overall, it seems that a tighter coordination of the two most fundamental ingredients of the MIP solvers, branching and cutting, can lead to strong improvements.

Typically cuts are not generated at nodes of a branch-and-bound tree much deeper than the root node as this would require either additional overhead for maintaining cuts at each node or a procedure that ensures the cuts are globally valid. Both of these options are expensive and indicate why cuts are usually only separated at or near the root node. Padberg and Rinaldi [125] state that using "classical cutting planes and tree-search is out of the question as far as the solution of large-scale combinatorial optimization problems is concerned" because of the additional computational cost involved. Nonetheless, some general-purpose cutting planes do theoretically work when derived from arbitrary nodes. Gomory cuts from a particular node can be valid for the whole tree when some specific conditions are satisfied [26]; see also the discussion of finding globally valid cuts in [122]. Our approach is different in that we do not just look at a single node but rather the set of leaf nodes of the partial branch-and-bound tree; the observation that this set forms a disjunction implies that we can apply the full technology of disjunctive programming.

Multi-term disjunctive cuts coming from partial branch-and-bound trees have previously been proposed and tested in different contexts. One stream of research tests them (using L&PCs) for stochastic mixed-integer programming [87, 129, 132, 138]. The drawback of L&PCs is that generating a single cut involves optimizing the higher-dimensional *cut-generating linear program* (CGLP). Perregaard and Balas [127] present results on a relative of VPCs that requires row generation to guarantee cut validity and compare the time to generate cuts from multi-term disjunctions using their approach instead of solving the larger (CGLP). The authors conclude that solving the CGLP becomes much slower as the number of terms of the disjunction grows. This is relevant as VPCs use the same kind of linear program as in [127] but produce valid cuts immediately, which should make them only more computationally efficient. The row generation idea from [127] was recently revisited by Louveaux et al. [114] to analyze the strength of intersection cuts from two-row relaxations. In combination with results from [113], they conclude that the gap closed by cuts from two-row relaxations can be significant compared to Gomory cuts, though most of this strength comes from split sets that can be derived from the two-row relaxations. Chvátal et al. [54] experimented with partial branch-and-bound trees as part of local or target cut algorithms [11, 51], which are dual to the separation schemes in this paper and that of [114, 127]. Computational tests of L&PCs generated from multi-term disjunctions obtained via partial branch-and-bound trees were also performed in the context of the cutting plane tree algorithm [52, 53] and as part of studies elaborating on the connection between disjunctive cuts and the Reformulation-Linearization Technique [134, 135]. Finally, we mention the computational experience of solving the *seymour* problem documented by Ferris et al. [82], in which (globally valid) L&PCs were generated from partial branch-and-bound trees with 16 leaf nodes and with 256 leaf nodes to eventually yield 256 subproblems that were later solved to optimality individually.

Traditional general-purpose cutting planes focus mainly on simultaneously using multiple, typically simple, disjunctions. In contrast, we propose a different philosophy, which is to generate a large number of deep cuts from one strong non-simple disjunction instead. The existing literature demonstrates that an effective cut separator for general disjunctions may have great potential benefits. We build on this previous work by designing a fast cut generator that enables us to test large disjunctions. We describe properties of our cut-generation scheme in Section 4.2, such as conditions under which we obtain facets of the disjunctive hull and theoretical results that benefit our implementation. Section 4.3 provides the setup of our computational experiments, the results of which are contained in Section 4.4. First, we validate the existence of strong cuts from our framework. This leads us to also evaluate the cuts as part of branch-and-bound. The results of these experiments indicate that a vital and challenging question that remains outstanding is which particular multi-term disjunctions lead to good cuts. Then, in Section 4.5, we compare our approach of using one large disjunction with that of using a collection of shallow disjunctions. Finally, drawing from our computational experience, we discuss in Section 4.6 some of the considerations for future, improved implementations of VPCs.

4.2 Point-ray linear program

Let $\mathcal{P}$ and $\mathcal{R}$ denote sets of points and rays in $\mathbb{R}^n$. Define the *point-ray linear program* (PRLP), taking the point-ray collection $(\mathcal{P}, \mathcal{R})$ and an objective direction $w \in \mathbb{R}^n$ as an input, as follows:

$$\begin{aligned} \min_{\alpha,\beta} \quad & \alpha^\mathsf{T} w \\ & \alpha^\mathsf{T} p \geq \beta && \text{for all } p \in \mathcal{P} \\ & \alpha^\mathsf{T} r \geq 0 && \text{for all } r \in \mathcal{R}. \end{aligned} \tag{PRLP}$$

The feasible solutions (α, β) to (PRLP) correspond to inequalities $\alpha^\mathsf{T} x \geq \beta$ that we call *$\mathcal{V}$-polyhedral cuts* (VPCs). Define the *point-ray hull* as $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$.

We first detail properties of the feasible region of (PRLP). In particular, we prove necessary and sufficient conditions for VPCs to be valid cuts for P_I, and we discuss the conditions under which VPCs are facet-defining for the disjunctive hull. We then analyze the theoretical strength of the cuts, part of which drive our choices of objective functions w for (PRLP). Finally, we present Algorithm 4.6, which shows our implementation for finding a set of points and rays and generating a set of cuts from that point-ray collection via (PRLP).

4.2.1 Globally proper point-ray collections

The PRLP is analogous to the linear program (2.2) used for GICs. The primary difference is the source of the points and rays, which may no longer correspond to the intersection of edges of P (or some relaxation of it) with the boundary of a P_I-free convex set. To transition to the setting of this chapter, we adapt Definition 2.2 of a *proper* point-ray collection from Chapter 2.

Definition 4.1. *The point-ray collection* $(\mathcal{P}, \mathcal{R})$ *is called* globally proper *if* $\alpha^{\mathsf{T}} x \geq \beta$ *is valid for* P_I *whenever* (α, β) *is feasible to* (PRLP).

Definition 2.2 requires that feasible solutions of (PRLP) correspond to valid inequalities for P_I only when they cut some point of the connected component of $P \setminus P_D$ containing $\bar{x}$. Thus, there is a dependence in the definition on P_D and on $\bar{x}$, both of which are removed in Definition 4.1. Consider a *standard intersection cut* (SIC), such as the one shown in Figure 4.1. If the SIC is $\alpha^{\mathsf{T}} x \geq \beta$, then (α, β) is a feasible solution to the (PRLP) from the SIC point-ray collection, such that every constraint is satisfied at equality. However, $(-\alpha, -\beta)$ is also feasible to this (PRLP), corresponding to the invalid cut $\alpha^{\mathsf{T}} x \leq \beta$. Thus, the SIC point-ray collection is proper but not globally proper. The point-ray collections in Chapter 2 were similarly proper but not necessarily globally proper; in contrast, the present chapter builds globally proper point-ray collections.

The utility of working with a globally proper point-ray collection is that one does not need to check whether $\bar{x}$ is cut away in order to ensure validity. This is useful because $\bar{x}$ is a transient point, no longer truly important after the first cut that removes it. One must consider this fact for any cutting plane method, but especially ones that attempt to forego rounds of cuts (which of course provide useful information on subsequent LP optima) in favor of producing a large number of cuts all at once. Unfortunately, much experimentation is required in order to understand which are the relevant cutting planes to generate that are satisfied by $\bar{x}$. As a result, though our point-ray collections will be globally proper, our computational experiments will only produce cuts that remove $\bar{x}$; the non-violating cuts remain the subject of future research.

Theorem 4.2 shows that the extreme ray (because β is not normalized) solutions to (PRLP) correspond to facet-defining inequalities for the point-ray hull.

Theorem 4.2. *The inequality* $\alpha^{\mathsf{T}} x \geq \beta$ *is valid for* $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ *if and only if* (α, β) *is a feasible solution to* (PRLP) *from the point-ray collection* $(\mathcal{P}, \mathcal{R})$. *The inequality defines a facet if and only if the solution* (α, β) *is an extreme ray of* (PRLP).

Proof. Every point $\hat{x} \in \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ can be expressed as a convex combination of the points and rays in $(\mathcal{P}, \mathcal{R})$. For any inequality $\alpha^{\mathsf{T}} x \geq \beta$ satisfied by all of these points and rays, it follows that $\alpha^{\mathsf{T}} \hat{x} \geq \beta$. Every extreme ray (α, β) to (PRLP) has the additional property that n affinely independent points and rays from the point-ray collection satisfy $\alpha^{\mathsf{T}} x = \beta$, which means that the inequality defines a facet of $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$. The reverse directions follow, respectively, from the definitions of valid and facet-defining. □

As a direct corollary, we obtain a necessary and sufficient condition for a point-ray collection to be globally proper.

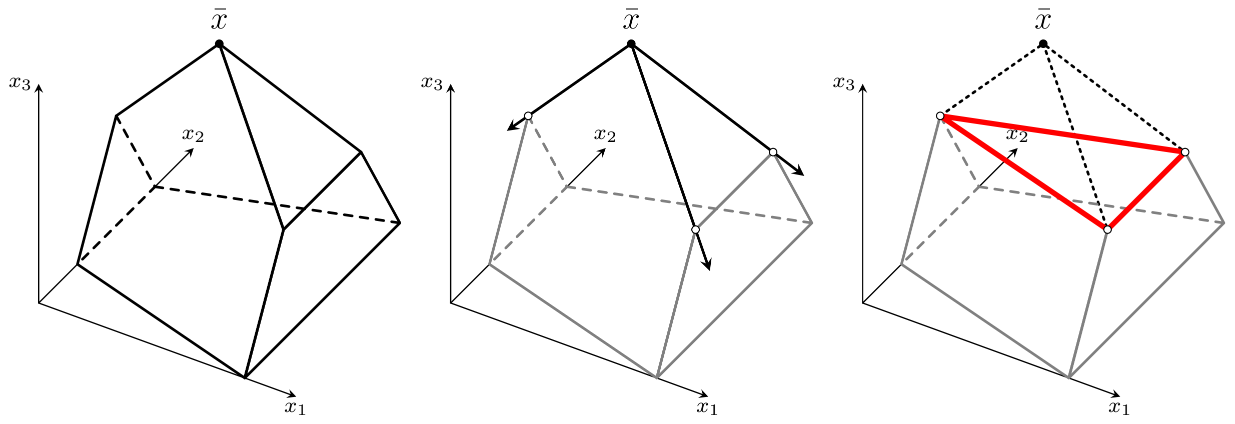


Figure 4.1: Example of the point-ray collection for a SIC. The first panel shows the feasible region P and the optimal solution $\bar{x}$. The second panel shows the cone $C(\mathcal{N})$ and the three points obtained by intersecting it with the boundary of the split disjunction on variable x_1. The third panel shows the the SIC, which is also the convex hull of the intersection points.

Corollary 4.3. *A point-ray collection $(\mathcal{P}, \mathcal{R})$ is globally proper if and only if $P_I \subseteq \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$.*

Proof. Sufficiency of the condition follows from Theorem 4.2: every feasible solution to (PRLP) is valid for $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ and hence for P_I. Necessity is similarly evident as, otherwise, there exists an extreme ray of (PRLP) that cuts a point in P_I. $\square$

This is distinct from testing whether a point-ray collection is merely proper in the sense of Definition 2.2, for which we provide a necessary and sufficient condition in Theorem 4.4. Denote by $P_{\bar{x}}$ the connected component of $P \setminus P_D$ that contains $\bar{x}$.

Theorem 4.4. *A point-ray collection $(\mathcal{P}, \mathcal{R})$ is proper if and only if the line segment between every $\hat{x} \in P_{\bar{x}}$ and every point in P_I intersects $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$.*

Proof. Necessity is clear as otherwise a valid inequality for $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ separates both some $\hat{x} \in P_{\bar{x}}$ and a point in P_I. We show sufficiency. For every $\hat{x} \in P_{\bar{x}} \setminus (\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R}))$, there exists a facet $\alpha^\mathsf{T} x \geq \beta$ of $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ that cuts $\hat{x}$. The point-ray collection is proper if every such inequality is valid for P_I. Take any $p \in P_I$. We are given that there exists a point $q \in \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ on the line segment between $\hat{x}$ and p. Since $\hat{x}$, p, and q are all in the convex set P, $\alpha^\mathsf{T} \hat{x} < \beta$, and $\alpha^\mathsf{T} q \geq \beta$, it follows that $\alpha^\mathsf{T} p \geq \beta$. $\square$

4.2.2 VPCs from simple cone relaxations

For $t \in \mathcal{T}$, let $\mathcal{P}^t_*$ and $\mathcal{R}^t_*$ be the complete set of extreme points and rays of P^t, and define $\mathcal{P}_* := \cup_{t \in \mathcal{T}} \mathcal{P}^t_*$ and $\mathcal{R}_* := \cup_{t \in \mathcal{T}} \mathcal{R}^t_*$. As a corollary of Theorem 4.2, we know that not only is this point-ray collection globally proper, but also that extreme rays of the (PRLP) from this point-ray collection correspond to facet-defining inequalities for the disjunctive hull.

Corollary 4.5. *The point-ray collection $(\mathcal{P}_*, \mathcal{R}_*)$ is globally proper. Every extreme ray (α, β) to the associated* (PRLP) *corresponds to a facet-defining inequality $\alpha^\mathsf{T} x \geq \beta$ for P_D. Conversely, for every facet $\alpha^\mathsf{T} x \geq \beta$ of P_D, the solution (α, β) to* (PRLP) *is feasible and defines an extreme ray.*

Working with the complete $\mathcal{V}$-polyhedral description of each disjunctive term is, however, impractical, as the number of points and rays can grow exponentially large in m and n, as we showed in Proposition 2.1. A natural alternative would be to use some small, strict subset of the point-ray collection $(\mathcal{P}_*, \mathcal{R}_*)$. It is not difficult to see that this could lead to invalid cuts, as shown in Appendix D.1. This is the reason Perregaard and Balas [127] and Louveaux et al. [114] employ constraint generation to obtain valid cuts.

We take a different direction from the previous work by way of the following corollary to Corollary 4.3, stating that as long as the point-ray hull forms a $\mathcal{V}$-polyhedral relaxation of P_D, then the point-ray collection is globally proper.

Corollary 4.6. *If $P^t \subseteq \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ for all $t \in \mathcal{T}$, then the point-ray collection $(\mathcal{P}, \mathcal{R})$ is globally proper.*

Hence, to implement VPCs, instead of pursuing all facet-defining inequalities for the disjunctive hull, we use a relaxation of P_D with a compact $\mathcal{V}$-polyhedral description. The simplest such relaxation for each disjunctive term is the polyhedral cone $C(\mathcal{N}(p^t))$ formed from an optimal solution p^t to $\min_x\{c^\mathsf{T} x : x \in P^t\}$ and the cobasis $\mathcal{N}(p^t)$ associated with p^t. We refer to the resulting set of points and rays as a *simple point-ray collection* and the cuts from the (PRLP) from this point-ray collection *simple* VPCs. We state their validity as Lemma 4.7, a corollary of Corollary 4.6 and the fact that $P_I \subseteq \cup_{t \in \mathcal{T}} P^t \subseteq \cup_{t \in \mathcal{T}} C(\mathcal{N}(p^t))$.

Lemma 4.7. *If, for all $t \in \mathcal{T}$, $\mathcal{P}^t$ contains p^t and $\mathcal{R}^t$ is the set of n extreme rays of $C(\mathcal{N}(p^t))$, then the point-ray collection $(\cup_{t \in \mathcal{T}} \mathcal{P}^t, \cup_{t \in \mathcal{T}} \mathcal{R}^t)$ is globally proper.*

4.2.3 VPCs corresponding to facets of the disjunctive hull

Using a $\mathcal{V}$-polyhedral relaxation such as $C(\mathcal{N}(p^t))$ for each disjunctive term has the advantage of producing a compact point-ray collection, but it may restrict the set of inequalities obtainable from (PRLP). Some inequalities valid for the disjunctive hull may not even be feasible to (PRLP), as demonstrated by the example in Figure 4.2.

While not all facet-defining inequalities for P_D can be generated, it is natural to ask under which conditions VPCs are guaranteed to correspond to facet-defining inequalities for P_D. This is what we address in Proposition 4.8.

Proposition 4.8. *Let $(\mathcal{P}, \mathcal{R})$ be a globally proper point-ray collection. If every point in $\mathcal{P}$ belongs to P_D and every ray in $\mathcal{R}$ is an extreme ray of P, then every cut corresponding to an extreme ray of* (PRLP) *defines a facet of P_D.*

Proof. By Theorem 4.2, every extreme ray of (PRLP) corresponds to an inequality that is facet-defining for $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$. By definition of facet, this inequality is satisfied at equality by n affinely independent extreme points or rays of the point-ray hull. Every

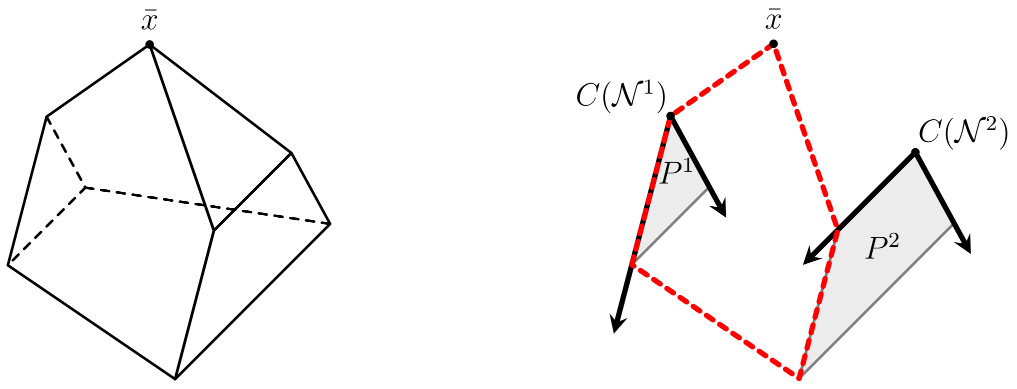


Figure 4.2: This example shows that using the relaxations $C(\mathcal{N}(p^t))$ may prevent certain valid inequalities from being generated. The right panel shows that the dashed inequality that is valid for P_I would violate a ray of the cone $C(\mathcal{N}^2)$.

point in $\mathcal{P}$ belongs to P_D by assumption, and, by the Fundamental Theorem of Integer Programming [117], the recession cones of P and $\operatorname{conv}(P_I)$ coincide, so every extreme ray of P also belongs to the disjunctive hull. Thus, the inequality is also facet-defining for P_D. $\square$

It may be difficult to meet or check the conditions of Proposition 4.8 in practice. Hence, we next give a weaker sufficient condition under which facets of $\operatorname{conv}(\mathcal{P})+\operatorname{cone}(\mathcal{R})$ correspond to facets of P_D. Given an extreme ray (α, β) of (PRLP) from a point-ray collection $(\mathcal{P}, \mathcal{R})$, we will say that a point $p \in \mathcal{P}$ or ray $r \in \mathcal{R}$ is nonbasic (or in the cobasis) if the slack variable of the corresponding constraint in (PRLP) is nonbasic.

Proposition 4.9. *Let $(\mathcal{P}, \mathcal{R})$ be a globally proper point-ray collection and (α, β) be an extreme ray of* (PRLP)*. If all points in $\mathcal{P}$ belong to P_D and, for every nonbasic ray $r \in \mathcal{R}$, there exists $\epsilon > 0$ and a nonbasic point p such that $p + \epsilon r \in P_D$, then $\alpha^\intercal x \geq \beta$ defines a facet of P_D.*

Proof. The cobasis corresponding to (α, β) provides n affinely independent points from P_D that satisfy $\alpha^\intercal x = \beta$. This is because the constraints of (PRLP) defined by the nonbasic variables are necessarily affinely independent. Moreover, our assumptions imply that, for every nonbasic ray $r \in \mathcal{R}$, there is a nonbasic point $p \in \mathcal{P}$ and $\epsilon > 0$ such that $\hat{p} := p + \epsilon r$ belongs to P_D. It follows that $\alpha^\intercal \hat{p} = \beta$ as $\alpha^\intercal p = \beta$ and $\alpha^\intercal r = 0$, by definition of nonbasic. $\square$

We might observe that, although for simple VPCs, each ray in $\mathcal{R}$ actually only comes from just one disjunctive term, Proposition 4.9 requires checking all pairs of points and rays (across disjunctive terms). Proposition 4.10 shows this is unnecessary for split disjunctions; it is sufficient to guarantee facet-defining inequalities for P_D through a local property.

Proposition 4.10. *Given two terms P^1 and P^2 of a split disjunction, if the basis defining p^t is unique for $t \in [2]$, then the simple VPCs corresponding to extreme rays of* (PRLP) *from this point-ray collection are all facet-defining for P_D.*

Proof. Under the assumption, for each $t \in [2]$ and $j \in [n]$, every ray r^{jt} of $C(\mathcal{N}(p^t))$ emanating from p^t contains in its relative interior a point $p^{jt} \in P^t$. The result follows from

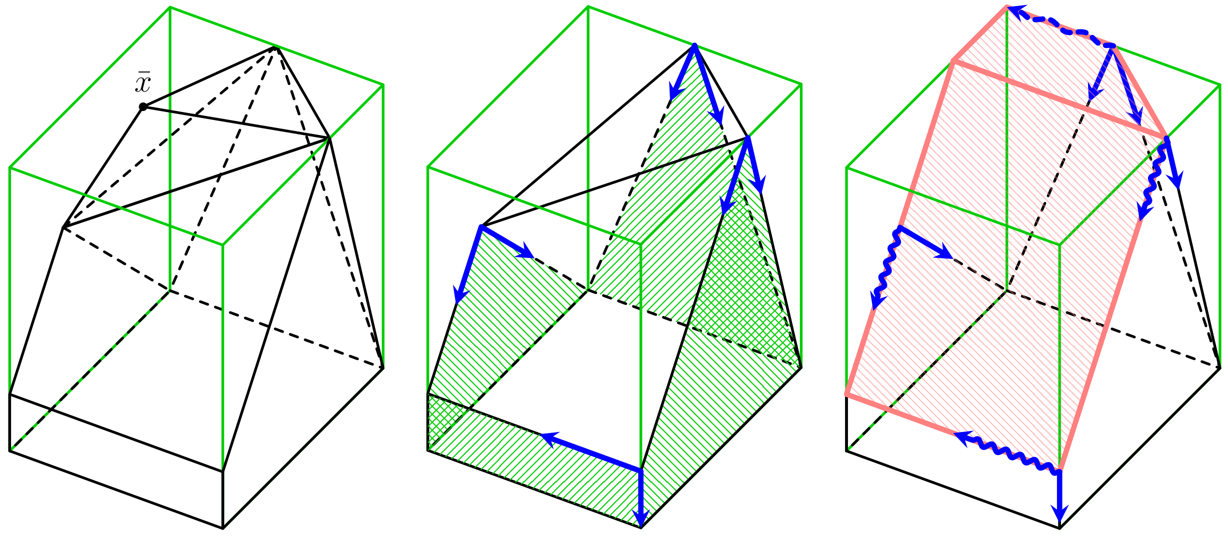


Figure 4.3: Example of a four-term disjunction that shows even if the point-ray collection is composed only of points that belong to the disjunctive hull and rays that contain such points in their relative interior, facets of $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ may still not correspond to facet-defining inequalities for P_D. The first panel shows P; the second panel shows P^t and the cones $C(\mathcal{N}(p^t))$ for $t \in [4]$ as well as the disjunctive hull; and the third panel shows $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ and the points and rays (in wavy lines) that are tight for each of the facets of this point-ray hull. Observe that the dashed wavy line is a ray that is added to $\mathcal{R}$ from one term of the disjunction, but affects the point-ray hull when it originates from a point from a different term.

applying Proposition 4.9 and the fact that $\alpha^\mathsf{T} p^1 = \alpha^\mathsf{T} p^2 = \beta$ because these are the only two nonhomogeneous constraints of (PRLP). $\square$

In contrast, Figure 4.3 shows that an analogous (and logical-seeming) result is false for more general disjunctions. The intuition is that, although a ray originates from some particular point, it is ultimately added to *all* points in $\mathcal{P}$ to calculate the disjunctive hull.

4.2.4 Normalization for the PRLP

As we presented it, (PRLP) faces the same issue as (CGLP) without any normalization, in that it is unbounded for every nonzero objective direction w, as one can arbitrarily scale any feasible solution (α, β) by any positive scalar. An unbounded direction corresponds to a ray γ belonging to the recession cone of the feasible region to (PRLP). In theory, this can be used to generate the valid cut $\gamma^\mathsf{T} x \geq 0$, but, in practice, the unbounded directions returned by our linear programming solver are not guaranteed to be extreme rays of the recession cone and as a result may correspond to weak cuts, so we reject them.

One solution is to add a normalization constraint that bounds the magnitude of the cut coefficients, which takes the form $\sum_{i=1}^{n} |\alpha_i| \leq 1$. Such a normalization has the undesirable characteristic that it may add extreme points to the feasible region of (PRLP) that do not correspond to facet-defining inequalities for the disjunctive hull.

A second normalization, adopted in [114, 127] and proposed by Balas and Perregaard [33], is to add the constraint $\alpha^\mathsf{T}(p - \bar{x}) = 1$, for some $p \in P_D$. This guarantees that (PRLP) is always bounded and has certain other nice properties, but depends on a good choice of p.

We take a third approach for normalizing (PRLP) that involves fixing β to a constant value. As observed in [31], it suffices to consider only three values for β: $\{-1, 0, +1\}$. This might indicate that one would need to solve three linear programs to generate cuts with such a normalization, as stated in the discussion of normalizations in [114]. We avoid this issue by formulating (PRLP) in the *nonbasic space* defined by the cobasis $\mathcal{N}$ at $\bar{x}$. In this space, the optimal solution $\bar{x}$ is the origin. As a result, if we are looking for cuts that are violated by $\bar{x}$, it suffices to fix $\beta = 1$. Another advantage of working in the nonbasic space is that (PRLP) can be expected to be much sparser than if it were formulated in the structural space of variables. This is because the number of nonzero components in every row roughly corresponds to the number of simplex pivots from $\bar{x}$ to the point or ray for that row.

Though in this chapter, the assumption is that $\beta = 1$, in practice we will actually use some scaled positive constant determined by the input, as $\beta = 1$ could lead to numerical issues, due to how it causes the cut coefficients scale. To illustrate this, suppose $c^\mathsf{T} p^t \geq 10^8$ for all $t \in \mathcal{T}$. Then, if $\beta = 1$, $\alpha = c/10^8$ is a feasible solution to (PRLP). However, coefficients less than 10^{-7} are often regarded as zero by solvers, so we may end up generating cuts incorrectly with improper scaling.

4.2.5 Choosing strong and successful objectives

Having set up the constraints of (PRLP), we now turn to the objective directions w. Choosing these carefully is critical to the success of any VPC algorithm, as the objectives directly determine the nature of the VPCs obtained. Aside from the type of cuts obtained, it is also important to make the cut-generating process efficient. We say that every time we solve (PRLP) and a new cut is not generated, a *failure* occurs. In our experiments with GICs from Chapter 2 and an early implementation of VPCs, failures were frequent (over 85% of the objectives tried).

One reason for failure is that (PRLP) may be infeasible for a particular point-ray collection. Figure 4.3 actually illustrates such a a situation. In that example, infeasibility is a consequence of $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ not having any facet-defining inequalities that cut away $\bar{x}$. Section 2.6.2 gives the precise theoretical conditions for when (PRLP) is guaranteed to be feasible. The next proposition, which can be derived from those results but is simpler to show directly, gives a sufficient condition for the feasibility of (PRLP) that we use in our implementation, as it applies to our experimental setup of working in the nonbasic space. Let $\underline{p}$ denote a point from $\mathcal{P}$ with minimum objective value, i.e., $\underline{p} \in \arg\min_p \{c^\mathsf{T} p : p \in \mathcal{P}\}$.

Proposition 4.11. *If $\beta = 1$, $\bar{x} = 0$, and $c^\mathsf{T}\underline{p} > 0$, then* (PRLP) *constructed from a simple point-ray collection is feasible.*

Proof. It suffices to observe that $\alpha = c/c^\mathsf{T}\underline{p}$ is a feasible solution to (PRLP). For $p \in \mathcal{P}$, $\alpha^\mathsf{T} p = c^\mathsf{T} p / c^\mathsf{T}\underline{p} \geq 1$ by definition of $\underline{p}$. For $r \in \mathcal{R}$, we need to show that $c^\mathsf{T} r \geq 0$. This follows from the fact that $p^t \in \arg\min_x \{c^\mathsf{T} x : x \in P^t\}$, which implies that p^t is also optimal when minimizing over $C(\mathcal{N}(p^t))$. For any ray $r \in C(\mathcal{N}(p^t))$, for all $\lambda > 0$, the point $p^t + \lambda r$ belongs to $C(\mathcal{N}(p^t))$. Hence, $c^\mathsf{T} p^t + \lambda c^\mathsf{T} r \geq c^\mathsf{T} p^t$. $\square$

The two other primary reasons for failures we observed were that, for the given objective direction w: (1) (PRLP) was feasible but unbounded, and (2) (PRLP) had a finite optimal solution but the corresponding cut was a duplicate of a previously generated cut. We will not be able to completely eliminate failures, but in the remainder of this section, we work towards choosing objectives that help reduce the failure rate. At the same time, we will target generating strong VPCs, while mostly ignoring potential effect of the cuts within branch-and-bound; this latter goal is poorly understood and hence difficult to target directly.

The obvious first candidate for an objective direction is to maximize the violation by $\bar{x}$, as is done in the case of L&PCs. Unfortunately, in the nonbasic space and with $\beta = 1$, $\bar{x}$ is simply the origin and all cuts have violation equal to 1. As proxies, we use two other objectives. First, we try the all-ones objective, $w_j = 1$ for all $j \in [n]$. The interpretation is that we seek an inequality that puts equal weight on cutting each of the rays of $C(\mathcal{N})$. Second, we add to P a round of GMICs, separated from $\bar{x}$, and calculate a new optimal solution $\widetilde{x}$; we then use $w = \widetilde{x}$, which finds a cut maximizing the violation with respect to $\widetilde{x}$.

Finding cuts that maximize violation with respect to points not in P_D is a paradigm that may place too much emphasis on cutting away irrelevant parts of the relaxation. The alternative is to find inequalities that minimize the slack with respect to points that do belong to P_D. Within this latter perspective, we are able to utilize whatever structural information we possess about the disjunctive hull. We will now discuss precisely what kind of information can be inferred from our $\mathcal{V}$-polyhedral relaxations of the disjunctive hull.

On one hand, we have stated that using a relaxation of each P^t, $t \in \mathcal{T}$, may lead to the (PRLP) yielding cuts that are not facet-defining for P_D. In contrast, the next result shows that, using even the potentially weak simple point-ray collection, the optimal value over the disjunctive hull can be obtained by optimizing over the points in $\mathcal{P}$.

Proposition 4.12. *Let $(\mathcal{P}, \mathcal{R})$ be a simple point-ray collection. The point $\underline{p} \in \arg\min_p\{c^\mathsf{T} p : p \in \mathcal{P}\}$ is also an optimal solution to $\min_x\{c^\mathsf{T} x : x \in \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})\}$ and to $\min_x\{c^\mathsf{T} x : x \in P_D\}$. Moreover, there are n facet-defining inequalities of $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ that can be added to P such that $\underline{p}$ is also an optimal solution to the resulting relaxation.*

Proof. This is a direct consequence of the fact that $p^t \in \arg\min_x\{c^\mathsf{T} x : x \in P^t\}$ for all $t \in \mathcal{T}$. The n inequalities are simply the ones determined by the cobasis of $\underline{p}$ from solving $\min_x\{c^\mathsf{T} x : x \in \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})\}$. □

We say that $c^\mathsf{T}\underline{p}$ is the *disjunctive lower bound* and examine whether we can achieve it via VPCs. Note that we can always add the inequality $c^\mathsf{T} x \geq c^\mathsf{T}\underline{p}$, but this is generally counterproductive, as such objective cuts tend to create multiple optimal solutions to the subsequent relaxation, which cause difficulties for solvers.

By Proposition 4.12, we know that we can attain the disjunctive lower bound via n facets of $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ tight at $\underline{p}$. One way to generate a cut tight at $\underline{p}$ is to use the objective direction $w = \underline{p}$. Absent numerical issues, the optimal solution will be some $\bar{\alpha}$) such that $\bar{\alpha}^\mathsf{T}\underline{p} = 1$. Though this is only one cut, it can be used to find other objective directions. We will work with a modified (PRLP), which we refer to as PRLP$^=$, in which the constraint $\alpha^\mathsf{T}\underline{p} \geq 1$ changed to $\alpha^\mathsf{T}\underline{p} = 1$. Let $\overline{\mathcal{R}}$ be the set of rays from $\mathcal{R}$ that are not tight for the cut; i.e., $\overline{\mathcal{R}} := \{r \in \mathcal{R} : \bar{\alpha}^\mathsf{T} r > 0\}$.

Proposition 4.13. *PRLP$^=$ with objective direction $\bar{r} \in \overline{\mathcal{R}}$ has a finite optimal solution. The optimal value is strictly less than $\bar{\alpha}^\mathsf{T}\bar{r}$ only if the resulting cut is distinct from $\bar{\alpha}^\mathsf{T}x \geq 1$. The optimal value is zero if and only if there exists a facet-defining inequality for* $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ *that cuts $\bar{x}$ and is tight on $\bar{r}$.*

Proof. The fact that PRLP$^=$ is finite and bounded is a direct result of the constraint $\alpha^\mathsf{T}\bar{r} \geq 0$ and the feasibility of PRLP$^=$. The second statement is obvious. The last statement comes from the one-to-one correspondence between extreme ray solutions to PRLP$^=$ and facet-defining inequalities for $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ that cut away $\bar{x}$. □

Propostion 4.13 resolves issue (1) mentioned above, of having a feasible (PRLP) that is unbounded. Unfortunately, we may still get failures from issue (2), meaning the optimal solution to PRLP$^=$ corresponds to a cut we previously generated. For example, it may be the case that there exists $r \in \overline{R}$ such that, for all α feasible to PRLP$^=$, $\alpha^\mathsf{T}r \geq \bar{\alpha}^\mathsf{T}r$. Using such an r as the objective for PRLP$^=$ could reproduce the solution $\bar{\alpha}$. To prevent such phenomena from excessively slowing down cut generation, we add a failure rate parameter, which detects when there is an unacceptably small percent of the objectives successfully producing new cuts. We discuss this in Section 4.3.

Lastly, we address the practical question: do all n facets from Proposition 4.9 exist as extreme rays of (PRLP) in our setup? Note that the example in Figure 4.4 shows that these n facets might not be the same as the n facet-defining inequalities of P_D that are tight at $\underline{p}$.

The answer to this question, which we give by way of example, is no. To see why, we need to understand which inequalities can be generated from (PRLP) in our setup. The specific modifications we have made from the general case are that we fix $\beta > 0$ and work in the nonbasic space in which $\bar{x}$ is the origin. This implies that we will never generate any facet-defining inequalities for $\operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$ that are satisfied by $\bar{x}$. One might initially assume that these inequalities are not necessary in order to attain the bound $c^\mathsf{T}\underline{p}$. We dispel that notion in Figure 4.4. The example demonstrates that ignoring inequalities that do not cut away $\bar{x}$ may lead to an optimal value (after adding cuts) that is strictly better than $c^\mathsf{T}\underline{p}$. Note that in this example, the point-ray collection uses the complete $\mathcal{V}$-polyhedral description of each P^t, so, unlike the example in Figure 4.2, this situation is not a consequence of using a relaxation of P_D. This may partially explain why, in our experiments, despite our refined objective choices, we do not always obtain the bound $c^\mathsf{T}\underline{p}$.

4.3 Computational setup

The technique we call $\mathcal{V}$-polyhedral cuts is quite general, and there are many possibilities for implementing it. In Appendix D.3, we list and briefly discuss some of the many possible parameters that may play a role in the VPC framework. We experiment with only a small subset of the parameters, so a more thorough tuning may improve upon our reported results. Section 4.2 covered the rationale driving some of our choices: given a valid disjunction, we described how to construct a globally proper point-ray collection from that disjunction, normalize the corresponding (PRLP), and judiciously select objective directions when generating VPCs. We put these elements together in Algorithm 4.6, after which we describe additional details of our experiments.

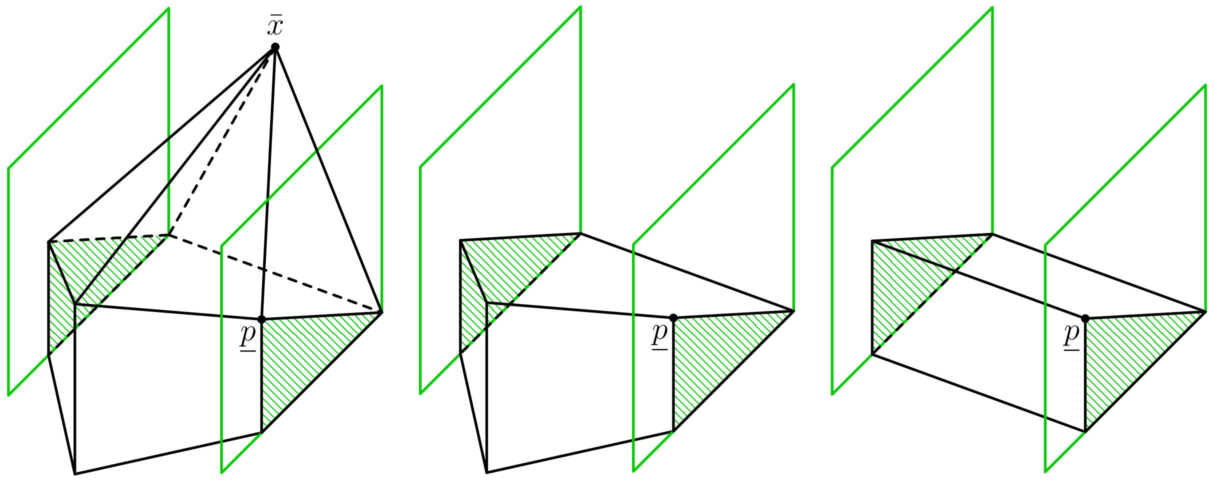


Figure 4.4: Example that shows an inequality tight at $\underline{p}$ that does not cut away $\bar{x}$ may be necessary for achieving the bound $c^\mathsf{T}\underline{p}$. In this example, we assume we are maximizing along the vertical axis. The first panel shows the original polytope. The second panel is the polytope after adding the only split inequality that cuts away $\bar{x}$. The third panel shows the polytope after adding all the valid split cuts.

Algorithm 4.6 Type 1 $\mathcal{V}$-Polyhedral Cuts

```
Input: Polyhedron $P$; disjunction $\vee_{t\in\mathcal{T}} P^t$.
 1: function VPC1($P, \{P^t\}_{t\in\mathcal{T}}$)
 2:     for $t \in \mathcal{T}$ do
 3:         $p^t \leftarrow$ optimal solution to $\min_x\{c^\mathsf{T}x : x \in P^t\}$.
 4:         $\mathcal{N}(p^t) \leftarrow$ cobasis associated with $p^t$.
 5:         $\mathcal{P}^t \leftarrow \{p^t\}$ and $\mathcal{R}^t \leftarrow$ rays of $C(\mathcal{N}(p^t))$.
 6:     $\mathcal{P} \leftarrow \cup_{t\in\mathcal{T}}\mathcal{P}^t$ and $\mathcal{R} \leftarrow \cup_{t\in\mathcal{T}}\mathcal{R}^t$.
 7:     Set up the constraints of (PRLP) using the point-ray collection $(\mathcal{P},\mathcal{R})$.
 8:     $\mathcal{C} \leftarrow \emptyset$.
 9:     Solve (PRLP) with all-ones objective $w = e$ and add the resulting cut to $\mathcal{C}$.
10:     Add GMICs to $P$ and let $\widetilde{x}$ denote an optimal solution to the refined relaxation.
11:     Solve (PRLP) with $w = \widetilde{x}$ and add the resulting cut to $\mathcal{C}$.
12:     $\underline{p} \leftarrow$ point from $\arg\min_p\{c^\mathsf{T}p : p \in \mathcal{P}\}$.
13:     Solve (PRLP) with $w = \underline{p}$ and add the resulting cut, $\bar{\alpha}^\mathsf{T}x \geq 1$, to $\mathcal{C}$.
14:     PRLP$^=$ $\leftarrow$ (PRLP) with added constraint $\alpha^\mathsf{T}\underline{p} = 1$.
15:     $\overline{\mathcal{P}} \leftarrow \{p \in \mathcal{P} : \bar{\alpha}^\mathsf{T}p > 1\}$, sorted in order of decreasing angle with the objective.
16:     $\overline{\mathcal{R}} \leftarrow \{r \in \mathcal{R} : \bar{\alpha}^\mathsf{T}r > 0\}$, sorted in order of decreasing angle with the objective.
17:     for all $\bar{w} \in \overline{\mathcal{P}} \cup \overline{\mathcal{R}}$ do
18:         Solve PRLP$^=$ with $w = \bar{w}$ and add the resulting cut to $\mathcal{C}$.
19:         Remove from $\overline{\mathcal{P}} \cup \overline{\mathcal{R}}$ all points and rays that are tight for the generated cut.
20:         If twice the cut limit objectives have been tried in this loop, break (go to step 21).
21:     return Set $\mathcal{C}$ of generated cuts.
22: end function
```

Our general choices are the following. The disjunctive terms provided as input to Algorithm 4.6 will be the leaf nodes of a partial branch-and-bound tree. We only use one round of cuts and all cuts are rank one with respect to P. We set a limit on the number of generated VPCs that is equal to the number of integer variables that are fractional at $\bar{x}$, i.e., the same as the limit on the number of Gomory cuts. A variety of tolerances and other limits are relevant in various parts of the code; we mention these in context later in this section.

In this basic setup, we do not vary the cut limit, the number of cut rounds (which we restrict to one in accordance with our goal of avoiding recursive cut generation), or impose limits on cut orthogonality or maximum density. These all merit further exploration, as prior work has repeatedly demonstrated that these values can mean the difference between an algorithm that works in practice and one that seems to produce negative results. For instance, the importance of cut orthogonality has been noted previously by, e.g., Balas et al. [25] and Ferris et al. [82]. See also the discussion in Karamanov [103, Chapter 3].

Environment. All algorithms are implemented in C++in the COIN-OR framework [112] using `Clp version 1.16r2294` and `Cbc version 2.9r2351` as the underlying branch-and-bound solvers, respectively, to generate VPCs. The solver used to test the effectiveness of VPCs when used as part of branch-and-bound is `Gurobi version 7.5.1` [96]. The machine used is a 64-bit `PowerEdge R515` with 64GB of memory and twelve `AMD Opteron 4176` processors clocked at 2.4GHz. The operating system is `Linux Fedora 19` and compiler is `g++ version 4.8.3 20140911 (Red Hat 4.8.3-7)`.

Evaluation. We evaluate cuts from two different perspectives: strength and effect on branch-and-bound. The strength of the cuts is assessed by the percent root gap closed (defined in Chapter 2) by one round of VPCs. As a baseline, we also report the percent gap closed by adding one round of GMICs, as well as the percent gap closed by using both VPCs and GMICs together. In addition, we report, both with and without the use of VPCs, the root gap closed by `Gurobi` after one cut pass and after the last round of cuts is added at the root. The effect on branch-and-bound is measured by the time `Gurobi` takes to solve the problem with VPCs added as user cuts; this is compared to the time taken without VPCs.

Instance selection. Instances were selected from the union of MIPLIB [5, 43, 44, 109], CORAL [56], and NEOS[1] sets by the following criteria: (1) the IP optimal value is known; (2) the number of rows and number of columns of the presolved instance is each at most 5,000; (3) the partial branch-and-bound tree with 64 leaf nodes does not find an IP optimal solution; (4) the disjunctive lower bound is strictly less than the maximum objective value of any leaf node. The last criterion aims to avoid infeasible instantiations of (PRLP), which is what typically occurs when $\max_{p \in \mathcal{P}} c^{\mathsf{T}} p = c^{\mathsf{T}} \underline{p}$. When we report gap closed, we will also filter by another condition, that $c^{\mathsf{T}} \underline{p} > c^{\mathsf{T}} \bar{x}$ from the 64-leaf partial tree, as instances in which the disjunctive lower bound and optimal value of the LP relaxation coincide are not good candidates for evaluation by gap closed.

Of the instances with at most 5,000 rows and columns, Table D.1 in Appendix D.4 lists the 75 that were removed from consideration and the reason for removal, leaving 195 for our

[1]These instances are currently not easily accessible online, so they can be provided upon request.

experiments. We modify the instances `stein27` and `stein45` from the versions in MIPLIB, which contain an additional constraint providing a lower bound for the objective value that is not present in the original formulation of the problem [86].[2]

Instance preprocessing. Every instance is first preprocessed by `Gurobi`'s presolve. This procedure is used in order to improve the fairness of the testing environment. It allows VPCs to be generated from the same version of the instance that would be used internally by the branch-and-bound solver. This is also a reason for turning presolve off during the subsequent branch-and-bound tests, as one round of presolve has already been applied. At the same, for some instances, preprocessing closes a significant portion of the integrality gap, which could make the process of finding strong cuts more difficult. For reproducibility, one must be aware that not only might `Gurobi`'s presolve algorithms change with a new version (release of the software), but also they depend on the random seed given to the solver. We do not experiment with this latter variability (we presolve with the random seed 628 only).

Generating a partial branch-and-bound tree. The partial branch-and-bound tree will be generated by the node, variable, and branch selection rules that follow, which are the defaults for `Cbc`. Node selection will be roughly by the *best-first* or *best-bound* rule, in which the next node to explore will be the one with the minimum objective value (though the number of fractional variables at each node is also considered, it is to a much lesser extent). Variable selection utilizes the outcome of strong branching on up to five fractional variables at each node. Between the two possible children of the node when branching on the variable that is selected, the direction is chosen by a similar rule to the node selection criteria, i.e., typically in the direction of the child with a lower optimal value.

We will use as the disjunction for each instance the set of leaf nodes of a partial branch-and-bound tree generated by full strong branching; We stop building the partial branch-and-bound tree after reaching 2, 4, 8, 16, 32, or 64 leaf nodes of the partial tree, independently. These leaf nodes of the partial tree form the disjunction that we input to Algorithm 4.6.

From the leaf nodes, we obtain a simple point-ray collection $(\mathcal{P}, \mathcal{R})$. When constructing the associated (PRLP) (in the nonbasic space defined by the cobasis at $\bar{x}$), we remove all duplicate rows. In addition, any rows that are actually bounds on the α variables are removed as explicit constraints and kept as bounds instead. Henceforth, we assume that $(\mathcal{P}, \mathcal{R})$ has no duplicates. We proceed with generating cuts from a given (PRLP) if it is feasible and solves to optimality within a minute when using no objective, i.e., just the feasibility problem.

Cut processing. Not every objective function we try for (PRLP) leads to a new cut, a subject we started discussing in Section 4.2.5. In that section, the failures discussed were (PRLP) being unbounded or leading to a duplicate cut. Other failures are imposed by conditions that we set. If the time to solve (PRLP) for an objective is greater than 5 seconds, we abandon the objective. If we do obtain a solution, since (PRLP) is formulated in the nonbasic space, we first convert the cut to the structural space, yielding a cut $\gamma^{\mathsf{T}} x \geq \gamma_0$. Next, we remove small coefficients: for $j \in [n]$, if $|\gamma_j| < \epsilon$, we ignore the coefficient, and

[2]In contrast to Chapters 2 and 3, we do not artificially reduce symmetry by replacing the objective for these instances.

if $\epsilon \le |\gamma_j| < \epsilon_{\text{coeff}}$, we replace x_j by either its lower or upper bound and adjust γ_0. In our experiments, $\epsilon = 10^{-7}$, and $\epsilon_{\text{coeff}} = 10^{-5}$. Assume that $\gamma^\mathsf{T} x \ge \gamma_0$ has been processed in this way. The cut is rejected if it is a duplicate of or dominated by a previously generated VPC. It will also be rejected if its dynamism ($\max_{j\in[n]}|\gamma_j| / \min_{j\in[n]}\{|\gamma_j| : \gamma_j \neq 0\}$) is higher than 10^8. In addition, if there exists some previously generated VPC $\alpha^\mathsf{T} x \ge \beta$ that is nearly parallel to $\gamma^\mathsf{T} x \ge \gamma_0$, i.e., if $\alpha \cdot \gamma / (\|\alpha\| \cdot \|\gamma\|) < \epsilon_{\text{orth}}$ ($\epsilon_{\text{orth}} = 10^{-3}$ in our setup), then we keep only one of these two cuts (the one that separates $\bar{x}$ by a greater Euclidean distance, or if these are equal, the sparser cut).

We solve (PRLP) until we exhaust all objectives or reach one of the stopping criteria:

1. Numerical difficulties are encountered while solving (PRLP).
2. The time limit for cut generation is reached.
3. The cut limit is reached.
4. The failure limit is reached.

An example of numerical difficulties we have encountered is when (PRLP) solves to optimality for one objective but is deemed primal infeasible for another. The time limit for cut generation is 900 seconds (the time to set up the partial tree and build (PRLP) is not counted against this). The cut limit is equal to the number k of fractional variables at the LP optimal solution.

The failure limit we use comes from some experimentation and is not trivial. It varies based on several parameters: there are different maximum failure rates depending on whether "few" or "many" cuts have been generated, and whether "many" objective functions have been attempted. Let $\phi_{\text{few\_cuts}} := 0.95$, $\phi_{\text{many\_cuts}} := 0.90$, $\phi_{\text{many\_obj}} := 0.80$. We define "few" cuts as $n_{\text{few\_cuts}} := 1$, "many" cuts as $n_{\text{many\_cuts}} := \lceil k/4 \rceil$, and "many" objectives as

$$n_{\text{many\_obj}} := \max\{\lceil n_{\text{few\_cuts}}/(1-\phi_{\text{few\_cuts}}) \rceil, \lceil n_{\text{many\_cuts}}/(1-\phi_{\text{many\_cuts}}) \rceil\}.$$

Hence, the default for $n_{\text{many\_obj}}$ is $\max\{20, 10\lceil k/4 \rceil\}$. After each cut, we test whether the current failure ratio (number of unsuccessful objectives as a proportion of the total number of objectives attempted) is greater than the appropriate threshold ($\phi_{\text{few\_cuts}}$, $\phi_{\text{many\_cuts}}$, or $\phi_{\text{many\_obj}}$); if it is, then we return that we have reached the failure limit. We also say we reached the failure limit if the first $\lceil n_{\text{few\_cuts}}/(1-\phi_{\text{few\_cuts}}) \rceil$ objectives all lead to failures; this is often an indicator of numerical issues with the instance. As more cuts are generated and more objectives are tried, we decrease the acceptable failure rate as there is likely to be diminishing marginal benefit for additional cuts and we wish to avoid spending excessive amounts of time attempting to generate cuts unsuccessfully.

Branch-and-bound settings. The VPCs will be given to `Gurobi` at the root as *user cuts*, which allows `Gurobi` to use its internal cut selection criteria. All of `Gurobi`'s default parameters are used except the following:

- Fix a random seed (we use 628).
- Set a time limit of 3600 seconds.

- Disable use of multiple threads.
- Disable presolve. (Each instance is already presolved during preprocessing.)
- Disable heuristics. (This is to reduce performance variability.)
- Disable dual reductions. (Required for enabling user cuts.)
- Provide the IP optimal value (with a small tolerance) via `bestBdStop`.

Using different partial tree sizes for VPCs could introduce a bias, so we also test how `Gurobi` performs on the instances across seven different random seeds ($i \cdot 628$ for $i \in [7]$), and we report the best result across these runs as well as on just the first random seed. Note that VPCs are not tested with these other random seeds, as the purpose of these other seeds is explicitly to give `Gurobi` an advantage over VPCs to negate the bias introduced by using multiple partial trees; on the other hand, we may be overcompensating, as in practice, it is likely some combination of the variability due to random seeds and VPCs would be utilized.

4.4 Computational results

The goals of the computational experiments are to assess (1) the strength of VPCs by the percent root gap closed by one round of the cuts, which we discuss in Section 4.4.1, and (2) the effectiveness of VPCs when added at the root and used as part of branch-and-bound, covered in Section 4.4.2. In Section 4.4.3, we analyze at a finer level of detail where our procedure works well and where it can be improved. In the subsequent tables and figures, "G" refers to GMICs, "V" refers to VPCs, "GurF" refers to `Gurobi` after one round of cuts at the root, "GurL" refers to `Gurobi` after the last round of cuts at the root, and "DB" refers to the value of the disjunctive lower bound $c^\mathsf{T}\underline{p}$ from the partial branch-and-bound tree for each instance with 64 leaf nodes (an upper bound on the gap we can close using VPCs on their own). For the columns including VPCs, sometimes a partial tree size will be specified (either as "V (ℓ)" or "ℓ leaves"), indicating that these results concern only the runs for partial trees with ℓ leaf nodes. When a partial tree size is not specified, the result shown will be the best across the six partial tree sizes tested.

4.4.1 Percent root gap closed

Table 4.1 provides a summary of the average percent gap closed by GMICs, VPCs, and VPCs used together with GMICs, as well as the percent gap closed by one round of cuts at the root by `Gurobi` and after the last round of cuts added by `Gurobi` at the root. Table D.2, in the appendix due to its length, contains the values for all the instances. In these results, no a posteriori strengthening techniques, as described in Section 1.5, are applied to VPCs: there are two sources of cut strength, one from the disjunction from which the cut is produced, and one from modularization taking advantage of integrality properties *after* the cut is generated. We are not using this second strength in this chapter, as applying the technique requires the use of additional information about the cut when it is derived from a disjunction that is not

Table 4.1: Summary statistics for percent gap closed by VPCs.

Set	# inst		G	DB	V	V+G	GurF	V+GurF	GurL	V+GurL
All	184	Avg (%)	17.28	24.03	15.60	26.95	25.99	33.03	46.48	52.07
		Wins		114	91	156		143		116
≥10%	87	Avg (%)	14.41	37.73	29.55	33.47	20.03	32.59	38.81	49.95
		Wins		75	71	84		73		68
1K	81	Avg (%)	16.38	25.25	20.73	30.61	23.90	34.14	43.40	52.41
		Wins		57	51	70		66		57

simple. We investigate a method to apply strengthening to VPCs in Chapter 5. Note that, in contrast, the GMICs do take advantage of modularization.

Column 1 indicates the set of instances: the first two data rows concern the 184 instances for which the disjunctive lower bound is strictly greater than the LP optimal value, the second two data rows pertain to the subset of the 184 instances for which VPCs close at least 10% of the integrality gap, while the last two data rows refer to the subset of 81 instances with at most 1,000 rows and 1,000 columns. The first row for each set gives the average for the percent gap closed across the instances. The second row for each set shows the number of "wins", which is defined as an instance for which using VPCs closes strictly more (by at least 10^{-7}) of the integrality gap. For columns "DB", "V", and "V+G" wins are relative to column "G"; for "V+GurF" wins are counted with respect to column "GurF"; and for "V+GurL" wins are with respect to column "GurL".

Column 2 gives the number of instances in each set. Column 4 is the percent gap closed by GMICs when they are added to the LP relaxation. Column 5 is the percent gap closed as implied by the disjunctive lower bound from the partial tree with 64 leaf nodes. Column 6 is the percent gap closed by VPCs; the subsequent column is the percent gap closed when GMICs and VPCs are used together. Columns 8 and 9 show the percent gap closed by `Gurobi` cuts from one round at the root, first without and then with VPCs added as user cuts. Columns 10 and 11 show the same, but after the last round of cuts at the root. Columns 8 and 10 use the best result (maximum gap closed) by `Gurobi` across the seven random seeds tested.

The results indicate the strength of VPCs. Namely, using VPCs and GMICs together leads the average percent gap closed at the root to increase from 17% to 27%. VPCs on their own close strictly more gap than GMICs for 91 instances; in comparison, we see that for 114 instances the disjunctive lower bound is greater than the optimal value after adding GMICs, so there are only 23 additional instances for which VPCs on their own could have gotten stronger results. For 11 of those 23 instances, we achieve the cut limit, implying that a higher percent gap might be achieved if we permitted more cuts to be generated. When VPCs are used with GMICs together, more gap is closed for 156 of the 184 instances. Perhaps even more indicative of the utility of VPCs are the results when VPCs are used as user cuts within `Gurobi`. For the first round of cuts at the root, the percent gap closed goes from 25% (without VPCs) to 33% (with them), with strictly better outcomes for 143 of the 183 instances. For the last round of cuts at the root, the percent gap closed increases from

46.5% to 52% by using VPCs.

The results are even more pronounced for the other two sets of instances. For the instances in which VPCs perform well (close at least 10% of the integrality gap on their own), VPCs and GMICs together close over double the gap closed by GMICs on their own, with improvements for 84 of the 87 instances in that set, and VPCs provide nearly a 30% improvement in the gap closed after the last round of cuts at the root node of Gurobi (50% compared to 39%). For the 81 instances with up to 1,000 rows and columns, GMICs close 17% of the integrality gap, VPCs alone close 21%, while together they close 31%.

For the columns including VPCs, the result reported is the maximum percent gap closed across all partial tree sizes tested. One may initially assume that the strongest cuts would always come from the partial tree with 64 leaf nodes. This is indeed true of the disjunctive lower bound, but it does not always hold for VPCs. One reason, meaningful in conjunction with the fact that we generate a fixed number of cuts, is that there are likely to be more facet-defining (i.e., essential) inequalities for the disjunctive hull from stronger disjunctions. As a result, achieving the disjunctive lower bound may become more difficult, in particular given our relatively conservative cut limit. Another reason, on an intuitive level, is that more of the facet-defining inequalities for the deeper disjunctions may not cut away $\bar{x}$, which are cuts we do not generate in these experiments. Finally, the rate of numerical issues goes up as the disjunctions get larger; we investigate this more in Section 4.4.3.

Figure 4.5 shows how the average percent gap closed increases with disjunction size. In this figure, "0 leaves" refers to the gap that is closed without using any VPCs, while "Best" corresponds to the same values as in Table 4.1, i.e., the best value per instance is used across all partial tree sizes tested. We show the same four measures as in Table 4.1. In solid black is the percent gap closed by VPCs; dark gray without hatching indicates GMICs; dark gray with hatching is for the first round of cuts by `Gurobi`; and light gray with hatching is after the last round of `Gurobi` cuts at the root. In addition, the dashed line shows the average disjunctive lower bound (and upper bound on the value of the percent gap closed by VPCs). This figure shows that VPCs do get stronger as stronger disjunctions are used, and, at the same time, the marginal benefit of GMICs decreases as better disjunctive cuts are generated. An important conclusion from this figure is that our procedure avoids the "tailing-off" effect from recursive applications of cuts: without requiring recursion, by simply using a (sufficiently) stronger disjunction, we make steady progress toward the optimal value of (IP).[3] Nevertheless, we also see that there is much room for improvement in our implementation of the framework, as the gap closed by VPCs grows increasingly farther from the disjunctive lower bound with the use of larger partial trees.

4.4.2 Branch-and-bound effect

We now turn to the second metric, of the effect of our cuts on branch-and-bound in terms of time and number of nodes when VPCs are added as user cuts to `Gurobi`. There are 159 instances for which `Gurobi` is able to solve the instance to optimality within an hour either with or without using VPCs; we call this set "All" for this subsection. Of these 159

[3]In preliminary experiments, we tested multiple rounds of GMICs and observed the tailing-off phenomenon commonly reported in cutting plane literature. It is possible to apply the VPC procedure in rounds as well, which may make sense in practice, but this falls outside of the scope of the current study.

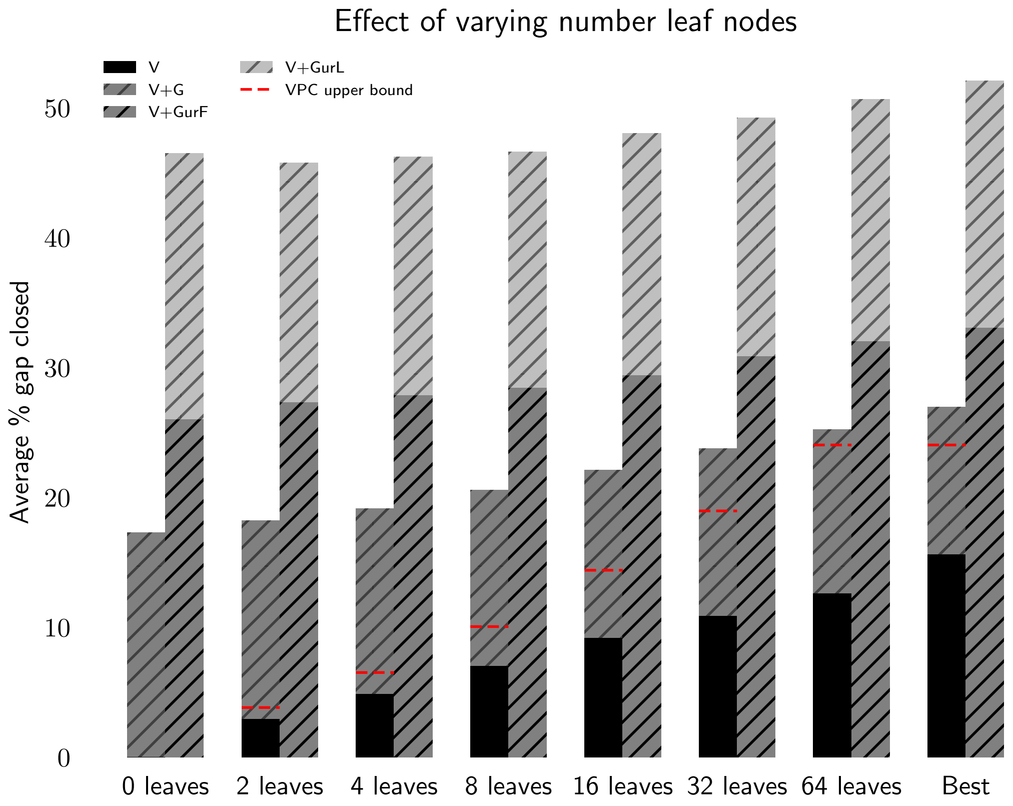


Figure 4.5: Average percent gap closed broken down by the number of leaf nodes used to construct the partial branch-and-bound tree. Shown is the gap closed for VPCs with and without GMICs, as well as the gap closed at the root by `Gurobi` after the first and last round of cuts. “0 leaves” refers to the percent gap closed when no VPCs are used. “Best” refers to the maximum gap closed, instance-by-instance, across all partial tree sizes.

Table 4.2: Summary statistics for time to solve instances with branch-and-bound.

			Time (s)					Nodes		
Set	# inst		Gur1	Gur7	V	w/PRLP	Total	Gur1	Gur7	V
All [0,3600)	159	Gmean	81.48	58.94	63.79	68.37	68.50	6069	4475	4549
		Wins1		87	89	45	45		105	109
		Wins7			41	12	12			64
All [10,3600)	76	Gmean	276.89	180.00	199.21	216.69	217.23	28642	18692	19152
		Wins1		49	43	33	33		47	47
		Wins7			13	8	8			21
All [100,3600)	37	Gmean	869.71	560.84	652.80	713.80	715.91	88735	57796	61536
		Wins1		23	20	17	17		25	19
		Wins7			4	2	2			6
All [1000,3600)	11	Gmean	2126.58	1877.63	2016.00	2017.51	2017.62	208021	173557	157995
		Wins1		4	2	2	2		6	4
		Wins7			0	0	0			1
6 trees [0,3600)	97	Gmean	65.56	54.28	54.04	56.59	56.67	6747	5588	5239
		Wins1		44	57	30	30		57	73
		Wins7			33	11	11			51
6 trees [10,3600)	41	Gmean	260.64	200.26	199.38	210.43	210.78	46013	34449	31386
		Wins1		24	24	21	21		24	29
		Wins7			10	7	7			17
6 trees [100,3600)	19	Gmean	973.42	723.02	765.35	811.93	812.85	200983	139998	135861
		Wins1		10	9	9	9		12	10
		Wins7			2	1	1			4
6 trees [1000,3600)	8	Gmean	2088.90	1861.25	1983.85	1984.85	1984.99	376028	314438	261487
		Wins1		3	1	1	1		4	3
		Wins7			0	0	0			1

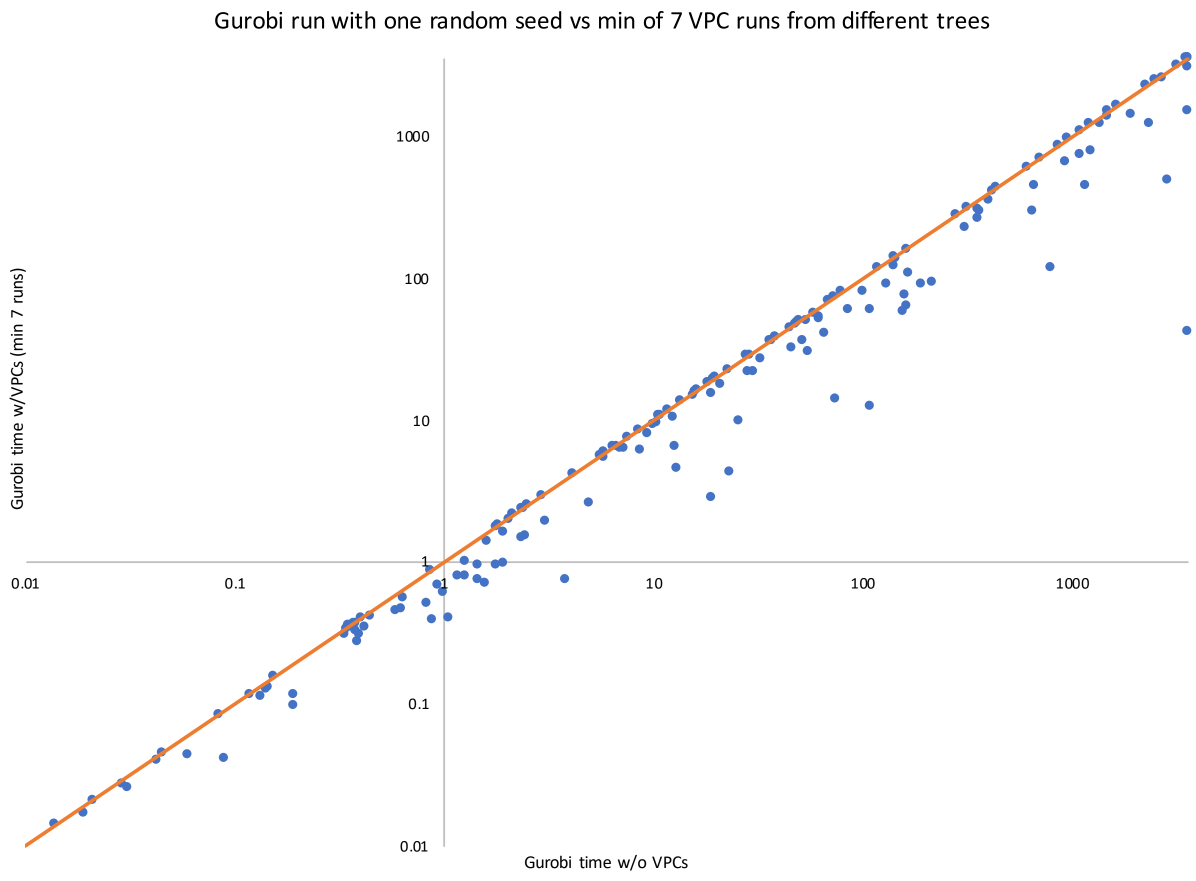


Figure 4.6: Plot in log-log scale showing on the horizontal axis the time to solve each of the instances using `Gurobi` with random seed 628 and on the vertical axis the minimum across the solution times when VPCs are added as user cuts from partial trees with ℓ leaves, $\ell \in \{0, 2, 4, 8, 16, 32, 64\}$. The solid line is the parity line; VPCs have more benefit for an instance the farther the corresponding data point is below this line. This plot indicates what can be expected given an oracle that can determine a good partial tree to use per instance.

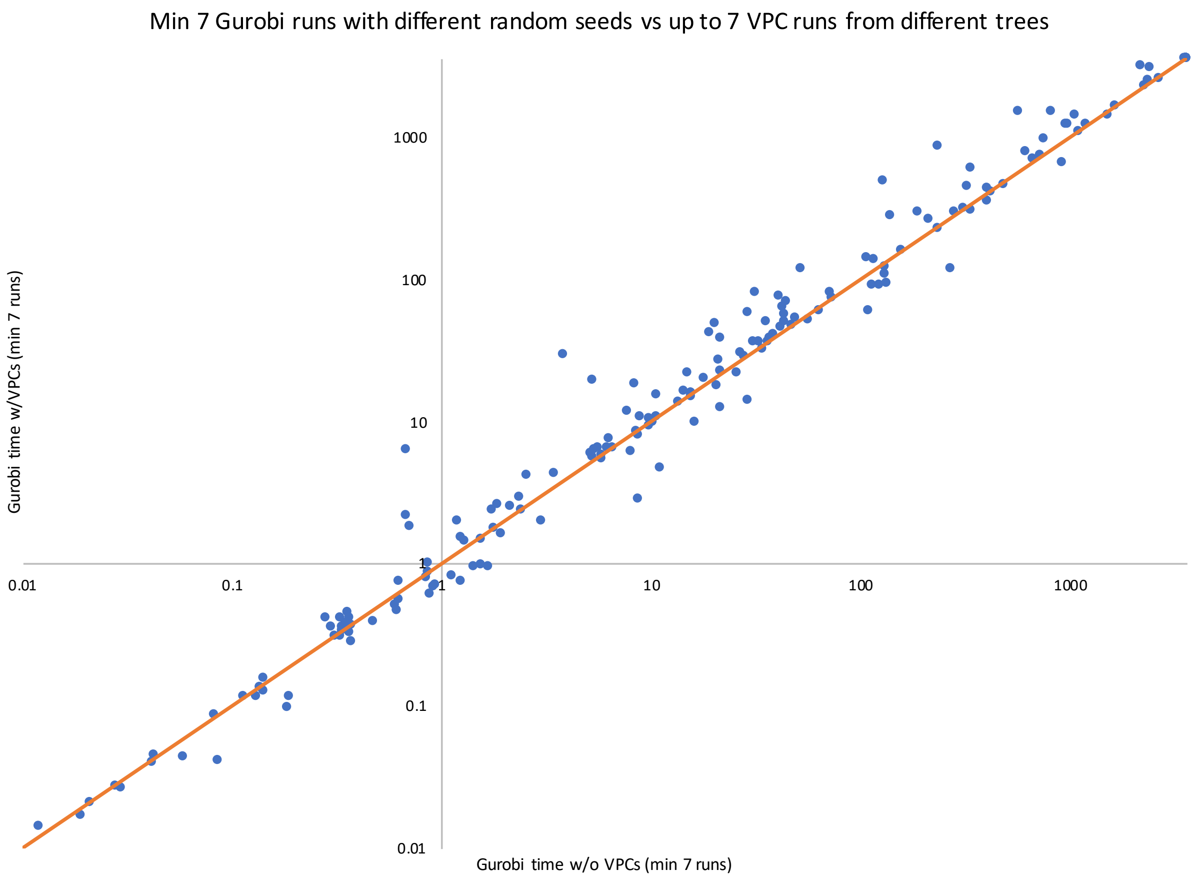


Figure 4.7: Plot in log-log scale showing on the horizontal axis the minimum time to solve each of the instances using `Gurobi` across seven random seeds ($i \cdot 628$ for $i \in [7]$) and on the vertical axis the minimum across the solution times when VPCs are added as user cuts from partial trees with ℓ leaves, $\ell \in \{0, 2, 4, 8, 16, 32, 64\}$. One aspect reflected in these results is the higher variability among more difficult instances.

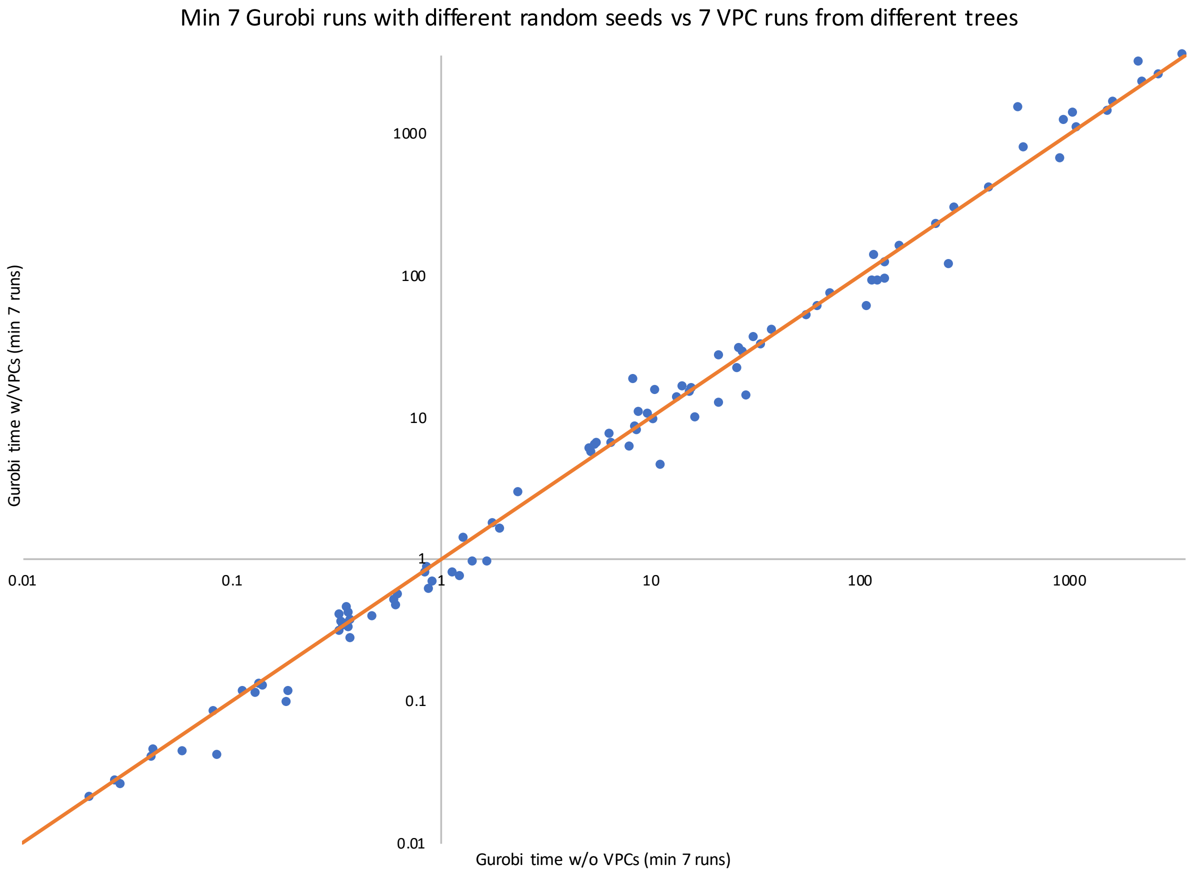


Figure 4.8: The same as Figure 4.7 but with only the instances for which cuts were generated successfully from all partial tree sizes. For easier instances, it appears that when VPCs help, they help consistently. Th results on the medium-difficulty instances are mixed: when VPCs help, they tend to improve solution times more than the improvement from using a better random seed, but using a "good" random seed helps much more frequently. For the very hard instances, our cuts do not seem to be competitive with a lucky random seed.

instances, there are 97 for which we were successfully able to generate VPCs for all six partial branch-and-bound trees; we refer to this as the "6 trees" set of instances. Table 4.2 contains a summary of the statistics for the set "All" in the top half and for the set "6 trees" in the bottom half. We further divide each set of instances into four bins, where bin $[t,3600)$ contains the subset of instances which `Gurobi` solved within an hour but took at least t seconds to solve for all experiments. The first column of the table indicates which set and bin is being considered. The second column is the number of instances in that subset. The next column indicates the three summary statistics presented for each subset. The first row, "Gmean", for each subset is a shifted geometric mean (with a shift of 60 for time and 1000 for nodes, as in [1]). The second row, "Wins1", is the number of "wins" for each column with respect to `Gurobi` run with one random seed, a baseline that we will denote by "Gur1". A win in terms of time is counted when the "Gur1" baseline is at least 10% slower, to account for some variability in runtimes. The third row, "Wins7", is the number of "wins" with respect to the best `Gurobi` run across seven random seeds, which we call the "Gur7" baseline.

We now describe the eight remaining columns of Table 4.2. Columns 4 and 5 give the values for the two baselines, "Gur1" and "Gur7". Column 6 provides the statistics for the fastest solution time by `Gurobi` when VPCs are added as user cuts (without accounting for cut generation time) across the six different partial trees tested per instance, but also including the time from "Gur1" as one of the possible minima, indicating the option of not using VPCs for that instance. Column 7 incorporates the total time taken solving (PRLP), which is the total cut generation time excluding the time taken to produce the partial branch-and-bound tree and set up the point-ray collection. Column 8 incorporates these excluded times as well. Columns 9 through 11 concern the number of nodes taken to solve each instance to optimality for the two baselines and for when VPCs are considered. For column 10, the value for "Wins7" in each bin is the number of instances for which the number of nodes for "Gur7" is fewer than the number of nodes taken by `Gurobi` with VPCs. The full results for the instances in set "All" are given in Table D.3, while the full results for the set "6 trees" are in Table D.4. However, we also plot, on a log-log scale, the values for "Gur1" against "V" (for set "All") in Figure 4.6, for "Gur7" against "V" (for set "All") in Figure 4.7, and for "Gur7" against "V" (for set "6 trees") in Figure 4.8.

The results show unequivocally that using VPCs vastly dominate the baseline "Gur1", yielding a drastic reduction in the average number of seconds and nodes to solve each instance. Of course, comparing the value of "V" to "Gur1" is tremendously biased; by definition, the values in column "V" weakly dominate the ones in column "Gur1". The purpose of such a comparison is showing the theoretical benefit of the VPC approach: with an ideal way to select a partial tree per instance (including knowing when not to use VPCs at all), these are the results one would see. Unfortunately, in practice, we do not have access to such an oracle, but the results suggest the need for crafting a good rule to take the place of this ideal oracle. The most obvious rule, of fixing ℓ leaf nodes for just one of the options $\ell \in \{2, 4, 8, 16, 32, 64\}$, does not work. These summary statistics for each partial tree size independently are given for the "6 tree" set in the appendix in Table D.5. Adding VPCs degrades the average performance of `Gurobi` for each partial tree size. Of course, this rule is just one simple idea for choosing a partial tree per instance and does not exclude other, more intelligent rules.

An alternative to such an oracle approach would be to run seven concurrent processors,

Table 4.3: Number of leaf nodes yielding the best result for each experiment per instance.

	Gap				Time	
	V	V+G	V+GurF	V+GurL	All	6 trees
No improvement	9	9	41	68	70	40
2 leaves	4	5	46	14	17	7
4 leaves	8	11	10	18	8	5
8 leaves	20	23	14	13	19	12
16 leaves	30	35	16	15	12	6
32 leaves	27	27	15	14	19	15
64 leaves	86	74	42	42	14	12

each one with a different partial tree setting, and to stop when the first of these processes solves the problem to optimality. A concurrent option is present in `Gurobi`, for which the current default is to give each of the allocated processors a different random seed. The comparison of "V" to "Gur7" can be seen as testing whether it is worth replacing random seeds with more aggressive cut generation as a source of variability. Such a comparison only really makes sense for the set "6 trees" (as we do not have statistics for seven runs for the other instances). What we see from Table 4.2, as well as in Figure 4.8, is that for easier instances (those that solve in under ten minutes), using VPCs seems to beat the random seeds strategy with consistency. On the other hand, for the harder instances (on a very small number of instances), better performance is provided by "Gur7". One explanation for this is that harder instances are known to have more variability, which is corroborated by our experiments, and random seeds may introduce more randomness than adding VPCs. Another explanation is that solvers often employ more expensive cut generation techniques for harder instances, such as generating globally valid cuts after the root node, which could reduce the advantage of our stronger cuts. One technique that is employed by solvers is producing mixed-integer rounding cuts where the point being separated is not $\bar{x}$, but rather the optimal solution to a deeper node of the branch-and-bound tree, though the separation is still done using the initial relaxation to maintain global validity of the cuts.

4.4.3 Partial tree size and objective function analysis

One aspect that is hidden in the preceding results is the number of leaf nodes used for the partial branch-and-bound tree to obtain the reported gap closed and branch-and-bound times. What is reported is the best result across all the tested sizes of the partial tree, i.e., with number of leaf nodes $\ell \in \{2, 4, 8, 16, 32, 64\}$. We next disaggregate the analysis to see how the different size partial trees perform alone.

Table 4.3 gives the distribution for which choice of partial tree was best for each of the experiments. For each instance, we look at the appropriate metric and take the size of the smallest partial tree that led to the best result for that metric, assuming that the best result led to an improvement over the relevant baseline for that metric. To determine improvement, we use the same tolerances as before: the gap closed is counted as improved when it is at

Table 4.4: Statistics about the density of generated cuts broken down by partial tree size.

	V (2)	V (4)	V (8)	V (16)	V (32)	V (64)
# inst w/VPCs and time < 3600s	155	141	134	131	118	109
# wins by time	46	26	37	39	37	36
Avg min cut density	0.280	0.288	0.343	0.410	0.436	0.464
Avg max cut density	0.433	0.455	0.523	0.551	0.561	0.565
Avg avg cut density	0.363	0.371	0.432	0.491	0.516	0.525
Avg avg cut density (win by time)	0.356	0.316	0.352	0.435	0.508	0.496
Avg avg cut density (non-win)	0.366	0.383	0.462	0.515	0.520	0.540

least 10^{-7} higher, and the time is considered improved when the instance solves at least 10% more slowly without using VPCs. The first column is the tree size (including the option that VPCs yielded no improvement), the second through fifth columns refer to gap closed (for the set of 184 instances with $c^\mathsf{T}\underline{p} > c^\mathsf{T}\bar{x}$), while the sixth and seventh columns concern branch-and-bound time for the sets "All" and "6 trees" (where improvement is counted against the "Gur1" baseline). As mentioned in the context of Figure 4.5, the tree with the most leaf nodes produces the best percent gap closed quite often, but not always; there are even instances for which $\ell = 2$, i.e., just a single split disjunction, yields the strongest cuts. In contrast, there is no single winner for best partial tree size in terms of branch-and-bound time, as we had already discussed in the context of Table D.5.

This phenomenon of stronger cuts not being directly correlated to better branch-and-bound performance is not easy to explain, due to the hard-to-predict effect of cuts on the branch-and-bound process. Nevertheless, there is one aspect of Table 4.2 that suggests a possible explanation. Observe that the number of times that using VPCs beats "Gur1" and "Gur7" in terms of number of nodes is typically higher than the number of wins for VPCs in terms of time. For each instance in which the number of nodes decreases but the time increases, what happens is that the time per node increases. There are several possible reasons for this. One is that cut filtering, i.e., deciding which cuts to apply at each node, could be slower when there are more cuts in the potential cut pool. Another possible reason is that the solving the relaxation at each node of the branch-and-bound tree could be slower with additional cuts added. It is commonly known that making the coefficient matrix denser will slow down the matrix operations needed to solve a linear program. Consequently, we next look at how the density of VPCs changes as deeper disjunctions are used.

Table 4.4 gives the average cut density statistics across the different partial tree sizes. The density of a cut is defined as the number of nonzero cut coefficients divided by the total number of coefficients. In this table, the columns are labeled $V(\ell)$ for $\ell \in \{2, 4, 8, 16, 32, 64\}$, referring to the runs in which the number of leaf nodes is ℓ. The set "All' of 159 instances is the one used to generate this table. The first data row is the number of these instances having VPCs for each partial tree size. The second data row gives the number of these instances that win on time compared to "Gur1". The next three rows give the average of the minimum, maximum, and average densities of VPCs for each instance. The penultimate row is the average of the average cut densities for those instances counted in the second data

Table 4.5: Statistics about the objectives leading to failures, broken down by partial tree size used for cut generation.

	V (2)	V (4)	V (8)	V (16)	V (32)	V (64)
# inst w/obj	191	174	168	164	149	129
# inst w/succ obj	191	174	166	162	145	126
# inst no obj	4	21	27	31	46	66
# inst all obj fail	0	0	2	2	4	3
# inst all obj succ	13	12	6	6	9	5
% obj fails	22.19	25.07	27.81	32.46	33.81	35.16
% fails dup	43.87	41.76	46.69	52.51	60.21	64.05
% fails unbdd	44.06	42.54	33.39	25.51	18.13	18.69
% fails tilim	2.09	4.85	10.86	13.80	11.50	9.43
% fails dyn	3.18	3.95	5.40	4.45	3.98	3.95
% fails all-ones	56.02	59.77	48.81	46.34	36.24	22.48
% fails post-GMIC opt	60.00	62.64	66.67	58.90	54.05	51.56
% fails DB	20.70	21.00	24.06	29.30	31.31	32.88
# obj / cut	2.26	1.93	2.08	2.56	2.87	3.34
(s) / obj	0.01	0.07	0.19	1.49	1.40	4.35
(s) / cut	0.03	0.10	0.28	3.38	6.65	23.25

row, i.e., those that win by time with respect to "Gur1". The last row is the average of the average cut densities for the instances that do not win by time.

The first observation is that the number of instances having cuts decreases from 155 for the setting with 2 leaf nodes to 109 for the 64 leaf node setting. The average cut density goes from 0.363 to 0.525 on average, which could help explain the fact that the cuts from the strogner disjunctions, which we have seen are stronger, yield worse branch-and-bound times. In addition, the last two rows show that there is a clear gap between those instances for which VPCs help and those for which they do not. Namely, for each column, the average of the average cut density is always lower for those instances that win by time. This suggests that future experiments may benefit from doing cut filtering by density, or possibly reducing the density of generated cuts a posteriori while sacrificing strength.

Next, via Table 4.5, we discuss our objective function choices, including statistics on the frequency of failures as a function of disjunction size. A caveat is that our analysis of objectives functions is somewhat limited by our relatively conservative strategy in selecting objectives to use, in order to limit time spent on cut generation and the types of failures discussed in Section 4.2.5. The objectives we choose utilize the results of Chapter 2, which show that a successful class of objectives is the set of points and rays of the point-ray collection, but, motivated by Proposition 4.12, we only do this for cuts tight at $\underline{p}$ (the loop starting at step 17 of Algorithm 4.6). It is natural to consider cuts that only lie on deeper points, but this, as well as cuts that are satisfied by $\bar{x}$, remains a topic for future research.

Table 4.6: Statistics about when generated cuts are active, broken down by partial tree size.

	V+G (2)	V+G (4)	V+G (8)	V+G (16)	V+G (32)	V+G (64)
% active GMIC	50.69	49.21	48.48	45.20	43.99	43.66
% active VPC	34.00	33.56	33.10	35.46	34.77	34.30
% active one-sided	81.82	81.82	81.82	81.82	81.82	81.82
% active all-ones	59.52	64.29	58.14	56.82	53.68	56.00
% active post-GMIC opt	80.26	75.38	71.43	62.69	60.29	62.90
% active DB	22.10	22.21	23.04	27.25	28.69	29.27

Table 4.5 contains the data for objective failures using the entire set of 195 instances. The columns of the table are the same as Table 4.4. The rows are divided into several sets. The first set gives statistics on the number of instances for each column for which: (1) objectives were tried, (2) VPCs were generated, (3) objectives were not tried, (4) none of the objectives yielded VPCs, and (5) all of the objectives led to distinct VPCs. The next set gives the average percent of the objectives that were failures. The subsequent set of rows looks at the cause of these failures, which fall into one of four categories:

- "Dup": the optimal solution to the PRLP is an exact duplicate of an existing cut
- "Unbdd": (PRLP) does not have a finite solution for that objective
- "Tilim": the time limit for (PRLP) is attained for that objective
- "Dyn": the dynamism of the new cut is too high

The following set of rows looks at the percent of failures for each class of objectives: "all-ones" ($w = e$), "post-GMIC" (step 11 of Algorithm 4.6), and "DB" (step 17 of Algorithm 4.6). The last set of rows looks at the average number of objectives required to generate each distinct VPC, as well as the average number of seconds taken per objective and per cut.

This table shows that failures become more frequent when using disjunctions with more terms, with average failure rate increasing from 22% to 35%. The primary reason for this is that more objectives lead to cuts that were previously generated, leading to, on average, 64% of the failures. The cause is that there are more unsuccessful objectives being tried for the "DB" class of objectives. The last set of rows of the table also show that cut generation can be extremely costly for stronger disjunctions, which could be mitigated by reducing the failure rate. In Appendix D.5, Table D.6 contains the statistics for the number of objectives tried and failures just for the best run (across all partial tree sizes) per instance, as well as the time per objective and per cut.

Lastly, we look at which classes of objective functions are more likely to lead to active cuts (after the addition of all cuts), as a different measure of the effect of our cuts. Table 4.6 gives averages for which cuts are active at the optimal solution after adding both GMICs and VPCs to P, for GMICs and VPCs, as well as individually based on the objective producing each VPC. The first row is the percent of GMICs that are active. The second row is the percent of VPCs that are active, averaged across those instances per each column for which

VPCs were generated. The next row concerns an additional set of cuts that is added in our procedure, which we call "one-sided". In the process of generating VPCs, while selecting variables for strong branching, we occasionally detect that one of the two possible branches is infeasible. In this case, we generate the "one-sided cut" (from the variable x_k, $k \in I$) $x_k \le \lfloor \bar{x}_k \rfloor$ if $P \cap \{x : x_k \ge \lceil \bar{x}_k \rceil\}$ is infeasible, or $x_k \ge \lceil \bar{x}_k \rceil$ if the opposite term is infeasible. These cuts are generated for 11 of the 195 instances, with a total of only 17 cuts. The last three rows of the table give the percent of active cuts for the three classes of objectives that we test. For each of the sources of VPCs, the value in each column is averaged only across those instances for which there exist cuts from that source.

Table 4.6 shows that the proportion of active VPCs remains relatively constant across the different partial tree sizes, but that the percent of active GMICs decreases as VPCs from stronger disjunctions are used. Aside from the one-sided cuts, of which an average of 82% are active across all partial trees tested, the objectives "all-ones" and "post-GMIC opt" lead to cuts that are frequently active, though these objectives yield at most two cuts in total per instance. Though the "DB" class of objectives leads to a smaller percentage of active cuts, it yields increasingly more active cuts as stronger disjunctions are utilized, and, of course, it is the source for the majority of the cuts that we generate.

4.5 Alternative cut-generating sets and point-ray collections

In this section, we briefly mention experiments with other choices for cut-generating sets (instead of those derived from partial branch-and-bound trees) and with potential refinements of the simple point-ray collection used for all of the previous results.

4.5.1 Gap closed using multiple split and cross disjunctions

Instead of using one large, multi-term disjunction, as we do above, the typical approach in the literature is to generate cuts from the union of several shallower disjunctions. We report results on preliminary computational experiments to assess the strength of VPCs obtained from multiple split disjunctions or multiple 2-branch (cross) disjunctions. An alternative that we do not test but merits exploration in the future is that of several partial branch-and-bound trees produced from different branching strategies.

Let $\sigma := \{j \in I : \bar{x}_j \notin I\}$ be the set of indices of integer variables that take fractional values in $\bar{x}$. For each $k \in \sigma$, there is a corresponding elementary split disjunction $(x_k \le \lfloor \bar{x}_k \rfloor) \vee (x_k \ge \lceil \bar{x}_k \rceil)$. We generate VPCs from each of the elementary split disjunctions applied to P. We also report on the strength of VPCs from the nonconvex P_I-free set corresponding to a union of two split disjunctions from pairs of indices in σ.

We experiment on a smaller set of instances (37 in total) in order to conserve computational resources, and we report only the percent gap closed (without testing the cuts' effect on branch-and-bound time). The reason for both of these restrictions is that these early experiments strongly support the use of partial trees as the cut-generating set over multiple split or cross disjunctions. The size restriction for the experiments in this section is at most 500 rows and 500 columns. The other instance selection criteria remain unchanged. The

Table 4.7: Summary on small instance set of average percent gap closed, average percent of GMICs and VPCs active at the optimum after adding both sets of cuts to P, average "cut ratio" between number of VPCs and number of GMICs, and the shifted geometric mean (by 60) for generating VPCs for each partial tree size as well as on runs with multiple split and cross disjunctions.

	G	V+G (2)	V+G (4)	V+G (8)	V+G (16)	V+G (32)	V+G (64)	V+G (splits)	V+G (crosses)
% gap closed	16.33	18.48	19.55	21.73	25.52	30.74	35.26	21.79	26.92
% active GMIC		50.40	49.68	45.01	40.11	35.37	36.90	45.65	35.82
% active VPC		22.09	22.59	30.57	29.81	29.26	27.90	14.19	10.45
Cut ratio		0.63	0.64	0.66	0.69	0.71	0.72	14.27	108.23
Time (geomean)		0.16	0.54	2.34	7.24	13.16	14.43	7.38	53.40

limit on the number of cuts per split or cross disjunction is set as $|\sigma|$ (which is the same as the limit from each partial branch-and-bound tree).

Table 4.7 has columns for "G" (GMICs), "V+G (ℓ)" for $\ell \in \{2, 4, 8, 16, 32, 64\}$ (VPCs used together with GMICs from a partial branch-and-bound tree with ℓ leaf nodes), and "V+G (splits)" and "V+G (crosses)" (values corresponding to using splits and crosses, respectively). The rows give the average percent gap closed, the average percent of active GMICs and VPCs at the post-cut optimum, the ratio between the number of VPCs and number of GMICs, and the geometric mean (with a shift of 60) of the time needed to generate cuts (including the time to generate the partial trees and set up the point-ray collections).

As Table 4.7 shows, the gap closed by VPCs from multiple split disjunctions is comparable to that from using a partial tree with 8 leaves, while multiple cross disjunctions yield a gap closed similar to that from partial trees with 16 leaves. However, when using splits and crosses, the number of VPCs is considerably larger than the number of GMICs, and cut generation time is also on average much greater. This data supports our conclusion that using partial branch-and-bound trees to generate disjunctions for our procedure is preferable to using multiple split or cross disjunctions.

4.5.2 Tightening the $\mathcal{V}$-polyhedral relaxation

We have seen in Figures 4.2 and 4.3 that using the relaxations $C(\mathcal{N}(p^t))$ for each term $t \in \mathcal{T}$ can limit the set of cuts that can be generated. A natural question to consider is whether a different relaxation would lead to stronger cuts.

One approach, which we have not tested computationally, involves refining the relaxations of each disjunctive term to some $C^t \subseteq C(\mathcal{N}(p^t))$ for each $t \in \mathcal{T}$ such that the following condition is satisfied for all $t, t' \in \mathcal{T}$, $t \neq t'$:

$$C^t \cap \{x \in \mathbb{R}^n : D^{t'} x \geq D_0^{t'}\} = \emptyset.$$

This would avoid the type of problem shown in Figure 4.3. The essential idea would involve activating hyperplanes, but the process is generally made simpler by the fact that the disjunctive inequalities in practice all take on a simple form (each is a bound on a variable).

A different idea is to keep a simple cone as the relaxation for each term of the disjunction, but to use a different cobasis as the origin. Specifically, we can apply the VPC procedure based on Lemma 4.7, not on p^t, but on a neighbor of p^t obtained by pivoting along any edge of P^t. We tested this procedure on VPCs generated from the set of elementary split disjunctions. The results were negative, in the sense that only a small additional percent gap was closed, whereas the extra computational expense involved was significant. Our interpretation of this outcome is that the VPCs from simple cones contain the vast majority of the cuts that affect the objective function value and are obtainable from each elementary split. This was in fact a primary motivation for pursuing more complicated disjunctions for cut generation.

4.6 Conclusion & open problems

This chapter presents a step toward merging cut-generation and branching in integer programming solvers by providing a computationally tractable method for generating cuts from partial branch-and-bound trees. The framework we introduce is to (1) select a disjunction, (2) choose a (compact) $\mathcal{V}$-polyhedral relaxation for each disjunctive term, and (3) selectively generate cuts by judiciously choosing objective directions to optimize over (PRLP) formed from the point-ray collection.

Our investigation touches each of these aspects. We test the traditional disjunctions used for cut generation (elementary split disjunctions and cross disjunctions), but we find that a disjunction produced from a branch-and-bound tree yields far better cuts (despite the fact that this is merely one disjunction, compared to the $O(n)$ elementary split disjunctions or $O(n^2)$ cross disjunctions). The quality of the cuts we obtain is closely tied with properties of the partial branch-and-bound tree. An open problem that remains is how to assess when to stop the partial branch-and-bound process and turn to cut generation.

The relaxation for each disjunctive term that we use is quite simple, but therein lies its advantage. We experimented with tighter relaxations, but the marginal improvement in gap closed was relatively little, and the tightening was expensive. However, we show examples highlighting the weakness of our simple relaxations, that only a subset of all the valid disjunctive cuts can be produced, and the computational results do, at times, reflect this weakness. Thus, there is an opportunity to improve the quality of the generated VPCs by considering tighter relaxations generated from structural information about each instance.

For the objective directions, we provide theoretical support for objective directions to (PRLP) that yield new and strong VPCs more frequently than previous approaches (reducing the percent of objectives failing to produce a cut from 80% in early experiments, to around 30% in the current implementation).

Overall, our computational results indicate that the cuts we generate are strong, as evidenced by the percent integrality gap they close (compared to both GMICs and the default cut setting of `Gurobi`). Moreover, we generate VPCs non-recursively, based only on logical conditions imposed on P and without implications from other cuts, and the integrality gap they close increases steadily with the use of stronger disjunctions, avoiding the common tailing off of strength experienced by other cut families that require recursive applications to reach strong cuts. In addition, for some instances, our results show that the extra computational effort pays off in reduced branch-and-bound time.

Last, but not least, though our experiments focus on generic mixed-integer programs, there seem to exist classes of instances for which VPCs works particularly well. We have not yet been successful in identifying what characteristics of an instance lend it to be particularly amenable to our framework, but there exists the possibility for future specialization of the VPC framework given special problem structure.

There are several other challenges moving forward in order to make VPCs truly practical. Some of the remaining open questions are listed below.

1. We used full strong branching to generate the partial branch-and-bound tree; however, while this is typically the strategy that empirically yields the smallest branch-and-bound trees, it may not be the best one for generating useful VPCs. For example, it may be better in some cases to use a strategy for the partial tree that differs substantially from the default branch-and-bound strategy employed.

2. What is the best (from the perspective of resulting cuts) size of the partial tree?

3. Inequalities that cut away the LP optimal solution may not be the ones that help most in branch-and-bound, so there may be benefit to generating inequalities from (PRLP) scaled with $\beta = -1$.

4. The best number of cuts that should be generated for an instance remains unresolved.

5. The $\mathcal{V}$-polyhedral relaxation we use is very simple; what are more complicated relaxations that can be used, and when do they merit the additional cost of obtaining them (versus, for example, simply generating a larger partial branch-and-bound tree)?

6. Can VPCs be strengthened in an efficient way?

7. A partial branch-and-bound tree may lend some partial information about the structure of the instance that can be used in many more ways than merely generating cuts, such as informing a better branching strategy. This has been partially explored in the past [85], but can the technique be effectively combined with cut generation?

Chapter 5

Towards a Correspondence Between $\mathcal{V}$-Polyhedral Cuts and Lift-and-Project Cuts

This chapter is based on joint work with Egon Balas.

5.1 Introduction & preliminaries

In this chapter, using a combination of polarity and duality arguments, we make a connection between *$\mathcal{V}$-polyhedral cuts* (VPCs) and *lift-and-project cuts* (L&PCs), by mapping a correspondence between feasible solutions to the *point-ray linear program* (PRLP) as in Chapter 4 and to the *cut-generating linear program* (CGLP) from Section 1.4.[1]

We work with the following disjunction

$$\bigvee_{t \in \mathcal{T}} \{x \in \mathbb{R}^n : A^t x \geq b^t\}.$$

The system $A^t x \geq b^t$ could, as before, denote the inequalities of P along with the inequalities of term t of a valid disjunction (1.1), but we also allow the more general case, such as the setting of Chapter 4, where each term is relaxed to be the simple polyhedral cone arising from a basic relaxation, in which case $A^t x \geq b^t$ would consist of the inequalities corresponding to nonbasic variables at a vertex for term t. Let $P^t := \{x \in \mathbb{R}^n : A^t x \geq b^t\}$ (which we assume is nonempty) be the $\mathcal{H}$-polyhedral (inequality) description of disjunctive term $t \in \mathcal{T}$, and let $(\mathcal{P}^t, \mathcal{R}^t)$ denote the equivalent $\mathcal{V}$-polyhedral description, i.e., the set of extreme points and rays of P^t, so that $P^t = \operatorname{conv}(\mathcal{P}^t) + \operatorname{cone}(\mathcal{R}^t)$. Let $P_D := \operatorname{cl}\operatorname{conv}(\cup_{t \in \mathcal{T}} P^t)$ with extreme points and rays given by the collection $(\mathcal{P} := \cup_{t \in \mathcal{T}} \mathcal{P}^t, \mathcal{R} := \cup_{t \in \mathcal{T}} \mathcal{R}^t)$. For $t \in \mathcal{T}$, let m'_t denote the number of rows of A^t. We first informally summarize some important disjunctive programming concepts and the two cut-generating paradigms that we are relating.

[1]We derive many of our results from first principles, as we find it more intuitive, but the interested reader can consult the texts [36, 98] for general principles of polarity and duality, as well as [140] in the context of polyhedra. An overview of convex analysis that pertains to disjunctive cuts, and a more formal technical treatment of disjunctive programming, is given by Balas [17, 19]; see also the paper by Cornuéjols and Lemaréchal [62].

As discussed in Section 1.4, one way to generate valid cuts for P_D is through the CGLP, which is an application of *disjunctive programming duality* [17, Section 4]. To recap, the essence of the technique is that an inequality $\alpha^\intercal x \geq \beta$ is valid for P_D if and only if the inequality is valid for each P^t, $t \in \mathcal{T}$. Consequently, by Farkas' lemma, $\alpha^\intercal x \geq \beta$ is valid for P_D if and only if the following system is feasible, in variables $(\alpha, \beta, \{v^t\}_{t \in \mathcal{T}})$:

$$\left.\begin{array}{l} \alpha^\intercal = v^t A \\ \beta = v^t b^t \\ v^t \in \mathbb{R}^{m'_t}_{\geq 0} \end{array}\right\} \text{ for all } t \in \mathcal{T}. \tag{5.1}$$

Note that a straightforward application of Farkas' lemma would produce $\beta \leq v^t b^t$ constraints. We enforce equality, which is without loss of generality if either all the constraints $Ax \geq b$ are subsumed by $A^t x \geq b^t$ (see Proposition 5.11), or if we assume the slack variable for the β constraint is accounted for in each $A^t x \geq b^t$ system via the redundant inequality $\mathbf{0}^\intercal x \geq -1$.

A convenient geometric view is obtained by rewriting the set of requirements (5.1) as

$$(\alpha, \beta) \in \bigcap_{t \in \mathcal{T}} \operatorname{cone}\left(\{(A^t_{i\cdot}, b^t_i)\}_{i \in m'_t}\right).$$

The vector $-A^t_{i\cdot}$ is called the *normal* to constraint $i \in [m'_t]$, which is the vector perpendicular to $A^t_{i\cdot} x = b^t_i$ in the direction that does *not* belong to the halfspace. The geometric insight from this interpretation is that $\alpha^\intercal x \geq \beta$ is valid for P^t and supporting at a point $p^t \in P^t$ if and only if $-\alpha$ lies in the *normal cone* at p^t, formed from the normals to constraints of P^t that are tight at p^t.[2] Hence, $-\alpha$ will lie in the intersection of the normal cones obtained from each disjunctive term.

An alternative way to generate disjunctive cuts is through the *reverse polar* of P_D [17, Section 5], which is defined with respect to a given $\bar{\beta} \in \mathbb{R}$ as

$$\{\alpha \in \mathbb{R}^n : \alpha^\intercal x \geq \bar{\beta} \text{ for all } x \in P_D\}.$$

Clearly this captures all of the valid inequalities for P_D having right-hand side $\bar{\beta}$.[3] Since $x \in P_D$ if and only if $x \in \operatorname{conv}(\mathcal{P}) + \operatorname{cone}(\mathcal{R})$, it holds that $\alpha^\intercal x \geq \bar{\beta}$ is valid for P_D if and only if it is satisfied by all of the points and rays in $(\mathcal{P}, \mathcal{R})$. This yields the system (PRLP), in variables $\alpha \in \mathbb{R}^n$, with respect to a fixed $\bar{\beta}$

$$\begin{array}{ll} \alpha^\intercal p \geq \bar{\beta} & \text{for all } p \in \mathcal{P} \\ \alpha^\intercal r \geq 0 & \text{for all } r \in \mathcal{R}. \end{array} \tag{PRLP($\bar{\beta}$)}$$

This yields exactly the same facets of P_D as (5.1) normalized by $\beta = \bar{\beta}$:

$$\left\{(\alpha, \{v^t\}_{t \in \mathcal{T}}) : (\alpha, \bar{\beta}, \{v^t\}_{t \in \mathcal{T}}) \text{ is feasible to (5.1)}\right\}. \tag{CGLP($\bar{\beta}$)}$$

[2]Formally, the normal cone at p^t is the closed convex cone of vectors forming obtuse angles with every ray $x - p^t$, $x \in P^t$, i.e., $\{\alpha \in \mathbb{R}^n : \alpha^\intercal (x - p^t) \leq 0 \text{ for all } x \in P^t\}$.

[3]Note again the relationship to the normal cone. If a cut $\bar{\alpha}^\intercal x \geq \bar{\beta}$ is valid for P^t, then $\bar{\alpha}$ belongs to the reverse polar as defined. If the cut is supporting at $p^t \in P^t$, then $\bar{\alpha}^\intercal p^t = \bar{\beta}$, which, plugging into the definition, means that $-\bar{\alpha}$ belongs to the normal cone at p^t.

The advantage of using (PRLP) over (CGLP) is the absence of the Farkas multipliers as variables, which implies that the optimization problem we solve to generate a cut is in the original dimension of the problem, rather than in a lifted space. This is much more efficient and enables us to test generating disjunctive cuts from disjunctions with many terms, as in Chapter 4. Moreover, while the two cut-generating approaches are theoretically equivalent (both contain all the facet-defining inequalities for the disjunctive hull; see Corollary 4.5), we showed that by working in the nonbasic space, setting $\bar{\beta} = 1$, and selecting objective functions judiciously, we can efficiently generate disjunctive cuts that are violated by $\bar{x}$. This β normalization can also be applied to the CGLP (as in (CGLP($\bar{\beta}$))), but most commonly other normalizations are used, which may lead to cuts from the CGLP that are not even supporting for the disjunctive hull [84]. VPCs thus have an immediate advantage over L&PCs.

There is, however, a disadvantage to the (PRLP) formulation, which is precisely that we do not obtain the Farkas multipliers, and hence we cannot directly apply the cut strengthening techniques described in Section 1.5. Thus, the practical motivation for pursuing a constructive correspondence between feasible solutions to (PRLP) and feasible solutions to (CGLP) is having a way to both efficiently generate and strengthen disjunctive cuts.

In Section 5.2, we start with an overview of several properties that are useful towards establishing the correspondence. Section 5.3 then describes the interesting direction of the correspondence, starting with a feasible cut generated via the PRLP and ending with a basis for the CGLP. Section 5.4 goes through the details of a complete example.

5.2 Relationship between $\mathcal{H}$- and $\mathcal{V}$-polyhedral descriptions

In this section, we state a few key properties relating valid cuts for P_D to the $\mathcal{H}$- and $\mathcal{V}$-polyhedral descriptions of P_D. As the discussion is notation-heavy, we provide Table 5.1 as a reference, ordered alphabetically.

Let $\bar{\alpha}^\intercal x \geq \bar{\beta}$ be a valid inequality for P_D. Let $(\bar{\alpha}, \{\bar{v}^t\}_{t \in \mathcal{T}})$ be a corresponding basic feasible solution to (CGLP($\bar{\beta}$)).[4] Let J^t be the indices of the basic variables for term $t \in \mathcal{T}$:

$$J^t := \{i \in [m'_t] : v^t_i \text{ is basic in } \{\bar{v}^t\}_{t \in \mathcal{T}}\}.$$

From J^t, an important subset is the set of indices of variables taking strictly positive values:

$$J^{t+} := \{i \in J^t : \bar{v}^t_i > 0\}.$$

For $t \in \mathcal{T}$ and $N \subseteq [m'_t]$, let $A^t_N x \geq b^t_N$ be the constraints of P^t indexed by N and define

$$\bar{F}^t(N) := \{x \in \mathbb{R}^n : A^t_N x = b^t_N\},$$
$$F^t(N) := \{x \in P^t : A^t_N x = b^t_N\} = \bar{F}^t(N) \cap P^t.$$

[4]When $\bar{\beta} = 0$, $(\bar{\alpha}, \{\bar{v}^t\}_{t \in \mathcal{T}})$ actually corresponds to an extreme ray of (CGLP($\bar{\beta}$)), which is not basic. In this case, the "basis" will refer to the basis of the vertex at which the ray originates (which is the origin).

Table 5.1: Reference for notation used in this chapter, listed alphabetically.

Symbol	Definition	Comments
$(\bar{\alpha}, \bar{\beta}, \{\bar{v}^t\}_{t \in \mathcal{T}})$	Extreme ray of (5.1)	
$A^t x \geq b^t$	Inequalities defining term $t \in \mathcal{T}$	
$\bar{\beta}_t$	$\bar{\beta}_t := \min_x \{\bar{\alpha}^\mathsf{T} x : x \in P^t\}$	Not equal to $\bar{\beta}$ for terms $t \in \mathcal{T} \setminus \mathcal{T}^*$
$\bar{F}^t(N)$	$\bar{F}^t(N) := \{x \in \mathbb{R}^n : A^t_N x = b^t_N\}$	Feasible region defined by inequalities indexed by $N \subseteq [m'_t]$
$F^t(N)$	$F^t(N) := \{x \in P^t : A^t_N x = b^t_N\}$	Face of P^t defined by inequalities indexed by $N \subseteq [m'_t]$
F^{t*}	$\operatorname{conv}(\mathcal{P}^{t*}) + \operatorname{cone}(\mathcal{R}^{t*})$	Face of P^t exposed by the cut
J^t	$\{i \in [m'_t] : v^t_i \text{ is basic in } \{\bar{v}^t\}_{t \in \mathcal{T}}\}$	Indices of basic variables for term t
J^{t+}	$\{i \in J^t : \bar{v}^t_i > 0\}$	Indices of the strictly positive variables for term $t \in \mathcal{T}$
$N^t(x)$	$\{i \in [m'_t] : A^t_{i\cdot} x = b^t_i\}$	Indices of the inequalities of P^t tight for $x \in \mathbb{R}^n$
$N^t(p, r)$	$\bigcap_{x \in \{p + \lambda r : \lambda \in \mathbb{R}_{\geq 0}\}} N^t(x)$	Indices of the inequalities tight for all points on ray $r \in \mathbb{R}^n$ originating at p
N^{t*}	$\bigcap_{x \in P^t : \bar{\alpha}^\mathsf{T} x = \bar{\beta}_t} N^t(x)$	Indices of the inequalities tight for every element on the face of P^t exposed by $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}_t$
P^t	$P^t := \{x \in \mathbb{R}^n : A^t x \geq b^t\}$	Disjunctive term $t \in \mathcal{T}$
$(\mathcal{P}^t, \mathcal{R}^t)$	Extreme points and rays of P^t	
$(\mathcal{P}^{t*}, \mathcal{R}^{t*})$	$\mathcal{P}^{t*} := \{p \in \mathcal{P}^t : \bar{\alpha}^\mathsf{T} p = \bar{\beta}\}$, $\mathcal{R}^{t*} := \{r \in \mathcal{R}^t : \bar{\alpha}^\mathsf{T} r = 0\}$	Extreme points and rays of P^t that are tight for the cut
$\mathcal{T}^*$	$\{t \in \mathcal{T} : \exists p^t \in P^t \text{ s.t. } \bar{\alpha}^\mathsf{T} p^t = \bar{\beta}\}$	Indices of the disjunctive terms that the cut is tight on

We first state a simple relationship between J^{t+} and the indices of the tight inequalities of P^t at a given $\hat{x} \in \mathbb{R}^n$, using the complementary slackness condition $(A^t x - b^t)v^t = 0$.[5]

Proposition 5.1. *For any $t \in \mathcal{T}$ and $\hat{x} \in \mathbb{R}^n$ such that $A^t_{i\cdot}\hat{x} \geq b^t_i$ for all $i \in J^{t+}$, $\bar{\alpha}^\mathsf{T}\hat{x} = \bar{\beta}$ if and only if $A^t_{i\cdot}\hat{x} = b^t_i$ for all $i \in J^{t+}$, i.e., $\{x \in \mathbb{R}^n : \bar{\alpha}^\mathsf{T} x = \bar{\beta}\} = \bar{F}^t(J^{t+})$.*

Proof. We have that $\bar{v}^t A^t = \bar{\alpha}$ and $\bar{v}^t b^t = \bar{\beta}$.

$\Longleftarrow$ Since $A^t_{i\cdot}\hat{x} = b^t_i$ for all $i \in J^{t+}$ and $\bar{v}^t_i = 0$ for all $i \notin J^{t+}$,

$$\bar{\alpha}^\mathsf{T}\hat{x} = \bar{v}^t A^t \hat{x} = \sum_{i \in J^{t+}} \bar{v}^t_i A^t_{i\cdot}\hat{x} = \sum_{i \in J^{t+}} \bar{v}^t_i b^t_i = \bar{v}^t b^t = \bar{\beta}.$$

$\Longrightarrow$ We are given that $\bar{\alpha}^\mathsf{T}\hat{x} = \bar{\beta}$. Then

$$\bar{\beta} = \bar{\alpha}^\mathsf{T}\hat{x} = \bar{v}^t A^t \hat{x} = \sum_{i \in J^{t+}} \bar{v}^t_i A^t_{i\cdot}\hat{x} \geq \sum_{i \in J^{t+}} \bar{v}^t b^t_i = \bar{v}^t b^t \geq \bar{\beta}.$$

It follows that $A^t_{i\cdot}\hat{x} = b^t_i$ for all $i \in J^{t+}$. □

Let $\mathcal{T}^*$ denote the index set for the disjunctive terms for which the cut is supporting, i.e., for which there exists a point in P^t tight for $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$:

$$\mathcal{T}^* := \{t \in \mathcal{T} : \text{there exists } p^t \in P^t \text{ for which } \bar{\alpha}^\mathsf{T} p^t = \bar{\beta}\}.$$

Let $(\mathcal{P}^{t*}, \mathcal{R}^{t*})$ denote the points and rays from $(\mathcal{P}^t, \mathcal{R}^t)$ that are tight for the cut. That is, $p \in \mathcal{P}^{t*}$ if and only if $p \in \mathcal{P}^t$ and $\bar{\alpha}^\mathsf{T} p = \bar{\beta}$, and $r \in \mathcal{R}^{t*}$ if and only if $r \in \mathcal{R}^t$ and $\bar{\alpha}^\mathsf{T} r = 0$.

Denote the face of P^t defined (exposed) by the cut by

$$F^{t*} := \{x \in P^t : \bar{\alpha}^\mathsf{T} x = \bar{\beta}\} = \operatorname{conv}(\mathcal{P}^{t*}) + \operatorname{cone}(\mathcal{R}^{t*}).$$

Lemma 5.2. *If $t \in \mathcal{T}^*$, then* $\operatorname{rank}(A^t_{J^t}) = |J^t|$.

Proof. By definition of a basis, the columns indexed by J^{t+} are linearly independent, which means the vectors $\{(A^t_{i\cdot}, b^t_i)\}_{i \in J^{t+}}$ are linearly independent. Let $t \in \mathcal{T}^*$ and assume for the sake of contradiction that $\{A^t_{i\cdot}\}_{i \in J^t}$ are not linearly independent, so there exists $\lambda \neq 0$ such that $\lambda^\mathsf{T} A^t_{J^{t+}} = 0$. I claim that this implies that $\lambda^\mathsf{T} b^t_{J^{t+}} = 0$, which is a contradiction. Let $p \in P^t$ such that $\bar{\alpha}^\mathsf{T} p = \bar{\beta}$. By Proposition 5.1, $A^t_{i\cdot} p = b^t_i$ for all $i \in J^{t+}$. Hence, $0 = \lambda^\mathsf{T} A^t_{J^{t+}} p = \lambda^\mathsf{T} b^t_{J^{t+}}$. □

Lemma 5.3. *For $t \in \mathcal{T}^*$, $F^{t*} = F^t(J^{t+})$ and* $\dim(F^{t*}) \leq \dim(\bar{F}^t(J^{t+})) = n - |J^{t+}|$.[6]

Proof. The first statement is immediate by Proposition 5.1 and the definition of $F^t(J^{t+})$. For the second statement, observe first that $F^{t*} \subseteq \bar{F}^t(J^{t+})$. Next, when $\bar{F}^t(J^t) \neq \emptyset$, $\bar{F}^t(J^{t+})$ has dimension $n - \operatorname{rank}(A^t_{J^{t+}}) = n - |J^{t+}|$, where the equality is due to Lemma 5.2. □

[5] In the language of convex analysis, if $\hat{x} \in P^t$, Proposition 5.1 and Lemma 5.3 imply that $-\bar{\alpha}$ lies in the normal cone at $\hat{x}$ if and only if $\hat{x}$ lies on the face of P^t *exposed* by $\bar{\alpha}^\mathsf{T} x = \bar{\beta}$. See [98, Proposition 5.3.3].

[6] Using the convention that $\dim(\emptyset) = -1$.

The utility of Proposition 5.1 and Lemma 5.3 in the context of this chapter is clear. In one direction, it shows that a basic feasible solution to (CGLP($\bar{\beta}$)) provides the indices J^{t+} that we can map to the face of P^t exposed by the cut, which results in the set of points and rays that are tight for the cut, from which we can ultimately find a cobasis for (PRLP($\bar{\beta}$)).[7]

The converse direction, which is the one of interest, of finding J^{t+} from a given $(\mathcal{P}^{t*}, \mathcal{R}^{t*})$, requires us to take into account potential degeneracy. For any $x \in \mathbb{R}^n$, let

$$N^t(x) := \{i \in [m'_t] : A^t_{i\cdot} x = b^t_i\}$$

index the inequalities of P^t, $t \in \mathcal{T}$, that are tight at x. With this, for $x \in P^t$, we can rephrase Proposition 5.1 as $\bar{\alpha}^\mathsf{T} x = \bar{\beta}$ if and only if $J^{t+} \subseteq N^t(x)$. Slightly abusing notation, for a ray $r \in \mathbb{R}^n$ emanating from a point $p \in \mathbb{R}^n$, let

$$N^t(p, r) := \bigcap_{x \in \{p + \lambda r : \lambda \in \mathbb{R}_{\geq 0}\}} N^t(x).$$

For $t \in \mathcal{T}^*$, define

$$\begin{aligned}
N^{t*} &:= \bigcap_{x \in F^{t*}} N^t(x) \\
&= \{i \in [m'_t] : A^t_{i\cdot} x = b^t_i \text{ for all } x \in F^{t*}\} \\
&= \{i \in [m'_t] : A^t_{i\cdot} p = b^t_i \text{ for all } p \in \mathcal{P}^{t*},\ A^t_{i\cdot} r = 0 \text{ for all } r \in \mathcal{R}^{t*}\} \\
&= \bigcap_{p \in \mathcal{P}^{t*}} \left(N^t(p) \cap \bigcap_{r \in \mathcal{R}^{t*}} N^t(p, r) \right)
\end{aligned}$$

as the indices of the inequalities satisfied at equality by all elements of P^t that are tight for the cut, corresponding to the intersection of all the cobases of the points and rays in $(\mathcal{P}^{t*}, \mathcal{R}^{t*})$; these cobases can be saved when collecting the points and rays $(\mathcal{P}^t, \mathcal{R}^t)$.

We first state a lemma relating N^{t*} to J^{t+} and the dimension of the face exposed by the cut—namely, if N^{t*} and J^{t+} coincide, then the inequality in Lemma 5.3 holds at equality. Knowing this dimension can be useful; for example, it can be incorporated as a criterion for deciding which next cuts to pursue in the VPC framework. Note that we have an *upper bound* on the dimension of F^{t*} by $(\mathcal{P}^{t*}, \mathcal{R}^{t*})$. Indeed, $\dim(F^{t*}) \leq |\mathcal{P}^{t*}| + |\mathcal{R}^{t*}| - 1$, where equality will only hold when these points and rays are all affinely independent.[8]

Lemma 5.4. *For $t \in \mathcal{T}^*$, $n - |N^{t*}| \leq \dim(F^{t*}) \leq |\mathcal{P}^{t*}| + |\mathcal{R}^{t*}| - 1$. If $(\mathcal{P}^{t*}, \mathcal{R}^{t*})$ are affinely independent, then $\dim(F^{t*}) = |\mathcal{P}^{t*}| + |\mathcal{R}^{t*}| - 1$. If $J^{t+} = N^{t*}$, then $\dim(F^{t*}) = n - |J^{t+}|$.*

[7]This direction of the correspondence is obviously the easier one, and we do not really need the derived results, as $\bar{\alpha}$ is immediately feasible to (PRLP($\bar{\beta}$)), and the only remaining task is to identify a valid basis. However, if something other than the β normalization is used for the CGLP, there may exist basic feasible solutions that correspond to inequalities that are not facet-defining or even supporting for the disjunctive hull, which means they will not exist as basic feasible solutions to (PRLP). See the discussion in [84].

[8]This is actually satisfied in the context of the basis cone relaxations used in Chapter 4. The basis cones are *simplicial*, meaning every face of the cone is a simplex, which implies that all the points and rays incident to the face are affinely independent.

Proof. The upper bound is clear. Next, we have that $\dim(F^{t*}) = n - \operatorname{rank}(A^t_{N^{t*}}) \geq n - |N^{t*}|$. When $N^{t*} = J^{t+}$, $\operatorname{rank}(A^t_{N^{t*}}) = |N^{t*}| = |J^{t+}|$ by Lemma 5.2, proving the claim. □

The set of points and rays on the face of P^t exposed by the cut provide N^{t*}. To identify J^{t+} from this set, we have the following proposition.

Proposition 5.5. *For $t \in \mathcal{T}^*$, $F^{t*} = F^t(N^{t*})$ and $J^{t+} \subseteq N^{t*}$. If $i \in N^{t*}$ and $F^t(N^{t*} \setminus \{i\}) \neq F^{t*}$, then $i \in J^{t+}$. If $F^t(N^{t*} \setminus \{i\}) \neq F^{t*}$ for all $i \in N^{t*}$, then $J^{t+} = N^{t*}$.*

Proof. The statement $J^{t+} \subseteq N^{t*}$ is obvious by Lemma 5.3: for any $i \in J^{t+}$, $F^{t*} = F^t(J^{t+})$ implies that $A^t_{i\cdot}x = b^t_i$ for all $x \in F^{t*}$ (vacuously for $t \notin \mathcal{T}^*$), which means that $i \in N^{t*}$.

Let $i \in N^{t*}$ and consider $F' := F^t(N^{t*} \setminus \{i\}) \supseteq F^{t*}$. Suppose there exists $x^i \in F' \setminus F^{t*}$. This is a point that satisfies all but one of the inequalities of N^{t*} and for which $\bar{\alpha}^\mathsf{T} x^i > \bar{\beta}$, as $x^i \notin F^{t*}$. By Proposition 5.1, x^i must violate one of the inequalities indexed by J^{t+}, and since only i is removed from N^{t*}, a superset of J^{t+}, it must be that $i \in J^{t+}$. Thus, when all $i \in N^{t*}$ satisfy the given property and belong to J^{t+}, the equality $J^{t+} = N^{t*}$ holds. □

Corollary 5.6. *For $t \in \mathcal{T}^*$, if $\dim(\bar{F}^t(J^{t+})) = \dim(F^{t*})$, and $A^t x \geq b^t$ is irredundant, then $J^{t+} = N^{t*}$.*

Proof. The conditions imply that there are no additional implied equalities of P^t added to J^{t+} when intersecting $\bar{F}^t(J^{t+})$ with P^t, so $N^{t*} = J^{t+}$. □

As a consequence of Proposition 5.5, when we are constructing a feasible solution to (CGLP($\bar{\beta}$)) from a feasible solution to (PRLP($\bar{\beta}$)), the basic variables can be chosen from N^{t*}. In general, identifying the strictly positive variables requires testing for redundancy in the description of F^{t*}. We can avoid this step when $\{F^{t*}\}_{t \in \mathcal{T}^*}$ belong to a class of polyhedra called *simple*, which are precisely those that are not primal degenerate at any extreme point or ray, i.e., for which every extreme point and ray is defined by a unique basis [140]. Note that this is the situation for the basis cone relaxations used for VPCs in Chapter 4.

Definition 5.7. *An n-dimensional pointed polyhedron P is called* simple *if every extreme point of P is incident with exactly n facets of P and every extreme ray of P is incident with exactly $n-1$ facets of P.*

Corollary 5.8. *If $t \in \mathcal{T}^*$ and F^{t*} is simple, i.e., $|N^t(p)| = n$ for all $p \in \mathcal{P}^{t*}$ and $|N^t(r)| = n-1$ for all $r \in \mathcal{R}^{t*}$, then $J^{t+} = N^{t*}$.*

Proof. Let $d := \dim(F^{t*})$. As F^{t*} is simple, $|N^{t*}| = n - d$, i.e., F^{t*} is contained in exactly $n-d$ facets of P^t. (To see this, observe that $|N^{t*}| \geq n-d$ and $A^t_{N^{t*}}$ has full row rank because the constraints indexed by N^{t*} are a subset of the n linearly independent inequalities at any vertex of F^{t*}.) For any $i \in N^{t*}$, $F^t(N^{t*} \setminus \{i\})$ is defined by $n-d-1$ facets of P^t, so it is has one higher dimension than F^{t*}, meaning it contains a point of P^t that does not belong to F^{t*}. Since this holds for all $i \in N^{t*}$, by Proposition 5.5, $J^{t+} = N^{t*}$. □

Connection to irregular cuts. We end this section with a sufficient condition for identifying *irregular cuts* (introduced in Section 1.4.2) using a $\mathcal{V}$-polyhedral description of the disjunctive terms (complementing the $\mathcal{H}$-polyhedral characterization given by Balas and Kis [30]). Suppose that, for all $t \in \mathcal{T}$, the first m inequalities of $A^t x \geq b^t$ are the inequalities $Ax \geq b$ defining the original linear programming relaxation (LP). Let $K^t := J^t \cap [m]$ and K^{t+} denote the indices of the strictly positive multipliers. Define K as the union of the basic CGLP variables, i.e., $K := \cup_{t \in \mathcal{T}} K^t$.

Balas and Kis [30] characterize two types of irregular bases. if $|K| > n$, then the basis is irregular of *type 1*, while if $\operatorname{rank}(A_K) < |K|$, the basis is called irregular of *type 2*. For intuitive reasons, we refer to a type 1 basis as *type* $>$, and a type 2 basis as *type* $<$. A facet-defining cut $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ is irregular if every possible basis of (CGLP($\bar{\beta}$)) with $\alpha = \bar{\alpha}$ is irregular.

Theorem 5.9. *Suppose that for each $t \in \mathcal{T}$, the constraints $A^t x \geq b^t$ contain $Ax \geq b$ as the first m inequalities. Let $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ be a valid facet-defining cut for P_D. Suppose the associated sets F^{t*}, for $t \in \mathcal{T}^*$, are all simple. Let $N := \cup_{t \in \mathcal{T}^*} N^{t*} \cap [m]$. If $|N| > n$, then the cut is irregular and every corresponding basis of* (CGLP($\bar{\beta}$)) *is of type $>$. If* $\operatorname{rank}(A_N) < |N|$*, then the cut is irregular and every corresponding basis of* (CGLP($\bar{\beta}$)) *is of type $<$.*

Proof. Corollary 5.8 implies that, regardless of which basis is chosen for (CGLP($\bar{\beta}$)) that corresponds to this same cut, the set of positive multipliers J^{t+} will equal N^{t*}, so $K \supseteq N$. Thus, when $|N| > n$, this gives rise to a set K with $|K| > n$, which implies the cut is irregular of type $>$. When $\operatorname{rank}(A_N) < |N|$, the conditions of a type $<$ basis are satisfied: any CGLP basis will have a set of positive multipliers containing two linearly dependent rows. □

5.3 From VPCs to L&PCs

In the previous section, we laid the groundwork for using a feasible solution $(\bar{\alpha}, \bar{\beta})$ to (PRLP) to get a basic feasible solution to (CGLP($\bar{\beta}$)). We proved that in the case that a given valid cut exposes only simple faces on each disjunctive term, then the intersection of all the cobases of the points and rays tight on that face provides the indices of the positive CGLP variables. Fortunately, the setting of Chapter 4 is exactly one in which we use a simple (in fact, also simplicial) polyhedral relaxation of each P^t, and hence every face of that relaxation is simple. The general setting is more challenging due to the potential presence of degeneracy. (See Example 2 in Section 5.4.)

Our goal is to define the basic variables $\{J^t\}_{t \in \mathcal{T}}$. To set the stage, what we need are as many basic variables as there are rows of the system (CGLP($\bar{\beta}$)) excluding the nonnegativity constraints, which is $|T|(n+1)$ basic variables. Without loss of generality, all of the α variables will be basic (as they are free variables). We will show later in this section, in Lemma 5.10, how to handle terms $t \in \mathcal{T} \setminus \mathcal{T}^*$, by constructing J^t such that $|J^t| = n+1$. This implies that the remaining basic variables must satisfy $\sum_{t \in \mathcal{T}^*} |J^t| = |\mathcal{T}^*|(n+1) - n$.

For each $t \in \mathcal{T}^*$, let $p^t \in P^t$ such that $\bar{\alpha}^\mathsf{T} p^t = \bar{\beta}$. There exists a cobasis for p^t, $N^t \subseteq N^t(p^t)$ with $|N^t| = n$, such that $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ is valid for

$$C^t := \{x \in \mathbb{R}^n : A^t_{N^t} x \geq b^t_{N^t}\}.$$

We will select J^t as a subset of N^t. Note that this automatically means linear independence of the vectors we choose is satisfied (in accordance with Lemma 5.2). This basis cone C^t has one extreme point, p^t, and the extreme rays $\{r^i\}_{i \in N^t}$ are given by the columns of $(A^t_{N^t})^{-1}$.[9] Suppose for notational convenience that $(\mathcal{P}^t, \mathcal{R}^t)$ refers to the point-ray description of C^t (rather than of P^t). Thus, we define the sets $(\mathcal{P}^{t*}, \mathcal{R}^{t*})$, F^{t*}, and N^{t*} with respect to C^t.

The fact that C^t is both simple and simplicial has two implications. First, the collection of tight points and rays $(\mathcal{P}^t, \mathcal{R}^t)$ is affinely independent. Second, the indices J^{t+} are uniquely determined and given to us simply as N^{t*}. By Lemma 5.4,

$$|\mathcal{P}^{t*}| + |\mathcal{R}^{t*}| - 1 = \dim(F^{t*}) = n - |N^{t*}| = n - |J^{t+}|.$$

We also know that $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ defines a facet of P_D, certified by $\cup_{t \in \mathcal{T}^*}(\mathcal{P}^{t*}, \mathcal{R}^{t*})$. Thus,

$$n = \sum_{t \in \mathcal{T}^*}(|\mathcal{P}^{t*}| + |\mathcal{R}^{t*}|) = \sum_{t \in \mathcal{T}^*}(\dim(F^{t*}) - 1) = \sum_{t \in \mathcal{T}^*}(n - |J^{t+}| - 1) = |\mathcal{T}^*|(n+1) - \sum_{t \in \mathcal{T}^*}|J^{t+}|.$$

Thus, setting $J^t = N^{t*}$ for all $t \in \mathcal{T}^*$, we have the required number of basic variables for (CGLP($\bar{\beta}$)). Now we discuss how to actually find their values.

Having identified the indices J^{t+}, we now set the values $\bar{v}^t_i$, $i \in J^{t+}$, such that $\bar{\alpha}^\mathsf{T} = \sum_{i \in J^{t+}} \bar{v}^t_i A^t_{i\cdot}$. Each extreme ray of C^t corresponds to increasing the slack on one of the n constraints defining C^t. For any $j \in J^{t+}$, $A^t_{j\cdot} r^j = A^t_{j\cdot}(A^t_{N^t})^{-1}_{\cdot j} = 1$ and $A^t_{i\cdot} r^i = 0$ for all $i \in N^t \setminus \{j\}$, and the dot product $\bar{\alpha}^\mathsf{T} r^j$ is known (and strictly positive). It follows that,

$$\bar{\alpha}^\mathsf{T} r^j = \sum_{i \in J^{t+}} \bar{v}^t_i A^t_{i\cdot} r^j = \bar{v}^t_j A^t_{j\cdot} r^j \qquad \Longrightarrow \qquad \bar{v}^t_j = \bar{\alpha}^\mathsf{T} r^j.$$

In the case that there exists a point of P^t, $t \in \mathcal{T}^*$, tight for the cut and having no primal degeneracy, we would pick that point as p^t, and the rest of the construction for that term would proceed as above. When there is degeneracy at p^t, we have to take care to pick a subset $N^t \subset N^t(p^t)$ so that $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ is valid for the resulting basis cone.

Lastly, we address setting the values of v^t for a disjunctive term $t \in \mathcal{T} \setminus \mathcal{T}^*$, i.e., one for which the cut is not supporting. First, observe that there exists $\bar{\beta}_t$ such that $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}_t$ is supporting for P^t, and we can find $\bar{\beta}_t$ using $\mathcal{P}^t$. The first statement is due to the linear program $\min_x\{\bar{\alpha}^\mathsf{T} x : x \in P^t\}$ being bounded (it has value at least $\bar{\beta}$ because the given cut is valid for P^t). The second statement is due to the fact that a linear program achieves its optimum at an extreme point, and we have these extreme points in $\mathcal{P}^t$, so

$$\bar{\beta}_t := \min_{x \in P^t} \bar{\alpha}^\mathsf{T} x = \min_{p \in \mathcal{P}^t} \bar{\alpha}^\mathsf{T} p.$$

That is, we only have to check the dot product of $\bar{\alpha}$ with each point in the point-ray collection from term t. Analogously to the $\mathcal{T}^*$ case, let $p^t \in \arg\min_p\{\bar{\alpha}^\mathsf{T} p : p \in \mathcal{P}^t\}$, and let $N^t \subseteq N^t(p^t)$ denote a cobasis of p^t such that $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}_t$ is valid for $C^t := \{x \in \mathbb{R}^n : A^t_{N^t} x = b^t_{N^t}\}$.

Let J^t be the set containing N^t and the index of the inequality $\mathbf{0}^\mathsf{T} x \geq -1$ assumed to be in the system (say this corresponds to index 0). Observe that $\{(A^t_{i\cdot}, b^t_i)\}_{i \in N^t} \cup (\mathbf{0}^\mathsf{T}, -1)$ are

[9] Add slack variables $s^t_{N^t}$ for each row indexed by N^t, so that $A^t_{N^t} x - s^t_{N^t} = b^t_{N^t}$. Then observe that $x = (A^t_{N^t})^{-1} b^t_{N^t} + (A^t_{N^t})^{-1} s^t_{N^t} = p^t + \sum_{i \in N^t} r^i s^t_i$. Keeping the rays unscaled is important for our derivation.

linearly independent, as $A^t_{N^t}$ has rank n because it defines a vertex of P^t, so $\mathbf{0}^\intercal$ cannot be written as a linear combination of the rows of $A^t_{N^t}$, meaning $(\mathbf{0}^\intercal, -1)$ is linearly independent of the vectors $(A^t_{N^t}, b^t_{N^t})$.

We set the values of v^t by the same recipe as before for all $i \in N^t$, i.e., by setting $\bar{v}^t_i = \bar{\alpha}^\intercal r^i$, where r^i denotes the column of $(A^t_{N^t})^{-1}$ associated with $i \in N^t$. This means that

$$\sum_{i \in N^t} \bar{v}^t_i b^t_i = \bar{\alpha}^\intercal \sum_{i \in N^t} (A^t_{N^t})^{-1}_{\cdot i} b^t_i = \bar{\alpha}^\intercal (A^t_{N^t})^{-1} b^t = \bar{\alpha}^\intercal p^t = \bar{\beta}_t.$$

For $i = 0$, we simply set $\bar{v}^t_0 = \bar{\beta}_t - \bar{\beta}$. This implies that $\bar{v}^t b^t = \bar{\beta}_t - (\bar{\beta}_t - \bar{\beta}) = \bar{\beta}$, as desired. We also can check that the coefficient on $\bar{\alpha}_j$, $j \in [n]$, is correctly set:

$$\bar{v}^t A^t_{\cdot j} = \sum_{i \in N^t} \bar{v}^t_i A^t_{ij} = \sum_{i \in N^t} \bar{\alpha}^\intercal (A^t_{N^t})^{-1}_{\cdot i} A^t_{ij} = \bar{\alpha}^\intercal (A^t_{N^t})^{-1} A^t_{\cdot j} = \bar{\alpha}^\intercal e_j = \bar{\alpha}_j.$$

We have proved the following lemma, which completes the desired correspondence.

Lemma 5.10. *If $t \in \mathcal{T} \setminus \mathcal{T}^*$, then $|J^t| = n + 1$ without loss of generality.* □

A final computational remark: in the context of Chapter 4, we have already performed the expensive inverse computation to get the rays of each C^t (at the time the point-ray collection is determined).

5.4 Correspondence examples

In this section, we give three examples of the correspondence between the $\mathcal{V}$-polyhedral and $\mathcal{H}$-polyhedral cut-generating approaches (i.e., between feasible solutions of (PRLP) and (5.1)). The first example shows the construction of an irregular basis (but not an irregular cut), the second example illustrates the effect of degeneracy, and the third example shows the effect of a disjunctive term for which the cut is not supporting.

Example 1.

Consider the following problem:

$$\begin{aligned} \min_{x_1, x_2} \quad & -x_2 \\ & 2x_1 - x_2 \geq 1/2 && \text{(c1.1)} \\ & -(1/2)x_1 - x_2 \geq -3/4 && \text{(c1.2)} \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

The optimal solution to the linear programming relaxation is $\bar{x} = (1/2, 1/2)$. A valid disjunction is $(-x_1 \geq 0;\ -x_2 \geq 0) \vee (x_1 \geq 1;\ -x_2 \geq 0)$. The disjunctive hull introduces two new inequalities (cuts):

$$\begin{aligned} (1/2)x_1 - x_2 &\geq 1/2 && \text{(c1.3)} \\ -x_2 &\geq 0. && \text{(c1.4)} \end{aligned}$$

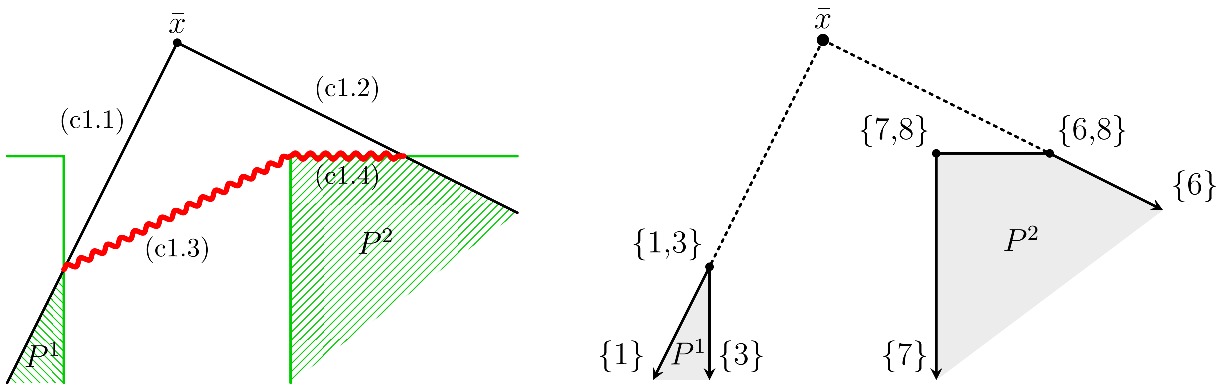


Figure 5.1: Left panel: feasible region of P and the disjunction for the example. Right panel: the inequalities tight for each extreme point, numbered as in the CGLP.

The disjunction, incorporating the inequalities of P, is

$$\begin{pmatrix} 2x_1 - x_2 \geq 1/2 \\ -(1/2)x_1 - x_2 \geq -3/4 \\ -x_1 \geq 0 \\ -x_2 \geq 0 \end{pmatrix} \vee \begin{pmatrix} 2x_1 - x_2 \geq 1/2 \\ -(1/2)x_1 - x_2 \geq -3/4 \\ x_1 \geq 1 \\ -x_2 \geq 0 \end{pmatrix}.$$

To distinguish between the inequalities of the two disjunctive terms, we will refer to the inequalities defining P^1 by numbers 1 through 4 and to the inequalities defining P^2 by numbers 5 through 8, which will also correspond to the numbering for the columns (variables) of the CGLP for this problem. The LP feasible region and the boundary of the disjunctive terms are shown in the left panel of Figure 5.1. The right panel indicates the cobasis of each extreme point and ray of P^1 and P^2, where the numbers in the braces indicate the inequalities tight at that point or ray.

The following is the feasible region of the CGLP for this example.

$$\begin{array}{ccccccccccccc}
v_1^1 & v_2^1 & v_3^1 & v_4^1 & v_5^2 & v_6^2 & v_7^2 & v_8^2 & \alpha_1 & \alpha_2 & \beta & & \\
2 & -1/2 & -1 & 0 & & & & & -1 & & & = & 0 \\
-1 & -1 & 0 & -1 & & & & & & -1 & & = & 0 \\
1/2 & -3/4 & 0 & 0 & & & & & & & -1 & \geq & 0 \\
 & & & & 2 & -1/2 & 1 & 0 & -1 & & & = & 0 \\
 & & & & -1 & -1 & 0 & -1 & & -1 & & = & 0 \\
 & & & & 1/2 & -3/4 & 1 & 0 & & & -1 & \geq & 0 \\
 & & & & & & & & & & 1 & = & 1/2
\end{array}$$

The PRLP, with β fixed to 1/2, is as follows, constructed from the extreme points and rays of P^1 and P^2.[10] We also indicate the cobasis of each point and ray, as well as the slack with respect to $\bar{\alpha}^\mathsf{T} = (1/2, -1)$, corresponding to cut (c1.3). We use this solution to

[10]Note that P^1 and P^2 share an extreme ray $(0, -1)$, and this extreme ray is redundant, in the sense that it can never be tight for any valid cut for P_D. This demonstrates the importance of preprocessing the PRLP in practice, which we discussed in Chapter 4.

construct a basic feasible solution to (CGLP($\bar{\beta}$)).

α_1	α_2		β	cobasis	slack	name
0	$-1/2$	$\geq$	$1/2$	$\{1,3\}$	0	p^1
0	-1	$\geq$	0	$\{3\}$	1	r^1
-1	-2	$\geq$	0	$\{1\}$	$3/2$	r^3
1	0	$\geq$	$1/2$	$\{7,8\}$	0	p^2
$3/2$	0	$\geq$	$1/2$	$\{6,8\}$	$1/4$	p^3
1	$-1/2$	$\geq$	0	$\{6\}$	1	r^5
0	-1	$\geq$	0	$\{8\}$	1	r^8

Observe that the tight points and rays for this cut are $(0,-1/2)$ and $(1,0)$; call them p^1 and p^2. We have that $N^{1*} = N^1(p^1) = \{1,3\}$ and $N^{2*} = N^2(p^2) = \{7,8\}$. Following the procedure of the previous section, we define the cones C^1 and C^2. The apex of C^1 is p^1 and its rays are r^1 and r^3, while the apex of C^2 is p^2 and its rays are r^7 and r^8. These are all defined in the point-ray collection except for r^7, which can be calculated to equal $(1,0)$.[11] We set values $\bar{v}^1_1 = \bar{\alpha}^\intercal r^1 = 1$, $\bar{v}^1_3 = \bar{\alpha}^\intercal r^3 = 3/2$, $\bar{v}^2_7 = \bar{\alpha}^\intercal r^7 = 1/2$, and $\bar{v}^2_8 = \bar{\alpha}^\intercal r^8 = 1$. Observe that

$$\bar{v}^1(A^1,b^1) = \bar{v}^1_1(2,-1,1/2) + \bar{v}^1_3(-1,0,0) = (1/2,-1,1/2) = (\bar{\alpha}^\intercal,\bar{\beta})$$
$$\bar{v}^2(A^2,b^2) = \bar{v}^2_7(1,0,1) + \bar{v}^2_8(0,-1,0) = (1/2,-1,1/2) = (\bar{\alpha}^\intercal,\bar{\beta}).$$

Hence, we have identified the desired Farkas multipliers, as desired.

In this case, the set of inequalities of P associated with positive CGLP variables is $K = \{1\}$. However, neither the basis nor the cut is irregular. Although the number of basic multipliers on the constraints of P is one (fewer than $n = 2$), the associated submatrix of A has full row rank, so we can complete a cobasis $N \supseteq K$ that gives rise to the cut, namely $N = \{1,2\}$. The fact that the cut is not irregular can also be seen by using the CGLP variables $\{v^1_3, v^1_4\}$ and $\{v^2_7, v^2_8\}$, which yield the set convex S for this example, i.e., that the cut is an intersection cut from the disjunction $(-(3/2)x_1 \geq 0) \vee ((1/2)x_1 - x_2 \geq 1/2)$.

If we were to use the technique of Chapter 4, then prior to generating a cut, we would choose a basis cone for each disjunctive term and get the corresponding simple point-ray collection. One of the possibilities for $\arg\min_x\{-x_2 : x \in P^2\}$ is the point $p^3 = (3/2,0)$ (instead of $(1,0)$). If that point were chosen, the cut (c3.3) would not be feasible for $C(N^2(p^3))$. Moreover, the points and rays $(\mathcal{P}^1,\mathcal{R}^1)$ would all be immediately redundant. The reason is that one of the rays at $C(N^2(p^3))$ is $(-1,0)$, which intersects the boundary of the first disjunctive term. In Section 4.5.2, we discussed a potential remedy were such a situation to arise. The idea is to "activate" disjunctive inequalities to prevent rays originating in one term of the disjunction from intersecting other terms.

Example 2.

We illustrate the complication caused by degeneracy by the instance shown in Figure 5.2. The figure shows the feasible region to a polyhedron P (truncated by the unit cube), whose

[11]Note that the rays in the point-ray collection are unscaled, corresponding directly to $(A^t_{N^{t*}})^{-1}$, as otherwise we have to normalize when computing the values of the CGLP multipliers.

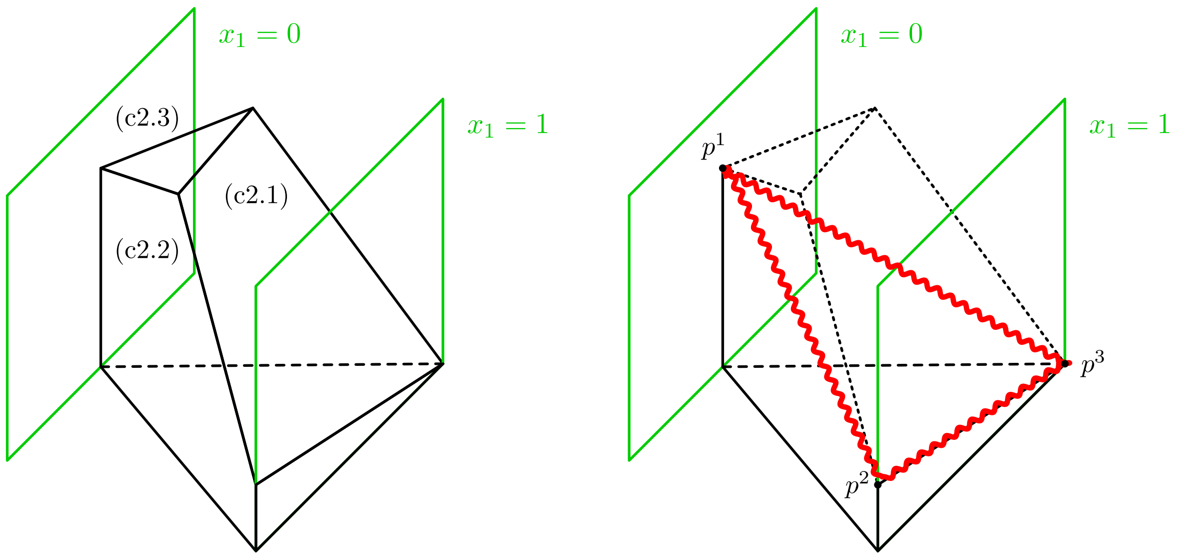


Figure 5.2: Example 2 illustrating disjunctive terms with primal degeneracy. The initial polyhedron has no degeneracy, but it is introduced after taking the disjunction.

constraints are below, and a cut derived from the disjunction $(-x_1 \geq 0) \vee (x_1 \geq 1)$.

$$-(13/8)x_1 - (1/4)x_2 - x_3 \geq -15/8 \tag{c2.1}$$
$$(1/2)x_1 + x_2 \geq 1/2 \tag{c2.2}$$
$$(1/2)x_1 - x_3 \geq -3/4 \tag{c2.3}$$
$$(1/2)x_1 - x_2 \geq -1/2 \tag{c2.4}$$
$$x_2 \geq 0 \tag{c2.5}$$

The disjunctive cut we generate is

$$-(5/8)x_1 - (1/4)x_2 - x_3 \geq -7/8,$$

from which we set $\bar{\alpha}^{\mathsf{T}} = (-5/8, -1/4, -1)$ and $\bar{\beta} = -7/8$. The cut, depicted in the right panel of the figure, is incident to a point on P^1, $p^1 = (0, 1/2, 3/4)$, that is tight for four inequalities: three defining P (constraints 2–4), and one being $x_1 = 0$. These four inequalities comprise N^{1*}: $N^{1*} = \{2, 3, 4, 0_1\}$ (where 0_1 refers to the disjunction-defining inequality $-x_1 \geq 0$). Note that the original polyhedron is simple, but P^1 is not.

Applying the correspondence procedure detailed in Section 5.3, we require the choice of a set $N^1 \subseteq N^1(p^1)$ such that $\bar{\alpha}^{\mathsf{T}} x \geq \bar{\beta}$ is valid for $C(N^1)$. We cannot choose an arbitrary subset of three indices from $N^1(p^1)$. Of course, one of the inequalities in N^1 must be $-x_1 \geq 0$ (otherwise, we have not impose the disjunction, and the cut is not valid for $C(\{2, 3, 4\})$). For the remaining inequalities, we need to pick two of $\{2, 3, 4\}$. If we pick $\{2, 3\}$, we again get an invalid basis cone: one of the rays (corresponding to the slack on inequality (c2.2)) is $(0, 1, 0)$, and $\bar{\alpha}^{\mathsf{T}}(0, 1, 0) = -1/4 \ngeq 0$). It can be verified that the only valid choice for this example is $N^1 = \{3, 4, 0_1\}$,[12] which results in the cone $C(N^1)$ having three rays: $r^3 = (0, 0, -1)$,

[12]We are essentially finding an optimal basis of $\min_x\{\bar{\alpha}^{\mathsf{T}} x : x \in P^1\}$ from knowing the optimal solution.

$r^4 = (0, -1, 0)$, and $r^{0_1} = (-1, -1/2, -1/2)$. Calculating the dot product of the cut with each of these rays, we find $\bar{v}^1_3 = 1$, $\bar{v}^1_4 = 1/4$, and $\bar{v}^1_{0_1} = 5/4$, satisfying $\bar{v}^1(A^1, b^1) = (\bar{\alpha}^\intercal, \bar{\beta})$.

On the other side of the split, the cut is incident to two points, p^2 and p^3 as labeled in the right panel of Figure 5.2, resulting in $N^{2*} = N^2(p^2) \cap N^2(p^3) = \{1, 0_2\}$ (0_2 corresponding to $x_1 \geq 1$). The point p^2 is nondegenerate, having cobasis $N^2 := N^2(p^2) = \{1, 2, 0_2\}$. Using this point to define the basis cone for term 2, we get that the rays of $C(N^2)$ (by taking the inverse of $A^2_{N^2}$) are $r^1 = (0, 0, -1)$, $r^2 = (0, 1, -1/4)$, and $r^{0_2} = (1, -1/2, -3/2)$. From this, we calculate that $\bar{v}^2_1 = \bar{\alpha}^\intercal r^1 = 1$ and $\bar{v}^2_{0_2} = \bar{\alpha}^\intercal r^{0_2} = 1$ (it is unnecessary to check $\bar{\alpha}^\intercal r^2$, as $2 \notin N^{2*}$, which means the dot product will necessarily equal 0). It is easy to see that this solution satisfies $\bar{v}^2(A^2, b^2) = (\bar{\alpha}^\intercal, \bar{\beta})$.

Example 3.

In this third example, we demonstrate the case of a cut that is not supporting for one of the disjunctive terms, using a variant of the irregular cut illustrated in Figure 1.6. Consider the following integer program:

$$\begin{aligned} \min_{x_1,x_2} \quad & -x_1 - x_2 & \\ & (5/2)x_1 - x_2 \geq -5/4 & \text{(c3.1)} \\ & -x_1 + (5/2)x_2 \geq -5/4 & \text{(c3.2)} \\ & -x_1 - x_2 \geq -7/4 & \text{(c3.3)} \\ & x_1, x_2 \in \mathbb{Z} & \end{aligned}$$

Let $Ax \geq b$ denote the inequalities (c3.3)–(c3.1). We will use the three-term disjunction

$$\begin{pmatrix} Ax \geq b \\ -x_1 \geq 0 \\ -x_2 \geq 0 \end{pmatrix} \vee \begin{pmatrix} Ax \geq b \\ x_1 \geq 1 \\ -x_2 \geq 0 \end{pmatrix} \vee \begin{pmatrix} Ax \geq b \\ -x_1 \geq 0 \\ x_2 \geq 1 \end{pmatrix}.$$

The disjunctive hull introduces one cut, which we define as $\bar{\alpha}^\intercal x \geq \bar{\beta}$:

$$-x_1 - x_2 \geq -5/4. \tag{c3.4}$$

Figure 5.3 shows the LP feasible region in the left panel and the result of taking the disjunction (including the addition of the disjunctive cut) in the right panel.

The feasible region of (PRLP($\bar{\beta}$)), with $\bar{\beta} = -5/4$, is given below, as well as the slack of each point with respect to the cut (c3.4). The cobasis of each point is also indicated. As in Example 1, to better distinguish among the disjunctive terms, we number the inequalities sequentially: inequalities 1–5 for term 1, inequalities 6–10 for term 2. and inequalities 11–15

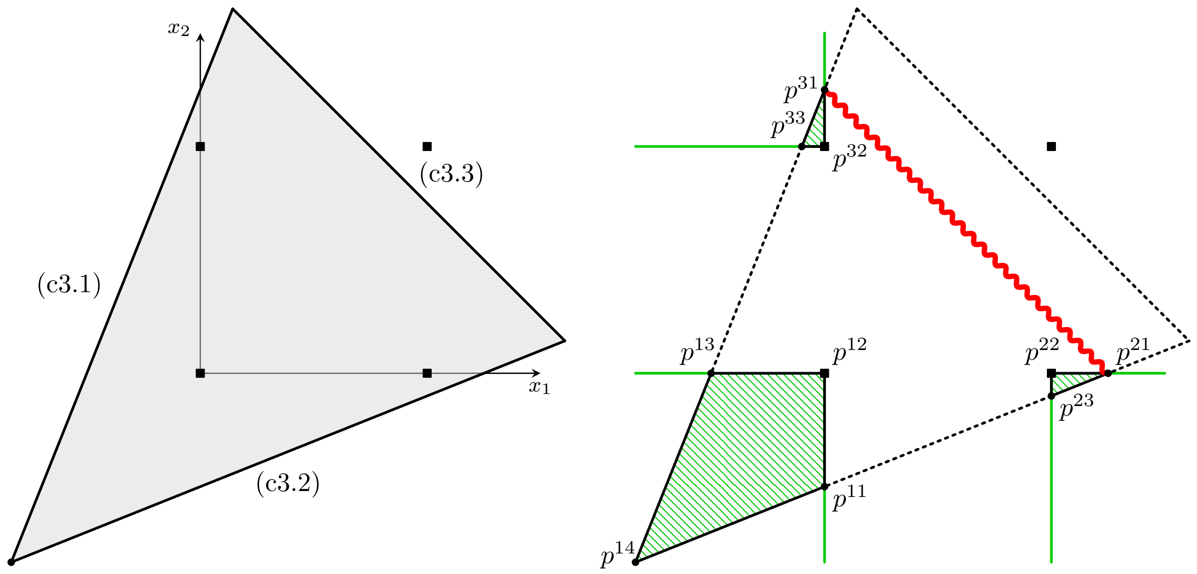


Figure 5.3: Example of an irregular cut from a three term non-simple disjunction. The term defined by $(x_1 \le 0, x_2 \le 0)$ ultimately causes irregularity of the type in which $|N| > n$.

for term 3, where the first three indices for each term refer to the original inequalities $Ax \ge b$.

α_1	α_2		β	cobasis	slack	name
0	$-1/2$	$\ge$	$-5/4$	$\{2,4\}$	$7/4$	p^{11}
0	0	$\ge$	$-5/4$	$\{4,5\}$	$5/4$	p^{12}
$-1/2$	0	$\ge$	$-5/4$	$\{1,5\}$	$7/4$	p^{13}
$-5/6$	$-5/6$	$\ge$	$-5/4$	$\{1,2\}$	$35/12$	p^{14}
$5/4$	0	$\ge$	$-5/4$	$\{7,10\}$	0	p^{21}
1	0	$\ge$	$-5/4$	$\{9,10\}$	$1/4$	p^{22}
1	$-1/10$	$\ge$	$-5/4$	$\{7,9\}$	$7/20$	p^{23}
0	$5/4$	$\ge$	$-5/4$	$\{11,14\}$	0	p^{31}
0	1	$\ge$	$-5/4$	$\{14,15\}$	$1/4$	p^{32}
$-1/10$	1	$\ge$	$-5/4$	$\{11,15\}$	$7/20$	p^{33}

Observe that none of the points from $P^1 = \operatorname{conv}(\{p^{11}, p^{12}, p^{13}, p^{14}\})$ are tight for the cut, so the cut is not supporting for that term. We will return to this term after getting the values of v^2 and v^3.

The only point tight (having zero slack) on P^2 for the cut is p^{21}. Its cobasis is $N^2(p^{21}) = \{7, 10\}$, where inequality 7 refers to (c3.2), and inequality 10 to $-x_2 \ge 0$. As this is the only tight point, and its cobasis is nondegenerate, we have that $N^{2*} = J^2$. To calculate the values of v^2, we need the rays of $C(N^2(p^{21}))$. Taking the inverse of $A^2_{N^2(p^{22})}$, we get that $r^7 = (-5/2, -1)$ and $r^{10} = (-1, 0)$. Thus, we calculate that $\bar{v}^2_7 = \bar{\alpha}^\intercal r^7 = 7/2$ and $\bar{v}^2_{10} = \bar{\alpha}^\intercal r^{10} = 1$.

For term 3, the only tight point is p^{31}, which leads to $J^3 = \{11, 14\}$, where inequality 11 corresponds to (c3.1) and inequality 14 to $-x_1 \ge 0$. The rays of $C(N^3(p^{31}))$ are $r^{11} = (0, -1)$ and $r^{14} = (-1, -5/2)$, resulting in values $\bar{v}^3_{11} = \bar{\alpha}^\intercal r^{11} = 1$ and $\bar{v}^3_{14} = \bar{\alpha}^\intercal r^{14} = 7/2$. It is easy to verify that $\bar{v}^t(A^t, b^t) = (\bar{\alpha}^\mathsf{T}, \bar{\beta})$ for $t \in \{2, 3\}$.

We now handle term 1, by first calculating the optimal point $\min_p\{\bar{\alpha}^\mathsf{T} p : p \in \mathcal{P}^1\}$—it is simply the point of $\mathcal{P}^1$ with minimum slack, which happens to be uniquely the point $p^{12} = (0, 0)$, with cobasis $N^1(p^{12}) = \{4, 5\}$ (which equals N^{1*} because the cobasis is nondegenerate). The rays of $C(N^1(p^{12}))$ are $r^4 = (-1, 0)$ and $r^5 = (0, -1)$, which yields values $\bar{v}^1_4 = \bar{\alpha}^\mathsf{T} r^4 = 1$ and $\bar{v}^1_5 = \bar{\alpha}^\mathsf{T} r^5 = 1$. As stated in the discussion preceding Lemma 5.10, assuming $\mathbf{0}^\mathsf{T} x \geq -1$ is also part of the system defining term 1, say with inequality index 0, we set $\bar{v}^1_0 = 5/4$ to compensate for the right-hand side disparity between $\bar{\beta} = -5/4$ and $\bar{\alpha}^\mathsf{T} p^{12} = 0$. Note that in this example, the inequalities defining P^t subsume the $Ax \geq b$ constraints. This means, by Proposition 5.11, that there exists a feasible solution v^1 without using positive weight on the redundant slack constraint. Moreover, the proof of that result is constructive. We leave it as an exercise that applying this construction produces another feasible solution for term 1, $\bar{v}^1_1 = (0, 0, 5/7, 2/7, 2/7)$. This happens to have only three positive variables, which we can choose for our basis.

The above basis is irregular. Define the set K as the indices of $Ax \geq b$ that have positive multipliers in $\{\bar{v}^t\}_{t \in \mathcal{T}}$. We find that $K = \{1, 2, 0\}$ (in the case that $\mathbf{0}^\mathsf{T} x \geq -1$ is used with positive weight) or $K = \{1, 2, 3\}$; thus the basic feasible solution we constructed is irregular. However, there actually exists a regular basis for this example. This is because, due to the particular structure of this instance, any three of the five variables $v^1_1, \ldots, v^1_5$ can be made basic, e.g., we can use $v^1 = (1, 0, 0, 7/2, 0)$, with v^1_1, v^1_4, and v^1_5 basic. The corresponding point of P is p^{14}, which is is feasible to the disjunction. Though we cannot take a standard intersection cut from this cobasis, we can generate a simple disjunctive cut via Lemma 1.2, which results precisely in the cut (c3.4). We discuss a variant of this example in more detail in Appendix A.

5.5 Enforcing equality in the β constraints

We prove that enforcing equality in the constraints $v^t b^t = \beta$ (as opposed to $v^t b^t \geq \beta$) holds without loss of generality, meaning that we do not lose any valid cuts obtainable from P_D.

Proposition 5.11. *Let $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ denote a feasible inequality for P_D that cuts some point of $P := \{x \in \mathbb{R}^n : Ax \geq b\}$, and $\bar{v}^t$, $t \in \mathcal{T}$, a set of Farkas multipliers certifying the validity of the cut. Suppose that $A^t x \geq b^t$ includes all the constraints $Ax \geq b$ for every $t \in \mathcal{T}$ for which the cut is not supporting on P^t. Then there exists $\{\bar{v}^t\}_{t \in \mathcal{T}}$ such that $\bar{v}^t b^t = \bar{\beta}$ for all $t \in \mathcal{T}$.*

Proof. Let $t \in \mathcal{T}$. We have that $\bar{v}^t A^t = \bar{\alpha}^\mathsf{T}$ and $\bar{v}^t b^t \geq \bar{\beta}$. If the cut is supporting for P^t, it always holds that $\bar{v}^t b^t = \bar{\beta}$: given $p^t \in P^t$ such that $\bar{\alpha}^\mathsf{T} p^t = \bar{\beta}$, it follows that

$$\bar{\beta} = \bar{\alpha}^\mathsf{T} p^t = \bar{v}^t A^t p^t \geq \bar{v}^t b^t \geq \bar{\beta},$$

so all the inequalities, including the last, hold at equality.

Now suppose $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ is not supporting for P^t. Since $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ for all $x \in P_D$, we have that: (1) there exists $\widetilde{\beta}_0 \in \mathbb{R}$ such that $\bar{\alpha}^\mathsf{T} x \geq \widetilde{\beta}_0$ is valid and supporting for P, and (2) there exists $\widetilde{\beta}_t \in \mathbb{R}$ (with $\widetilde{\beta}_t > \widetilde{\beta}_0$ and $\widetilde{\beta}_t > \bar{\beta}$) such that $\bar{\alpha}^\mathsf{T} x \geq \widetilde{\beta}_t$ is valid and supporting for P^t. (These can be obtained by minimizing $\bar{\alpha}^\mathsf{T} x$ over $x \in P$ and $x \in P^t$.)

Therefore, there exist nonnegative vectors $\widetilde{v}^0$ and $\widetilde{v}^t$ such that $\widetilde{v}^0 A^t = \bar{\alpha}$, $\widetilde{v}^0 b^t = \widetilde{\beta}_0$ (where $\widetilde{v}^t_i = 0$ for all indices not corresponding to the constraints of $Ax \geq b$), and $\widetilde{v}^t A^t = \bar{\alpha}$, $\widetilde{v}^t b^t = \widetilde{\beta}_t$. Hence, letting $\theta_t = (\widetilde{\beta}_t - \bar{\beta})/(\widetilde{\beta}_t - \widetilde{\beta}_0)$, which is nonnegative by the assumption that $\bar{\beta} > \min_x\{\bar{\alpha}^\mathsf{T} x : x \in P^t\} = \widetilde{\beta}_0$, we can set $\bar{v}^t = \theta_t \widetilde{v}^0 + (1 - \theta_t)\widetilde{v}^t$, from which it follows that $\bar{v}^t A^t = \bar{\alpha}$ and $\bar{v}^t b^t = \bar{\beta}$. □

5.6 Conclusion

This chapter establishes the details of a correspondence between the bases of the linear programs used to generate cuts from the $\mathcal{V}$-polyhedral and lift-and-project methodologies. In our experiments of Chapter 4, there exist instances for which Gomory cuts are much stronger than $\mathcal{V}$-polyhedral cuts. We hypothesize that modularization of the cut coefficients plays a substantial explanatory role, and this chapter provides a necessary step towards testing that hypothesis. Namely, the correspondence we develop enables applying cut strengthening techniques to $\mathcal{V}$-polyhedral cuts, by deriving the Farkas multipliers that are required for the technique. This theoretical derivation requires substantial complementary computational experimentation, to ascertain whether the strengthening be done efficiently in practice, and whether (and under what conditions) the extra effort to strengthen the cuts is worthwhile in terms of producing sufficiently stronger cuts.

He taught her how to split and define
But if you study the logistics
And heuristics of the mystics
You will find that their minds rarely move in a line
—Brian Eno, *Backwater*, 1977

Chapter 6

Conclusions & Future Research

This dissertation focuses on the development and evaluation of non-recursive methods for generating cutting planes for mixed-integer linear programs. We investigate three distinct frameworks: partial hyperplane activation for generalized intersection cuts (Chapter 2), cuttings planes by tilting on split disjunctions (Chapter 3), and $\mathcal{V}$-polyhedral cuts (Chapter 4).

These trace an organic progression of increasingly stronger cuts, culminating with $\mathcal{V}$-polyhedral cuts, in which we provide a computationally efficient approach for separating cuts from general valid disjunctions (as opposed to the limited class of simple disjunctions studied in much of the literature). Instead of the common practice of using a multitude of weak disjunctions derived from the integrality of one or two variables, we obtain cuts from one strong disjunction arising from the set of leaf nodes of a partial branch-and-bound tree. The computational results with $\mathcal{V}$-polyhedral cuts demonstrate that we are able to effectively generate cuts from these strong disjunctions that are substantially stronger than the baselines of Gomory cuts and those cuts that are currently implemented in the commercial solver `Gurobi`. Moreover, for many instances, providing the cuts to `Gurobi`, for use during branch-and-bound, results in reduced solving times. Given these promising results, in Chapter 5, we build a Farkas certificate for our cuts, which is a necessary ingredient for applying classical cut strengthening techniques to $\mathcal{V}$-polyhedral cuts.

Altogether, this dissertation contributes methods, theory, and computational experience on the frontier of cutting plane research. The results indicate two primary directions for related future research. First, a practical implementation of the cuts we introduce would benefit from a refined understanding of the interactions between the presolve, cutting plane, and branching components of a solver. For example, in the context of $\mathcal{V}$-polyhedral cuts, we have left open the questions of which instances are amenable to the cuts, and if they are applied, then which particular partial branch-and-bound tree should be used for cut generation. This represents an area of active research (see, e.g., [118]). Second, our single-minded purpose has been the setting of mixed-integer linear programming. However, some of our techniques have natural extensions to nonlinear programming, an area with increasing prevalence and importance. More concretely, there has already been recent work testing intersection cuts for polynomial optimization [42]. Simultaneously, progress has been made on resolve capabilities and branch-and-bound methods for nonlinear settings [41], which may enable the the extension of $\mathcal{V}$-polyhedral cuts to convex optimization settings. It is therefore worthwhile to pursue generalizations of our methods to nonlinear disjunctive programming.

Appendix A

On Regular Bases and Simple Disjunctive Cuts

In this chapter, we make a distinction between irregular cuts defined as those that cannot be obtained as a *standard intersection cut* (SIC) and a more general class of irregularity: cuts that cannot be obtained as a *simple disjunctive cut*, as defined in Lemma 1.2. The key difference is that simple disjunctive cuts apply without the assumption that the point being separated does not belong to the disjunctive hull. We also prove that regularity of a cut is related only to the positive variables in a solution to the *cut-generating linear program* (CGLP) for the terms of the disjunction for which the cut is supporting. After we formally define a *regular* basis, we give a motivating example, which generalizes the one from Figure 1.8 in Chapter 1.

Definition A.1. *A* regular *basis of* (CGLP) *is defined as one in which the set $K := \{i \in [m] : v_i^t > 0$ for some $t \in \mathcal{T}\}$ has cardinality at most n and A_K has full row rank.*

Consider the example shown in Figure A.1. The first panel of the figure gives the feasible region to the linear programming relaxation of the following integer program:

$$\begin{aligned} \min_{x_1,x_2} \quad & -x_2 \\ & 2x_1 - x_2 \geq -1 \\ & -x_1 + 2x_2 \geq -1 \\ & -x_1 - x_2 \geq -7/4 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

In the second panel, we display the imposition of the disjunction

$$\begin{pmatrix} -x_1 \geq 0 \\ -x_2 \geq 0 \end{pmatrix} \vee \begin{pmatrix} x_1 \geq 1 \\ -x_2 \geq 0 \end{pmatrix} \vee \begin{pmatrix} -x_1 \geq 0 \\ x_2 \geq 1 \end{pmatrix}.$$

The result is the generation of a new disjunctive cut,

$$-x_1 - x_2 \geq 1.$$

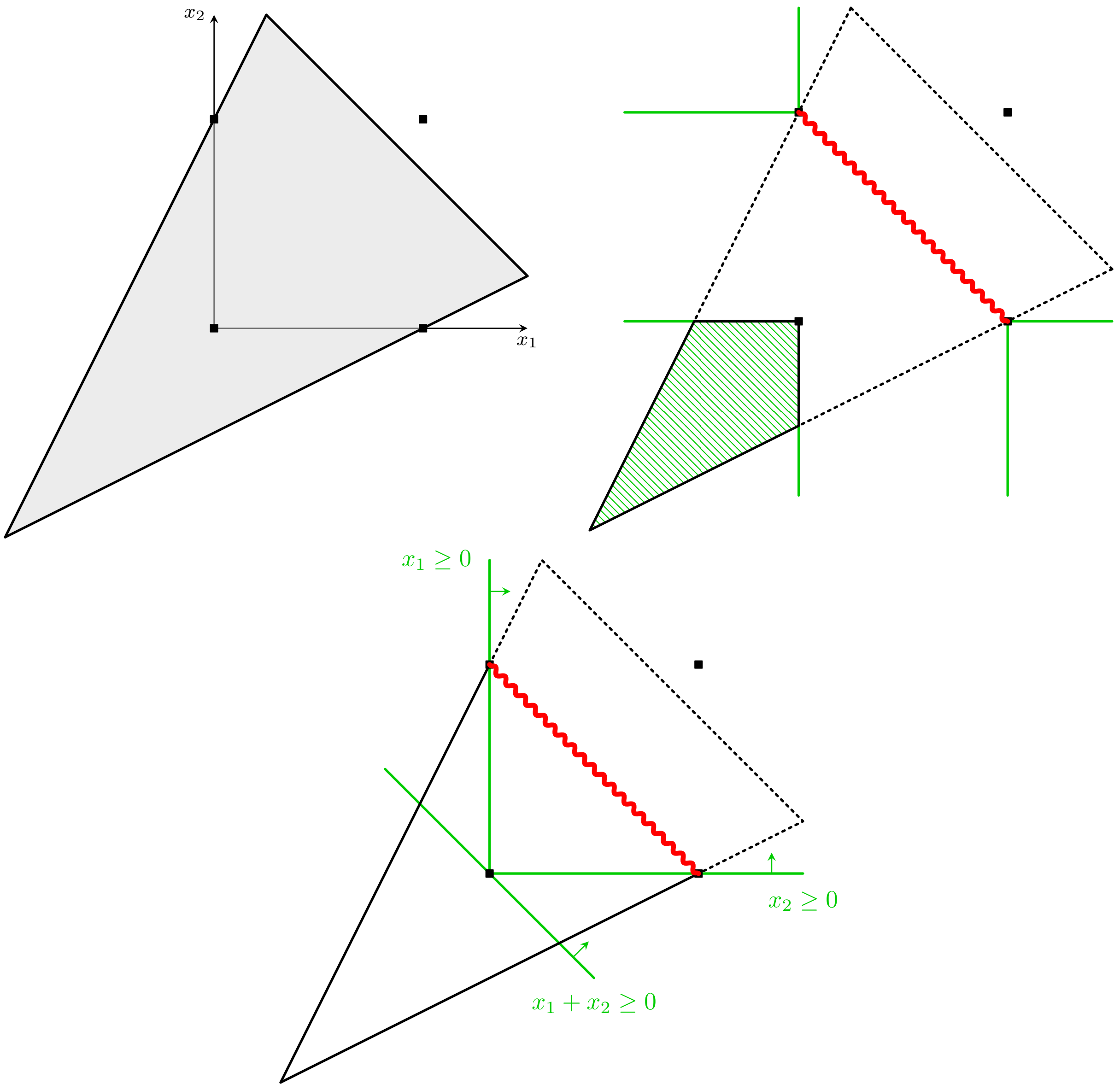


Figure A.1: Example of a cut that cannot be derived as a SIC, but can be generated as a simple disjunctive cut. The first panel shows the feasible region of the linear programming relaxation. The second panel shows the result of taking a three-term non-simple disjunction and the associated cut (with a wavy line). The last panel shows the boundary of the convex set S resulting from the CGLP variables on the disjunctive inequalities.

From the figure, it is clear that this cut cannot be obtained as a SIC. There are two bases corresponding to points of P that are not feasible to the disjunctive hull, and the SICs from these bases are dominated by the above cut.

This cut has associated regular and irregular bases, depending on the choice of basic variables for term 1. Suppose an irregular basis is returned, which results in the following multipliers for the corresponding CGLP:

$$\begin{aligned} u^1 &= (0,0,4/7) & u_0^1 &= (3/7,3/7)\\ u^2 &= (0,1,0) & u_0^2 &= (0,3)\\ u^3 &= (1,0,0) & u_0^3 &= (3,0).^{1} \end{aligned}$$

Multiplying the inequalities of the disjunction by the u_0^t multipliers gives rise to the simple disjunction

$$(-(3/7)x_1 - (3/7)x_2 \geq 0) \vee (-3x_2 \geq 0) \vee (-3x_1 \geq 0).$$

Reversing these inequalities and taking their conjunction yield the convex cut-generating set S shown in the bottom of Figure A.1.

The index set of the inequalities of P with positive CGLP variables is $N = \{1,2\}$. The associated rays of the cone $C(N)$ and optimal tableau are provided by the columns of A_N^{-1}:

$$A_N^{-1} = \begin{bmatrix} 2/3 & 1/3\\ 1/3 & 2/3 \end{bmatrix} \quad \Longrightarrow \quad \begin{aligned} x_1 &= -1 + (2/3)s_1 + (1/3)s_2\\ x_2 &= -1 + (1/3)s_1 + (2/3)s_2, \end{aligned}$$

where s_1 and s_2 are the nonnegative slacks associated with the first two constraints of P.

Using this optimal tableau, we can express the simple disjunction above in the space of the nonbasic variables:

$$(-(3/7)s_1 - (3/7)s_2 \geq -6/7) \vee (-s_1 - 2s_2 \geq -3) \vee (-2s_1 - s_2 \geq -3).$$

Normalizing each term by the right-hand side, we obtain

$$(-(1/2)s_1 - (1/2)s_2 \geq -1) \vee (-(1/3)s_1 - (2/3)s_2 \geq -1) \vee (-(2/3)s_1 - (1/3)s_2 \geq -1).$$

Finally, applying the formula for the simple disjunctive cut from Lemma 1.2, a valid cut for the disjunction is

$$\begin{aligned} &\max\{-1/2,-1/3,-2/3\}s_1 + \max\{-1/2,-1/3,-2/3\}s_2 \geq \min\{-1,-1,-1\}\\ &\Longrightarrow \quad -s_1 - s_2 \geq -3\\ &\Longrightarrow \quad -(2x_1 - x_2 + 1) - (-x_1 + 2x_2 + 1) \geq -3\\ &\Longrightarrow \quad -x_1 - x_2 \geq -1. \end{aligned}$$

We conjecture that the fact that the cut in this example can be derived as a simple disjunctive cut is no coincidence. Let $(\alpha, \beta, \{u^t, u_0^t\}_{t\in\mathcal{T}})$ denote a feasible solution to the CGLP such that $\alpha^\mathsf{T} x \geq \beta$ is not valid for P_D. Let S be defined as $\{x \in \mathbb{R}^n : u_0^t D^t x \leq u_0^t D_0^t \text{ for all } t \in \mathcal{T}\}$, and let $S' := \{x \in S : \alpha^\mathsf{T} x \leq \beta\}$. We first prove that, in a specific sense, the terms for which $\alpha^\mathsf{T} x \geq \beta$ is not supporting are redundant.

[1] A regular basis is, for example, obtained by setting $(u^1, u_0^1) = (u^2, u_0^2)$.

Proposition A.2. *Let $(\alpha, \beta, \{u^t, u_0^t\}_{t \in \mathcal{T}})$, S, and S' be defined as above. If $\alpha^\mathsf{T} x \geq \beta$ is not supporting for a term $t \in \mathcal{T}$, then the inequality $u_0^t D^t x \leq u_0^t D_0^t$ is redundant for S'.*

Proof. Note that the set S' is convex, and it is P_I-free (it is a restriction of S, which is P_I-free by definition). Moreover, $\alpha^\mathsf{T} x \geq \beta$ being valid for $P \setminus \operatorname{int} S$ means that the cut is also valid for $P \setminus \operatorname{int} S'$. Now suppose the cut is not supporting for some term $t \in \mathcal{T}$. The cut is valid for $\{x \in P : u_0^t D^t x \geq u_0^t D_0^t\}$, but it is not supporting for that set. It follows that $u_0^t D^t x \leq u_0^t D_0^t$ is, in turn, not supporting for the set S'. □

The utility of this is evident from the formula for the simple disjunctive cut. Namely, given a cobasis $\mathcal{N}$ and a disjunction $\vee_{t \in \mathcal{T}} (d^t s_{\mathcal{N}} \geq d_0^t)$ in the nonbasic space, the cut is

$$\sum_{j \in \mathcal{N}} \max_{t \in \mathcal{T}} d_j^t s_j \geq \min_{t \in \mathcal{T}} d_0^t,$$

so if we can replace the disjunction by one with fewer terms, we get a stronger cut.

We now state our conjecture for extending Theorems 9 and 12 by Balas and Kis [30] to cover simple disjunctive cuts, not merely SICs.

Conjecture A.3. *Let $(\alpha, \beta, \{u^t, u_0^t\}_{t \in \mathcal{T}})$, S, and S' be defined as above. Define $\mathcal{T}^* := \{t \in \mathcal{T} :$ there exists $p \in P^t$ for which $\alpha^\mathsf{T} p = \beta\}$. Let $K^* := \{i \in [m] : u_i^t > 0$ for some $t \in \mathcal{T}^*\}$. If $|K^*| \leq n$ and $\operatorname{rank}(A_{K^*}) = |K^*|$, then then the cut $\alpha^\mathsf{T} x \geq \beta$ is equivalent to the simple disjunctive cut associated with the sets S' and N.*

Suppose K^*, defined as in the conjecture, has cardinality at most n and $\operatorname{rank}(A_{K^*}) = |K^*|$. The idea is there exists a set $N \subseteq [m]$ that contains K^* and for which A_N is nonsingular. If the point $A_N^{-1} b_N \in \operatorname{int} S$, then the validity of the conjecture follows from [30, Theorem 9]. The result is also true when $A_N^{-1} b_N$ violates all the inequalities defining S: in this case, we can normalize each of these inequalities to have right-hand side -1 in the nonbasic space, as in the example, and the cut corresponds to intersecting each ray of $C(N)$ with the *farthest* facet-defining inequality for S.

Appendix B

Three Normalizations for Lift-and-Project Cuts

In this chapter, we discuss three normalizations for the CGLP and provide some geometric intuition for what they do to the feasible region of the *membership linear program* (MLP), the dual to (CGLP) (see Section 1.4). The motivation is a better understanding of *lift-and-project cuts* (L&PCs). A geometric presentation of different normalizations also appears in the dissertation of Soares [136, Chapter 3]. We give particular focus to the so-called *trivial* normalization, for which we provide the optimal solutions to the MLP and CGLP for an arbitrary general disjunction. The cut from the CGLP is a SIC when the point being separated is a basic solution of P, but, in general, the optimal solution may correspond to an irregular cut.

The notation is the same as in Section 1.4. Namely, $A^t := [A; D^t]$, $b^t := [b; D^t_0]$, and m_t is the number of rows of D^t. The variables of the CGLP as in Section 1.4 are $\alpha \in \mathbb{R}^n$, $\beta \in \mathbb{R}$, and $v^t \in \mathbb{R}^{m+m_t}_{\geq 0}$ for all $t \in \mathcal{T}$. As before, we can also disaggregate v^t into the first m and last m_t components, which we will refer to as $u^t \in \mathbb{R}^m_{\geq 0}$ and $u^t_0 \in \mathbb{R}^{m_t}_{\geq 0}$ (the multipliers on $Ax \geq b$ and on $D^t x \geq D^t_0$). The normalization constraint to (CGLP), with coefficients $n^\alpha \in \mathbb{R}^n$, $n_\beta \in \mathbb{R}$, $n^t \in \mathbb{R}^{m+m_t}$ for $t \in \mathcal{T}$, and $n_0 \in \mathbb{R}$, is

$$\alpha^{\mathsf{T}} n^\alpha + \beta n_\beta + \sum_{t \in \mathcal{T}} v^t n^t = n_0.$$

Three common normalization constraints for (CGLP) are the β-normalization (βNC) for a given $\bar{\beta}$, the standard normalization (SNC) and the trivial normalization (TNC). We denote by e the all-ones column vector with as many entries as needed, in context.

$$\beta = \bar{\beta} \tag{βNC}$$

$$\sum_{t \in \mathcal{T}} v^t e = 1 \tag{SNC}$$

$$\sum_{t \in \mathcal{T}} u^t_0 e = 1 \tag{TNC}$$

Other options for the normalization exist. Another normalization involves fixing the violation by setting n^α to a point in P. A recently proposed normalization, by Fischetti et al. [84], adapts (SNC) to be less sensitive to the scaling of the constraints of P.

We will refer to the MLP, which we restate here for convenience.

$$
\begin{aligned}
\min_{\{y^t, y_0^t\}_{t\in\mathcal{T}}, \gamma} \quad & n_0\gamma \\
& A^t y^t - b^t y_0^t + n^t\gamma \geq 0 && \text{for all } t \in \mathcal{T} \\
& y_0^t \geq 0 && \text{for all } t \in \mathcal{T} \\
& \sum_{t\in\mathcal{T}} y^t + n^\alpha \gamma = \bar{x} \\
& \sum_{t\in\mathcal{T}} y_0^t + n_\beta \gamma = 1
\end{aligned}
$$

We work with a given solution to (MLP), $(\{\bar{y}^t, \bar{y}_0^t\}_{t\in\mathcal{T}}, \bar{\gamma})$. Below is a recap of the helper variables we defined.

$$
\begin{aligned}
\mathcal{T}^* &:= \{t \in \mathcal{T} : \bar{y}_0^t > 0\}. \\
\hat{x}^t &:= \begin{cases} \bar{y}^t/\bar{y}_0^t & \text{for all } t \in \mathcal{T}^*, \\ \bar{y}^t & \text{for all } t \in \mathcal{T} \setminus \mathcal{T}^*. \end{cases} \\
\hat{\gamma}_t &:= \begin{cases} \bar{\gamma}/\bar{y}_0^t & \text{for all } t \in \mathcal{T}^*, \\ \bar{\gamma} & \text{for all } t \in \mathcal{T} \setminus \mathcal{T}^*. \end{cases} \\
\hat{y}_0^t &:= \bar{y}_0^t/(1 - n_\beta\bar{\gamma}) \quad \text{for all } t \in \mathcal{T}. \\
\hat{P}^t &:= \begin{cases} \{x : A^t x \geq b^t - n^t\hat{\gamma}_t\} & \text{for all } t \in \mathcal{T}^*, \\ \{x : A^t x \geq -n^t\hat{\gamma}_t\} & \text{for all } t \in \mathcal{T} \setminus \mathcal{T}^*. \end{cases}
\end{aligned}
$$

We also restate (and refer to) two reformulated versions of the feasible region of (MLP):

$$
\begin{aligned}
A^t\hat{x}^t &\geq b^t - n^t\hat{\gamma}_t && \text{for all } t \in \mathcal{T}^* \\
A^t\hat{x}^t &\geq -n^t\bar{\gamma} && \text{for all } t \in \mathcal{T} \setminus \mathcal{T}^* \\
\bar{y}_0^t &\geq 0 && \text{for all } t \in \mathcal{T} \\
\sum_{t\in\mathcal{T}^*} \bar{y}_0^t\hat{x}^t &= \bar{x} - n^\alpha\bar{\gamma} \\
\sum_{t\in\mathcal{T}^*} \bar{y}_0^t &= 1 - n_\beta\bar{\gamma}
\end{aligned} \tag{B.1}
$$

and

$$
\begin{aligned}
A^t\hat{x}^t &\geq b^t - n^t\hat{\gamma}_t && \text{for all } t \in \mathcal{T}^* \\
A^t\hat{x}^t &\geq -n^t\bar{\gamma} && \text{for all } t \in \mathcal{T} \setminus \mathcal{T}^* \\
\hat{y}_0^t &\geq 0 && \text{for all } t \in \mathcal{T} \\
\sum_{t\in\mathcal{T}^*} \hat{y}_0^t\hat{x}^t &= (\bar{x} - n^\alpha\bar{\gamma})/(1 - n_\beta\bar{\gamma}) \\
\sum_{t\in\mathcal{T}^*} \hat{y}_0^t &= 1.
\end{aligned} \tag{B.2}
$$

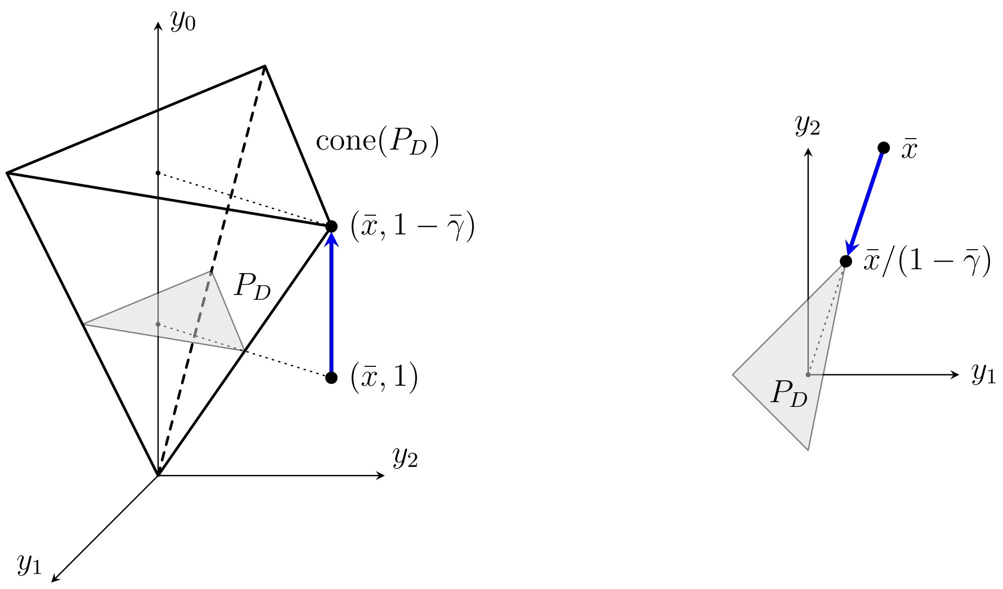


Figure B.1: Depiction of β-normalization.

B.1 β-normalization constraint

When we use (βNC), $n_\beta = 1$, $n_0 = \bar{\beta}$, while n^α and n^t (for all t) are equal to zero. The interpretation (using (B.1)) is that we are trying to represent the point $(\bar{x}, 1-\bar{\gamma})$ in cone(P_D). Note that for all $t \in \mathcal{T}$, $\hat{P}^t = P^t$ because $n^t = 0$. Figure B.1 gives a geometric representation: the normalization corresponds to the (minimum necessary) vertical shift until the translated point is in the region cone(P_D).

Another viewpoint, also shown in Figure B.1, comes from (B.2) and positive homogeneity:

$$(\bar{x}, 1-\bar{\gamma}) \in \operatorname{cone}(P_D) \iff (\bar{x}/(1-\bar{\gamma}), 1) \in \operatorname{cone}(P_D) \iff \bar{x}/(1-\bar{\gamma}) \in P_D.$$

Yet another interpretation is that, instead of scaling $\bar{x}$, we can equivalently apply the scaling to $\hat{P}_D$ (here equal to P_D):

$$\bar{x}/(1-\bar{\gamma}) \in P_D \iff \bar{x} \in (1-\bar{\gamma})P_D.$$

This last reformulation makes it clear that the β-normalization is in fact the same as the gauge function for the set P_D, i.e., the minimum scaling of P_D such that $\bar{x}$ belongs to the scaled set. Such a scaling does not always exist, but it is guaranteed to exist when $0 \in \operatorname{int} P_D$.

A consequence of the interpretation of the MLP as the gauge function when the normalization is (βNC) is that the dual, the CGLP, is then the support function of P_D. It also then follows that the cut generated by the CGLP will be one of the facets of P_D tight at $\bar{x}/(1-\bar{\gamma})$, which means one can identify the cut obtained from the optimal solution to the CGLP "geometrically", from looking at P_D and the position of $\bar{x}$.

B.2 Standard normalization constraint

When we use (SNC), $n_0 = 1$, $n^t = e$ for all $t \in \mathcal{T}$, $n^\alpha = 0$, and $n_\beta = 0$. In this case, the interpretation is: enlarge the feasible region by translating each constraint by an additive constant until $\bar{x}$ is in the enlarged region. As noted by Fischetti et al. [84], the optimum γ depends on the scaling of the constraints (they propose an alternative they call the Euclidean standard normalization constraint to remedy this difficulty).

The objective to the MLP under this normalization is $\min \gamma$. Recall that we let $\hat{\gamma}_t = \bar{\gamma}/\bar{y}_0^t$ for all $t \in \mathcal{T}^*$, and, as $n^t = e$ for all $t \in \mathcal{T}$, $\hat{\gamma}_t$ then represents the amount by which the constraints of P^t are translated. Thus, the objective can roughly be interpreted as finding the minimal amount of translation necessary to place $\bar{x}$ in the convex hull of the modified feasible regions $\hat{P}^t$. This is of course sensitive to the scaling of the constraints, which is exactly the motivation for the alternative, normalized, version of the standard normalization that is proposed in [84].

B.3 Trivial normalization constraint

We now turn to the trivial normalization (TNC), i.e., $n_0 = 1$, $n_i^t = 1$ for all $i \in \{m + 1, \dots, m + m_t\}$, while $n_i^t = 0$ for all $i \in [m]$, $n^\alpha = 0$, and $n_\beta = 0$. When working with a split disjunction, the normalization is $u_0^1 + u_0^2 = 1$. For a split disjunction, it is known [32, 47, 84] that the CGLP gives the SIC associated with $\bar{x}$. We derive an optimal solution to (MLP) and (CGLP) under the trivial normalization for an arbitrary disjunction.

B.3.1 Optimal solution to the MLP

From (B.2) under this normalization, we have that $\bar{x}$ is a convex combination of points $\hat{x}^t \in \hat{P}^t$, $t \in \mathcal{T}^*$, where $\hat{P}^t = \{x \in P : D^t x \ge D_0^t - \hat{\gamma}_t\}$. The key observation is that, unlike with some other normalizations, here $\hat{P}^t \subseteq P$ for each $t \in \mathcal{T}^*$. Since $\bar{x}$ is a vertex of P, it cannot be represented as a convex combination of any other vertices of P. It follows that $\hat{x}^t = \bar{x}$ and $D^t \hat{x}^t \ge D_0^t - \hat{\gamma}_t$, $t \in \mathcal{T}^*$. The geometric interpretation is that we are translating the inequalities $D^t x \ge D_0^t$ until $\bar{x}$ is expressible as a convex combination of points from the enlarged (with respect to P^t) regions. The upshot is that only those constraints of P that are tight at $\bar{x}$ can have nonzero Farkas multipliers (the variables of the CGLP). Figure B.2 illustrates this geometric intuition.

More formally, define, for all $t \in \mathcal{T}$, $\delta_t := \max_i\{D_{0i}^t - D_{i\cdot}^t \bar{x} : i \in [m_t]\}$ as the maximum violation of any constraint defining disjunctive term t.

Theorem B.1. *An optimal solution to* (MLP) *under the normalization* (TNC) *is*

$$\bar{y}^t = \left(\frac{1/\delta_t}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} \right) \bar{x} \quad \textit{for all } t \in \mathcal{T}$$

$$\bar{y}_0^t = \frac{1/\delta_t}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} \qquad \textit{for all } t \in \mathcal{T}$$

$$\bar{\gamma} = \frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}}.$$

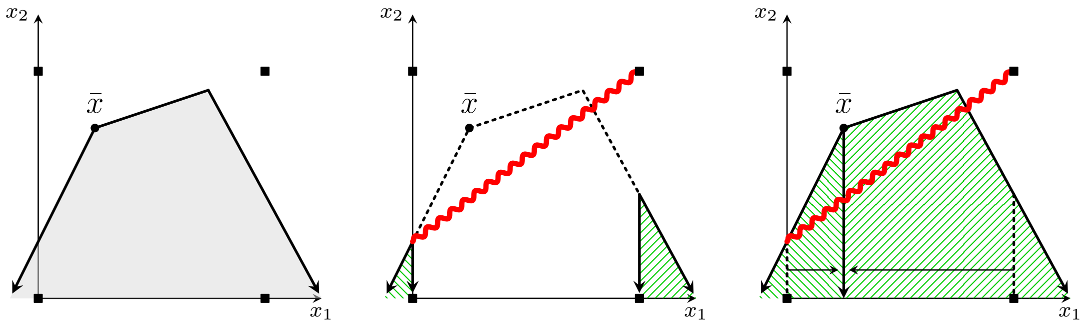


Figure B.2: Depiction of trivial normalization using the split $(x_1 \le 0) \vee (x_1 \ge 1)$. Each P^t is enlarged until it contains $\bar{x}$. The wavy line is the SIC from the basis at $\bar{x}$.

Proof. Note that $\mathcal{T}^* = \mathcal{T}$ using this solution, $\bar{y}^t = \bar{y}_0^t \bar{x}$, and $\bar{\gamma} = \bar{y}_0^t \delta_t$ for all $t \in \mathcal{T}$. First, we verify the solution is feasible. As $\bar{x}$ does not belong to the disjunctive hull, $D^t \bar{x} \ngeq D_0^t$ for every $t \in \mathcal{T}$; thus, $\delta_t > 0$, implying that $\bar{y}_0^t \ge 0$, for all $t \in \mathcal{T}$. Since, clearly, $\sum_{t \in \mathcal{T}} \bar{y}_0^t = 1$, we have that $\sum_{t \in \mathcal{T}} \bar{y}^t = \bar{x}$. Finally, we check the remaining set of constraints of (MLP): $A^t y^t - b^t y_0^t + n^t \gamma \ge 0$. Recall that $A^t = [A; D^t]$ and $b^t = [b; D_0^t]$. For the constraints of P, the corresponding entry of n^t is 0, so $A\bar{y}^t - b\bar{y}_0^t = \bar{y}_0^t (A\bar{x} - b) \ge 0$. For the constraints defining the disjunction, with e denoting the all-ones vector,

$$D^t \bar{y}^t - D_0^t \bar{y}_0^t + e\bar{\gamma} = \bar{y}_0^t (D^t \bar{x} - D_0^t + e\delta_t) = \bar{y}_0^t (e\delta_t - (D_0^t - D^t \bar{x})),$$

which is nonnegative by the definition of δ_t.

We now prove optimality. As is evident from the first set of constraints of reformulation (B.1), for any feasible solution to (MLP) and the corresponding indices $\mathcal{T}^*$ for which $y_0^t > 0$,

$$\gamma \ge \max_{t \in \mathcal{T}^*} \max_{i \in [m_t]} \left\{ y_0^t (D_{0i}^t - D_{i\cdot}^t \bar{x}) \right\} = \max_{t \in \mathcal{T}^*} \left\{ y_0^t \max_{i \in [m_t]} \left\{ D_{0i}^t - D_{i\cdot}^t \bar{x} \right\} \right\} = \max_{t \in \mathcal{T}^*} \left\{ y_0^t \delta_t \right\},$$

where we have used the fact that $n_i^t = 0$ for all $i \in [m]$ and our previous observation that $\hat{x}^t \in P$ and $\bar{x}$ is a vertex, from which it follows that $\hat{x}^t = \bar{x}$ for all $t \in \mathcal{T}^*$. The feasible solution must satisfy $\sum_{t \in \mathcal{T}^*} y_0^t = 1$ and $y_0^t \ge 0$ for all $t \in \mathcal{T}$. Consequently, γ is minimized when $y_0^t \delta_t$ takes the same value for all $t \in \mathcal{T}^*$ (otherwise, for whichever term has the highest value, the weight y_0^t can be reduced and distributed among the other terms). Hence, $y_0^{t'} = y_0^t \delta_t / \delta_{t'}$ for all $t, t' \in \mathcal{T}^*$; summing up over all t', we get that

$$1 = \sum_{t' \in \mathcal{T}^*} y_0^t = \sum_{t' \in \mathcal{T}^*} \frac{y_0^t \delta_t}{\delta_{t'}} \qquad \Longrightarrow \qquad y_0^t = \frac{1/\delta_t}{\sum_{t' \in \mathcal{T}^*} 1/\delta_{t'}}.$$

As a result,

$$\gamma \ge \max_{t \in \mathcal{T}^*} \left\{ y_0^t \delta_t \right\} = \frac{1}{\sum_{t' \in \mathcal{T}^*} 1/\delta_{t'}} \ge \frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}}.$$

This completes the proof, as the objective value of the solution from the theorem statement matches this bound. □

B.3.2 Optimal solution to the CGLP for simple disjunctions

Theorem B.1 validates our earlier intuition. For example, in the case of a simple disjunction $\vee_{t\in\mathcal{T}}(d^t x \ge d_0^t)$, for each $t \in \mathcal{T}$, δ_t is the amount by which $\bar{x}$ violates the inequality defining disjunctive term t. From (B.1), $\hat{\gamma}_t = \delta_t$, so each disjunctive term inequality is translated by precisely the amount δ_t needed to make $d^t\bar{x} = d_0^t$.

Now we turn to the dual to (MLP), i.e., (CGLP), from which we can derive the cut that the trivial normalization yields. As in Section 1.3, we add a slack variable s_{n+i} for each row, so that $Ax - s = b$. Given the disjunction $\vee_{t\in\mathcal{T}}(d^t x \ge d_0^t)$ and a cobasis $\mathcal{N} \subseteq \{n+1,\dots,n+m\}$, we next give the formula for SIC in the structural space (previously, in Section 1.3, we derived it in the nonbasic space). As before, index the rows with nonbasic slacks by $N := \{i \in [m] : n+i \in \mathcal{N}\}$, let A_N denote the $n \times n$ nonsingular submatrix of A with rows indexed by N, and let b_N be the subset of elements of b indexed by N. Define $s_{\mathcal{N}}$ as the nonbasic slack variables, and recall the optimal tableau in the nonbasic space that we stated in (1.2), obtained by multiplying through the rows indexed by N with A_N^{-1}, and using the fact that $A_N\bar{x} = b_N$:

$$x = \bar{x} + A_N^{-1}s_{\mathcal{N}} = \bar{x} + \sum_{j\in\mathcal{N}} r^j s_j.$$

This gives a representation of the optimal tableau for the structural variables only. The vector r^{n+i}, where $i \in N$, is simply column i of the matrix A_N^{-1}; we represent this by $(A_N^{-1})_{\cdot,i}$. We also use $(d^tA_N^{-1})_i = d^t(A_N^{-1})_{\cdot,i}$ for component i of the vector $d^tA_N^{-1}$. In addition, define $\lambda_i^t := (d^tA_N^{-1})_i/\delta_t$ and $\lambda_i := \max_{t\in\mathcal{T}} \lambda_i^t$. The value of λ_i will be the coefficient on variable s_{n+i} for the SIC in the nonbasic space.

Lemma B.2. *The SIC from the given basis and disjunction is $(\alpha^0)^\mathsf{T} x \ge \beta_0$, where*

$$(\alpha^0)^\mathsf{T} := \sum_{i\in N} \lambda_i A_{i\cdot}, \qquad \beta_0 := 1 + \sum_{i\in N}\lambda_i b_i.$$

Proof. We first convert the disjunction $\vee_{t\in\mathcal{T}}(d^t x \ge d_0^t)$ from the structural space to the nonbasic space. Using the tableau in (1.2),

$$d^tx - d_0^t = d^t\left(\bar{x} + \sum_{j\in\mathcal{N}} r^j s_j\right) - d_0^t = \sum_{j\in\mathcal{N}} d^tr^js_j - (d_0^t - d^t\bar{x}) = \sum_{j\in\mathcal{N}} d^tr^js_j - \delta_t.$$

The SIC in the nonbasic space is then (by applying the formula from Section 1.3)

$$\sum_{j\in\mathcal{N}} \max_{t\in\mathcal{T}}\left\{\frac{d^tr^j}{\delta_t}\right\} s_j \ge 1 \quad\Longrightarrow\quad \sum_{i\in N}\lambda_i s_{n+i} \ge 1 \quad\Longrightarrow\quad \sum_{i\in N}\lambda_i(A_{i\cdot}x - b_i) \ge 1,$$

from which the result follows. $\square$

For $i \in N$, let $t_i \in \arg\max_t \{\lambda_i^t : t \in \mathcal{T}\}$. Hence, the SIC is equivalent to

$$\left(\sum_{i\in N}\lambda_i^{t_i}A_{i\cdot}\right)x \ge 1 + \sum_{i\in N}\lambda_i^{t_i}b_i.$$

In the case of a simple disjunction, the feasible region of the CGLP under the trivial normalization is:

$$\begin{aligned} u^t A + u_0^t d^t &= \alpha^\intercal && \text{for all } t \in \mathcal{T} \\ u^t b + u_0^t d_0^t &\geq \beta && \text{for all } t \in \mathcal{T} \\ \sum_{t \in \mathcal{T}} u_0^t &= 1 \\ u^t, u_0^t &\geq 0 && \text{for all } t \in \mathcal{T} \end{aligned} \tag{B.3}$$

Theorem B.3. *Under the trivial normalization, an optimal solution to* (B.3) *is*

$$\begin{aligned} \bar{u}_0^t &:= \frac{1/\delta_t}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} && \textit{for all } t \in \mathcal{T} \\ \bar{u}_i^t &:= \frac{\lambda_i^{t_i} - \lambda_i^t}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} = \frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} \left(\frac{d^{t_i}}{\delta_{t_i}} - \frac{d^t}{\delta_t} \right) (A_N^{-1})_{\cdot,i} && \textit{for all } t \in \mathcal{T},\ i \in N \\ \bar{u}_i^t &:= 0 && \textit{for all } t \in \mathcal{T},\ i \notin N \\ \bar{\alpha} &:= \frac{\alpha^0}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} = \sum_{i \in N} \bar{u}_0^{t_i} (d^{t_i} A_N^{-1})_i A_{i\cdot} \\ \bar{\beta} &:= \frac{\beta_0}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} = \frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} + \sum_{i \in N} \bar{u}_0^{t_i} (d^{t_i} A_N^{-1})_i b_i. \end{aligned}$$

and yields the SIC arising from the basis corresponding to N.

Proof. We first show feasibility. Clearly, $\bar{u}_0^t > 0$ for all $t \in \mathcal{T}$ and $\sum_{t \in \mathcal{T}} \bar{u}_0^t = 1$. For any solution in which $u_i^t = 0$ for all $i \notin N$, we have that

$$u^t A + u_0^t d^t = \alpha^\intercal \quad \implies \quad u_N^t A_N = \alpha^\intercal - u_0^t d^t.$$

Let $\hat{t} \in \mathcal{T}$ be an arbitrary term index. It follows that

$$\begin{aligned} \bar{u}_N^{\hat{t}} &= \bar{\alpha}^\intercal A_N^{-1} - \bar{u}_0^{\hat{t}} d^{\hat{t}} A_N^{-1} \\ &= \frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} \left(\sum_{i \in N} \frac{(d^{t_i} A_N^{-1})_i A_{i\cdot} A_N^{-1}}{\delta_{t_i}} - \frac{d^{\hat{t}} A_N^{-1}}{\delta_{\hat{t}}} \right) \\ &= \frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} \left(\sum_{i \in N} \frac{(d^{t_i} A_N^{-1})_i e_i^\intercal}{\delta_{t_i}} - \sum_{i \in N} \frac{(d^{\hat{t}} A_N^{-1})_i e_i^\intercal}{\delta_{\hat{t}}} \right) \\ &= \frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} \left(\sum_{i \in N} \left(\frac{d^{t_i}}{\delta_{t_i}} - \frac{d^{\hat{t}}}{\delta_{\hat{t}}} \right) (A_N^{-1})_{\cdot,i} e_i^\intercal \right). \end{aligned}$$

This matches our definition of $\bar{u}^t$ and shows feasibility of the constraints involving α for all disjunctive terms, as $\hat{t}$ was arbitrarily chosen. Note that $\bar{u}_i^t = 0$ for all $i \in N$ such that $t_i = t$, and $\bar{u}_i^t \geq 0$ for all $t \in \mathcal{T}$ and $i \in N$, by the definition of t_i.

Next, for the β constraints, we show that $(\bar{u}^t b + \bar{u}_0^t d_0^t) - \bar{\beta} = 0$ for any $t \in \mathcal{T}$. Expanding terms, and using the facts that $A_N^{-1} b_N = \bar{x}$ and $\delta_t = d_0^t - d^t \bar{x}$,

$$\begin{aligned}
(\bar{u}^t b + \bar{u}_0^t d_0^t) - \bar{\beta} &= \frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} \left(\sum_{i \in N} \left(\frac{d^{t_i}}{\delta_{t_i}} - \frac{d^t}{\delta_t} \right) (A_N^{-1})_{\cdot,i} b_i + \frac{d_0^t}{\delta_t} - \left(1 + \sum_{i \in N} \frac{d^{t_i} (A_N^{-1})_{\cdot,i} b_i}{\delta_{t_i}} \right) \right) \\
&= \frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} \left(-\sum_{i \in N} \frac{d^t (A_N^{-1})_{\cdot,i} b_i}{\delta_t} + \frac{d_0^t}{\delta_t} - 1 \right) \\
&= \frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} \left(-\frac{d^t \bar{x}}{\delta_t} + \frac{d_0^t}{\delta_t} - 1 \right) = 0.
\end{aligned}$$

This completes the proof of feasibility. Moreover, it is clear from the definition of $\bar{\alpha}$ and $\bar{\beta}$ that the cut $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ is simply the SIC scaled by $1/\sum_{t' \in \mathcal{T}}(1/\delta_{t'})$. Lastly, we show optimality. The objective value is

$$\bar{\alpha}^\mathsf{T} \bar{x} - \bar{\beta} = \frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}} \left(\sum_{i \in N} \frac{(d^{t_i} A_N^{-1})_i A_{i\cdot} \bar{x}}{\delta_{t_i}} - \left(1 + \sum_{i \in N} \frac{(d^{t_i} A_N^{-1})_i b_i}{\delta_{t_i}} \right) \right) = -\frac{1}{\sum_{t' \in \mathcal{T}} 1/\delta_{t'}}.$$

Note that this is exactly the negative of the optimal objective value to (MLP) (in which we negated the objective function that would be obtained by directly taking the dual to (CGLP)), as proved in Theorem B.1, which proves optimality by duality. ☐

Theorem B.3 gives the same values of the CGLP variables as in Balas and Kis [30, Theorem 7], but proves that the solution is optimal under the trivial normalization (though we do not take the extra step of constructing a basis). As a direct corollary to Theorem B.3, we also recover the previously known result for the split disjunction case, as in [24, Section 3.3], [33, Theorem 16], and [84, Theorem 2].

Theorem B.4. *The following is an optimal solution to* (B.3) *under normalization* (TNC) *when the disjunction is* $(-\pi^\mathsf{T} x \geq -\pi_0) \vee (\pi^\mathsf{T} x \geq \pi_0 + 1)$. *Define* $f_0 := \pi^\mathsf{T} \bar{x} - \pi_0$.

$$\begin{aligned}
\bar{u}_0^1 &:= 1 - f_0 \\
\bar{u}_0^2 &:= f_0 \\
\bar{u}_i^1 &:= \bar{u}_i^2 = 0 && \text{for all } i \in [m] \setminus N \\
\bar{u}_i^1 &:= \max\{0, (\pi^\mathsf{T} A_N^{-1})_i\} && \text{for all } i \in N \\
\bar{u}_i^2 &:= \max\{0, -(\pi^\mathsf{T} A_N^{-1})_i\} && \text{for all } i \in N \\
\bar{\alpha} &:= f_0 \bar{u}^1 A + (1 - f_0) \bar{u}^2 A \\
\bar{\beta} &:= f_0(1 - f_0) + f_0 \bar{u}^1 b + (1 - f_0) \bar{u}^2 b
\end{aligned}$$

Moreover, $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ *is equivalent to the SIC from the basis corresponding to* N.

Proof. One can directly plug in the values $\delta_1 = f_0$, $\delta_2 = 1 - f_0$, and so on, into Theorem B.3. Some variables have simpler definitions than in the general case because $d^1 = -\pi^\mathsf{T} = -d^2$ and the scaling term $1/(1/\delta_1 + 1/\delta_2) = f_0(1 - f_0) = \delta_1 \delta_2$, allowing some quantities to cancel. ☐

B.3.3 Optimal solution to the CGLP for general disjunctions

In Theorem B.1, we gave an optimal solution to (MLP) under the trivial normalization of the CGLP for an arbitrary (not necessarily simple) disjunction. We then gave the corresponding dual solution, but only for a simple disjunction, in Theorem B.3. In fact, it is not hard to derive the optimal solution to (CGLP) under the trivial normalization for any disjunction. The key insight is based on our geometric interpretation of the trivial normalization: namely, for each $t \in \mathcal{T}$, the disjunctive term inequalities $D^t x \geq D^t_0$ are translated until the first one of them is satisfied at equality by $\bar{x}$. For each term, fix $i_t \in [m_t]$ to be the index of one of these inequalities, i.e., $i_t \in \arg\max_i \{D^t_{0i} - D^t_{i\cdot}\bar{x} : i \in [m_t]\}$. Then define $d^t := D^t_{i_t,\cdot}$ (row i_t of D^t) and $d^t_0 := D^t_{0i_t}$. (This is a valid simple disjunction.)

Theorem B.5. *The values $(\bar{\alpha}, \bar{\beta}, \{\bar{u}^t, \bar{u}^t_0\}_{t\in\mathcal{T}}\}$ as defined in Theorem B.3, except*

$$\begin{aligned} \bar{u}^t_{0i_t} &:= \frac{1/\delta_t}{\sum_{t'\in\mathcal{T}} 1/\delta_{t'}} && \text{for all } t \in \mathcal{T} \\ \bar{u}^t_{0i} &:= 0 && \text{for all } t \in \mathcal{T},\ i \neq i_t \end{aligned}$$

are an optimal solution to (CGLP) *under the trivial normalization and yield the SIC arising from the basis corresponding to N.*

Proof. Everything proved in Theorem B.3 goes through as before when we ignore the original disjunction and work only with the derived simple disjunction $\vee_{t\in\mathcal{T}}(D^t_{i_t,\cdot}x \geq D^t_{0i_t})$. □

The consequence of Theorem B.5 is that, given any basis to P, we immediately (without having to solve the higher-dimensional (CGLP)) can find the Farkas multipliers certifying the validity of the SIC from that basis. The utility of having these Farkas multipliers for the SIC from any disjunction is that we can now apply strengthening techniques such as based on the modularization of the coefficients corresponding to nonbasic integer-restricted variables. We discussed this in generality in Section 1.5.

An important factor to consider is that under the trivial normalization, for the case of general disjunctions, we encounter a scaling problem that is not present for simple disjunctions. Namely, for simple disjunctions, there is a uniquely defined SIC from a basis of P, which precisely yields an optimal solution to (B.3). On the other hand, for general disjunctions, every choice of nonnegative multipliers $\{u^t_0\}_{t\in\mathcal{T}}$ yields a convex P_I-free set, each of which has a corresponding SIC. Theorem B.5 defines one setting for these multipliers, in which only one inequality defining each disjunctive term has positive weight. However, this inequality depends on a calculation of violation, which is sensitive to scaling. As a result, the SIC obtained from the solution defined in Theorem B.5 may be dramatically weaker than a SIC from the best possible choice of multipliers $\{u^t_0\}_{t\in\mathcal{T}}$. We give an example of this next.

B.3.4 Trivial normalization example with a general disjunction

Consider P_I defined as the set of $(x_1, x_2) \in \mathbb{Z}^2$ solutions to

$$\begin{aligned} x_1 - x_2 &\geq -1/2 \\ -x_1 - x_2 &\geq -3/2. \end{aligned}$$

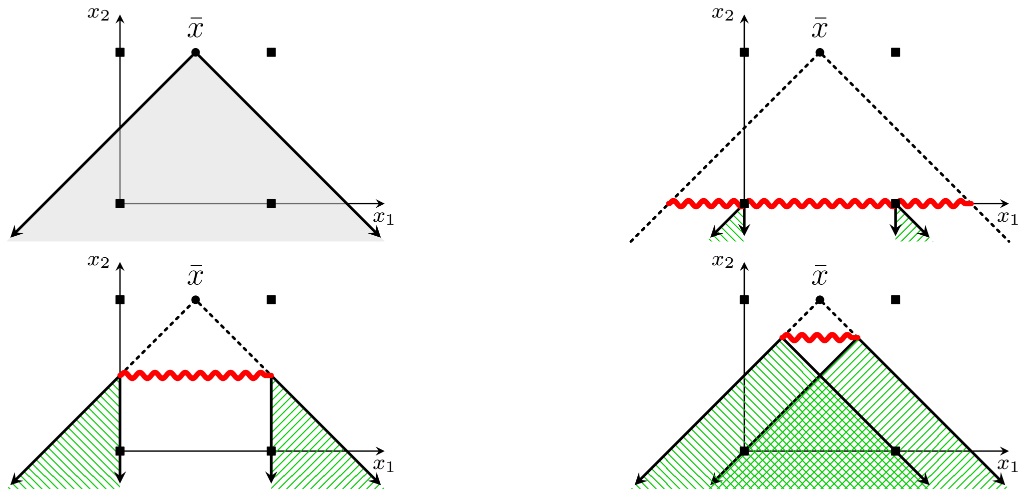


Figure B.3: Example showing that trivial normalization for a general disjunction is sensitive to scaling and may yield a weak SIC. The first panel shows the feasible region of P. The top right panel shows one SIC that can be obtained from the two-term general disjunction (B.4) with $u_0^1 = u_0^2 = (1/6, 1/3)$. The bottom left panel shows the cut obtained from the split disjunction $(-x_1 \geq 0) \vee (x_1 \geq 1)$. This is the same as the cut obtained from applying Theorem B.5. The last panel shows the cut that would be obtained from the disjunction $(x_1 - x_2 \geq 0) \vee (-x_1 - x_2 \geq -1)$, which would be the SIC obtained from applying Theorem B.5 under a different scaling (e.g., by multiplying the first inequality in both terms by 1/3).

Suppose we take the disjunction

$$\left(\begin{array}{r} -x_1 \geq 0 \\ \frac{1}{2}x_1 - \frac{1}{2}x_2 \geq 0 \end{array} \right) \vee \left(\begin{array}{r} x_1 \geq 1 \\ -\frac{1}{2}x_1 - \frac{1}{2}x_2 \geq -\frac{1}{2} \end{array} \right). \tag{B.4}$$

Recall that $\delta_t := \max_i\{D_{0i}^t - D_{i\cdot}^t \bar{x} : i \in [m_t]\}$. In this case, $\delta_1 = \max\{0 - (-1/2), 0 - (1/2)(1/2) - (-1/2)(1)\} = \max\{1/2, 1/4\} = 1/2$, achieved by the inequality $-x_1 \geq 0$, and similarly $\delta_2 = 1/2$, achieved by the inequality $x_1 \geq 1$. The matrix A_N^{-1} is

$$\begin{bmatrix} 1/2 & -1/2 \\ -1/2 & -1/2 \end{bmatrix}.$$

From this, applying Theorem B.5, an optimal solution to (B.3) is

$$\begin{aligned} \bar{u}_0^1 &= \bar{u}_0^2 = (1/2, 0) \\ \bar{u}^1 &= (1/2, 0) \\ \bar{u}^2 &= (0, 1/2) \\ \bar{\alpha}^\mathsf{T} &= (0, -1/2) \\ \bar{\beta} &= 0. \end{aligned}$$

This is the cut $x_2 \leq 1/2$, which is weaker than the disjunctive cut $x_2 \leq 0$ obtainable via the

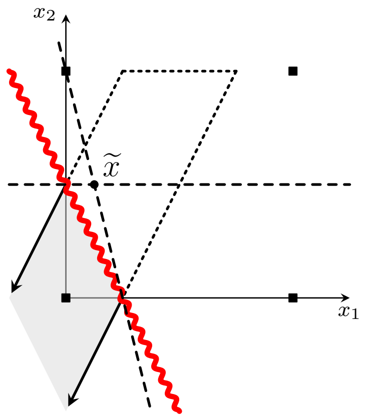


Figure B.4: Example of the trivial normalization yielding an irregular cut. The disjunction is $(-x_1 \geq 0) \vee (-x_2 \geq 0)$. The point $\widetilde{x}$ is the optimal vertex after adding the two SICs from P (shown as dashed lines). The wavy line is an irregular cut and the optimal solution to the CGLP (without the two SICs) under the trivial normalization when separating $\widetilde{x}$.

following (suboptimal) feasible solution to (B.3):

$$\begin{aligned}
\hat{u}_0^1 &= \hat{u}_0^2 = (1/6, 1/3) \\
\hat{u}^1 &= \hat{u}^2 = (0, 0) \\
\hat{\alpha}^{\mathsf{T}} &= (0, -1/6) \\
\hat{\beta} &= 0.
\end{aligned}$$

Note that Theorem B.5 would still produce the cut $x_2 \leq 1/2$ if every disjunctive term inequality were scaled to have norm 1, i.e., if δ_t corresponded to the maximum Euclidean distance by which $\bar{x}$ is cut off.

If a different scaling of (B.4) is used, e.g., if we multiply $-x_1 \geq 0$ and $x_1 \geq 1$ by 1/3, then the biggest violation comes from the second inequality for each disjunctive term and the theorem yields the even weaker SIC $x_2 \leq 3/4$. Regardless of scaling, however, Theorem B.5 cannot generate the strongest SIC, $x_2 \leq 0$, because for that cut, both components of u_0^1 and u_0^2 are positive. The example is depicted in Figure B.3.

B.3.5 Irregular cuts from the trivial normalization

We now return to the case of a simple disjunction to highlight the importance of the assumption we use in Theorem B.1 and Theorem B.3, that that the point we are separating, $\bar{x}$, is a vertex of P. Suppose that the objective function to (CGLP) is instead $\min \alpha^{\mathsf{T}}\widetilde{x} - \beta$, where $\widetilde{x} \in P$ but perhaps not a vertex. For example, $\widetilde{x}$ might be an optimal solution to $\min\{c^{\mathsf{T}}x : x \in \widetilde{P}\}$ where $\widetilde{P}$ is the polyhedron obtained after adding to P all SICs from feasible or infeasible bases of P.

Note that the geometric intuition for the trivial normalization still holds: the optimal solution to (MLP), as in the reformulation (B.1), corresponds to finding points $\hat{x}^t = y^t/y_0^t$ (for $t \in \mathcal{T}^*$, the indices for which $y_0^t > 0$) such that $\hat{x}^t \in P$ and $\widetilde{x} = \sum_{t\in\mathcal{T}^*} y_0^t \hat{x}^t$. When $\widetilde{x}$ is not a vertex of P, what is no longer true is the property that $\hat{x}^t = \widetilde{x}$ for all $t \in \mathcal{T}^*$. This implies that the solution defined in Theorem B.1, while still feasible, may not be optimal.

Intuitively, if $\widetilde{x}$ does not belong to the disjunctive hull P_D, then there exists a facet of the disjunctive hull that cuts $\widetilde{x}$, and it will be obtainable as a feasible solution to (CGLP). This solution may correspond to an irregular cut, as we state in Proposition B.6 and illustrate in Figure B.4.

Proposition B.6. *Let $\widetilde{x} \in P$ and $(\widetilde{\alpha}, \widetilde{\beta}, \{\widetilde{u}^t, \widetilde{u}_0^t\})$ denote an optimal solution to* (B.3) *when separating $\widetilde{x}$ using a simple disjunction $\vee_{t \in \mathcal{T}}(d^t x \geq d_0^t)$. The optimal value $\widetilde{\alpha}^\mathsf{T}\widetilde{x} - \widetilde{\beta}$ is negative if and only if $\widetilde{x} \notin P_D$. If $\widetilde{x} \notin P_D$ but satisfies every SIC obtainable from P using the given disjunction, then $\widetilde{\alpha}^\mathsf{T} x \geq \widetilde{\beta}$ is an irregular cut.*

Proof. Every valid inequality for P_D has a corresponding feasible solution to (B.3). Hence, the optimal objective value is negative if and only if there exists a valid inequality that cuts $\widetilde{x}$, which happens if and only if $\widetilde{x} \notin P_D$. Assume now that $\widetilde{x} \notin P_D$, so that $\widetilde{\alpha}^\mathsf{T}\widetilde{x} < \widetilde{\beta}$. When $\widetilde{x} \notin P_D$ but satisfies all possible SICs, the cut $\widetilde{\alpha}^\mathsf{T} x \geq \widetilde{\beta}$ necessarily corresponds to an irregular cut: if it were possible to represent it as a SIC, then it would be satisfied by $\widetilde{x}$. □

B.4 Concluding remarks about CGLP normalizations

In this chapter, we discussed normalizations for (CGLP) and their geometric interpretation via the dual to the CGLP given in (MLP). As we mentioned, we do not cover all possible normalizations, such as the so-called reverse polar normalization discussed in [33] and more recently investigated by Serra [133]. The geometric interpretations are helpful in understanding the CGLP, and they are also useful when building a correspondence between *$\mathcal{V}$-polyhedral cuts* (VPCs) and L&PCs, as we do in Chapter 5.

Moreover, our new results include giving the optimal solutions to the MLP and the CGLP under the trivial normalization. An interesting consequence of Theorems B.1 and B.5 is that we have some semblance of a correspondence, for general disjunctions, between the original space of variables and the higher-dimensional space of variables used for cut generation through the CGLP. This suggests that we might be able to implement a limited variant of pivoting in the original tableau while calculating reduced costs in the CGLP, as in [23, 32]. Moreover, we may even be able to obtain the irregular cuts from a disjunction, as we showed in Proposition B.6.

Appendix C

PHA Appendices

C.1 Tilting for degenerate hyperplanes

We previously showed how to tilt a hyperplane H defining P that is not degenerate, i.e., $\bar{x}$ does not lie on H. We defined H using n affinely independent points obtained by intersecting the n affinely independent rays of $\bar{C}$ with H. These points all lie on one-dimensional faces of $\bar{C}$. In the case that H is degenerate, we will instead use two-dimensional faces of $\bar{C}$ to define the hyperplane, which we will then modify to define a targeted tilting.

When a degenerate hyperplane H is activated on $\bar{C}$, each of the extreme rays of the new cone $\bar{C} \cap H^+$ that lie on H can be defined by H and $n-2$ hyperplanes of $\bar{C}$ that are not redundant for $\bar{C} \cap H^+$. Thus, each ray of the new cone lying on H is on a two-dimensional face of $\bar{C}$. Let $\mathcal{R}^{H*}$ denote an affinely independent set of $n-1$ of these rays lying on H. Let $\bar{\mathcal{R}}^c$ be the rays of $\bar{C}$ that are cut by H^+. For each $r \in \mathcal{R}^{H*}$, since it lies on a two-dimensional face of $\bar{C}$, we can define $(\bar{r}^1, \bar{r}^2) \in \bar{\mathcal{R}}^c \times (\bar{\mathcal{R}} \setminus \bar{\mathcal{R}}^c)$ such that r can be expressed as a convex combination of $\bar{r}^1$ (which is cut by H) and $\bar{r}^2$ (which is not cut by H); in particular, let λ_r^H be the multiplier such that $r = \lambda_r^H \bar{r}^1 + (1-\lambda_r^H)\bar{r}^2$. Figure C.1 depicts this construction.

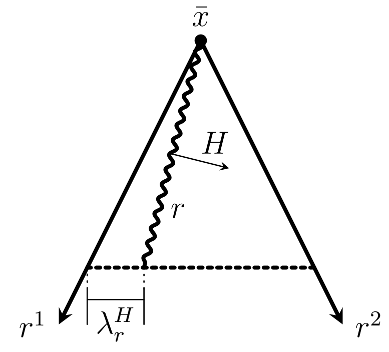


Figure C.1: Illustration of targeted tilting construction for a degenerate hyperplane.

If we know λ_r^H and the rays $\bar{r}^1$ and $\bar{r}^2$ for every $r \in \mathcal{R}^{H^*}$, then we can use this to give an alternate definition of H. To define a targeted tilting of H, we can modify the values λ_r^H for each $r \in \mathcal{R}^{H^*}$ using some δ_r. In order to coordinate with the definition of a targeted tilting that we gave in Section 2.4, the intersection points and rays created by activating the tilted hyperplane should either be identical to those obtained from activating H or coincide with some initial intersection point or ray. It is not difficult to see that this means the allowed values for δ_r are 0 and $-\lambda_r^H$. With such a tilting, Theorem 2.12 will still hold, i.e., activations can be performed by using H and $\mathcal{R}_A$ without a need for explicitly computing the tilted hyperplane.

C.2 Tilting example

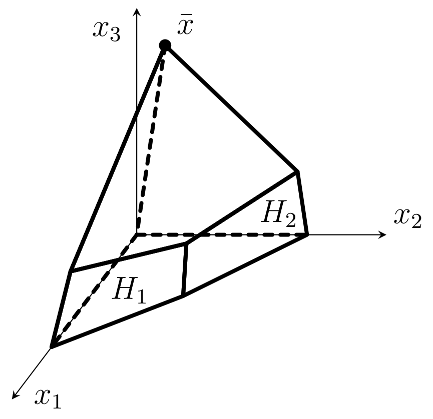
In this section, we demonstrate an example in which a valid tilting combined with the implicit computation used in Theorem 2.12 leads to invalid cuts. The example additionally provides intuition for the targeted tilting rule that if a hyperplane cuts a ray of $\bar{C}$ that has already been cut, then we should not tilt the hyperplane along that ray. This is sufficient, as we have shown, to allow us to apply the implicit computation scheme.

The left panel of Figure C.2 shows the feasible region of $P := \{x \in \mathbb{R}^3 : -2x_2 + x_3 \leq 0; -2x_1 + x_3 \leq 0; 12x_1 + 10x_2 - 5x_3 \leq 9; 10x_1 + 12x_2 - 5x_3 \leq 9; x_1 + x_2 + x_3 \leq 1\}$, and the right panel of the same figure shows the cone $\bar{C}$. The cut-generating set S is the unit box, $\{x \in \mathbb{R}^3 : 0 \leq x_1 \leq 1;\ 0 \leq x_2 \leq 1\}$. The hyperplane activations are of H_1 and then H_2. We tilt H_1 along r^2 so that the intersection of this ray with $\widetilde{H}_1$ coincides $r^2 \cap \operatorname{bd} S$, and H_2 will similarly be tilted along r^1 so that the intersection of r^1 with $\widetilde{H}_2$ is at the point $r^1 \cap \operatorname{bd} S$. The tilted hyperplanes are shown in the top panel of Figure C.3.

The tilting defined above is clearly valid. It also satisfies all but one of the conditions of being a targeted tilting; it does not meet the requirement that H_2 and $\widetilde{H}_2$ must intersect r^1 at the same point, as a result of the activation of H_1 on r^1 prior to the activation of H_2.

However, as shown in the bottom panel of Figure C.3, when the implicit computation algorithm is applied to these tilted hyperplanes, a point of $\operatorname{conv}(P \setminus \operatorname{int} S)$ is cut. This is because $\widetilde{H}_2$ intersects r^1 outside of the interior of S. Hence, the intersection point $\widetilde{H}_2 \cap r^1$ is not added to the point collection, as a result of step 5 of Algorithm 2.1. This intersection point is the same as $H_1 \cap r^1$, which has already been removed from the point collection during the activation of $\widetilde{H}_1$.

There may be many approaches in which a tilted hyperplane activation can be computed implicitly using only information from the untilted hyperplane. Our method prevents the situation in this example from occurring by requiring that r^1 intersects $\widetilde{H}_2$ at the same point as it intersects H_2. An example of an alternative would be to add the intersection point $r^1 \cap \widetilde{H}_2 = r^1 \cap H_1$ back into the point collection when activating $\widetilde{H}_2$, but then the intersection points obtained from activating H_1 on r^1 would be redundant.



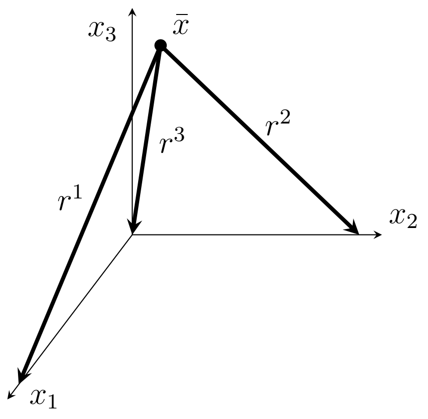


Figure C.2: Feasible region of P and $\bar{C}$.

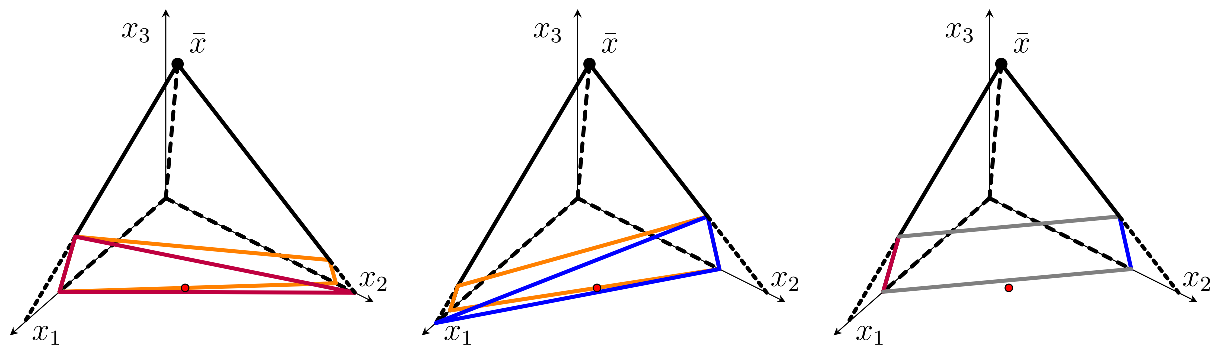


Figure C.3: Hyperplane activations leading to cutting a point in $\operatorname{conv}(P \setminus \operatorname{int} S)$.

Appendix D

VPC Appendices

D.1 Example of invalid cuts from a point-ray collection

This example shows that the using as the point-ray collection the optimal points p^t on each term $t \in \mathcal{T}$ along with the neighbors of p^t may lead to the generation of invalid cuts from the associated (PRLP).

$$
\begin{aligned}
\max_{x_1,x_2,x_3} \quad & x_3 \\
& -x_3 \leq -1/2 \\
& -(7/4)x_1 + 5x_2 - 2x_3 \leq 1 \\
& -x_1 - 5x_2 + 2x_3 \leq -1 \\
& -x_1 - (20/3)x_2 + (7/3)x_3 \leq -3/2 \\
& \ \ x_1 - x_2 + (3/2)x_3 \leq 3/2 \\
& \ \ 2x_1 - x_2 + 3x_3 \leq 7/2 \\
& -x_1 + 4x_2 + 2x_3 \leq 7/2 \\
& -x_1 + 4x_2 \leq 2 \\
& \ \ x_1, x_2, x_3 \in [0,1] \\
& \ \ x_1 \quad \text{integer}
\end{aligned}
$$

Let P denote the feasible region of the linear relaxation of the above integer program. Figure D.1 shows the feasible region of P. Point q^1 denotes the optimal solution to the linear programming relaxation. The vertices of P are:

$$
\begin{aligned}
q^1 &= \{1/2, 1/2, 1\} \\
q^2 &= \{1, 3/4, 3/4\} \\
q^3 &= \{1, 3/4, 1/2\} \\
q^4 &= \{1, 1/2, 2/3\} \\
q^5 &= \{1, 1/4, 1/2\} \\
q^6 &= \{0, 1/2, 3/4\} \\
q^7 &= \{0, 2/5, 1/2\}.
\end{aligned}
$$

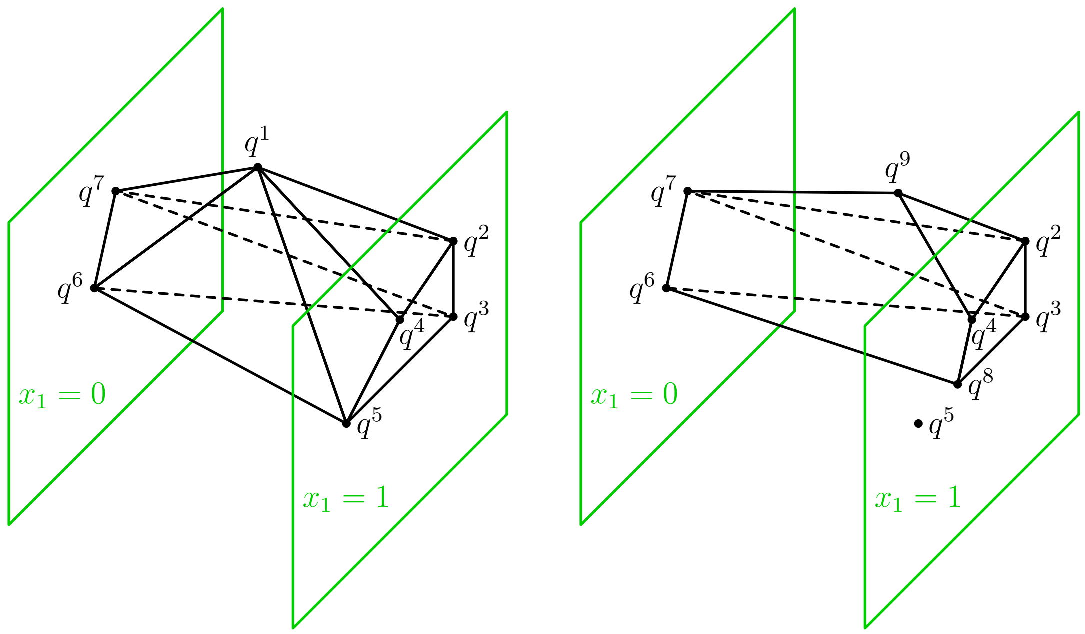


Figure D.1: The LP polytope P for the counter-example showing an optimal point on each disjunctive term and its neighbors as the point-ray collection may lead to invalid cuts.

We use as the valid disjunction the elementary split on x_1. If we solve $\max\{x_3 : x \in P,\ x_1 = 0\}$, the maximum is achieved by q^6. Solving $\max\{x_3 : x \in P,\ x_1 = 1\}$, the maximum is achieved by q^2. Consider using q^2 and q^6 and their neighbors as the collection of points given to (PRLP). Thus $\mathcal{P} = \{q^2, q^3, q^4, q^6, q^7\}$ and $\mathcal{R} = \emptyset$.

One cut that can be obtained from this set of points is $\bar{\alpha}^\mathsf{T} x \geq \bar{\beta}$ where $\bar{\alpha} = (-1/6, 5, -2)$ and $\bar{\beta} = 1$, which goes through q^6, q^7, and q^4, leaving q^2 and q^3 on the feasible side, and cutting off not only q^1, but also q^5. Indeed, $\bar{\alpha}^\mathsf{T} q^1 = -1/12 + 5/2 - 2 = 5/12 < 1$, while $\bar{\alpha}^\mathsf{T} q^4 = \bar{\alpha}^\mathsf{T} q^7 = \bar{\alpha}^\mathsf{T} q^6 = 1$, and $\bar{\alpha}^\mathsf{T} q^2 = 25/12 > 1$ and $\bar{\alpha}^\mathsf{T} q^3 = 31/12 > 1$. However, as seen in Fig. D.1, $\bar{\alpha}^\mathsf{T} q^5 = 1/12 < 1$, so that point of $P \cap \{x : x_1 = 1\}$ is cut off, making the cut invalid.

There is a relatively simple resolution for the above counterexample. If we require the generated cuts to be tight at the optimal solutions on each facet, q^6 and q^2, then all generated cuts will be valid for $\operatorname{conv}(P_D)$. We state this in Theorem D.1.

Theorem D.1. *For each $t \in \mathcal{T}$, set $\mathcal{P}^t$ to be an optimal solution p^t to $\arg\min_x\{c^\mathsf{T} x : x \in P^t\}$ as well as one point on each of the edges emanating from p^t within P^t. Let $\mathcal{P} := \cup_{t \in \mathcal{T}} \mathcal{P}^t$ and $\mathcal{R} := \emptyset$. Any feasible solution to* (PRLP) *formulated from these points and amended with the condition $\alpha^\mathsf{T} p^t = \beta$ for all $t \in \mathcal{T}$ yields a valid cut for P_I.*

Proof. Let (α, β) be a feasible solution to (PRLP) such that $\alpha^\mathsf{T} p^t = \beta$ for all $t \in \mathcal{T}$. Since $\alpha^\mathsf{T} p \geq \beta$ for all $p \in \mathcal{P}^t$, it holds that $\alpha^\mathsf{T}(p - p^t) \geq 0$. This means that, for each $t \in \mathcal{T}$, the generated cut is valid for the cone with apex at p^t and rays going through each of the points $p \in \mathcal{P}^t$. By convexity, this cone is a relaxation of P^t. It follows, by Corollary 4.6, that the cut is valid for P_I. □

D.2 Sample partial branch-and-bound tree

In the chapter, we use a partial branch-and-bound tree as the source of the disjunction for cut generation. The partial trees are produced from the branching strategy described in Section 4.3. In particular, there may exist nodes of the tree that are pruned, and the tree may be very asymmetric. We illustrate this with one sample tree, shown in Figure D.2, constructed from the instance `bm23` and terminated after finding 64 leaf nodes. This tree includes two pruned nodes (the leftmost leaf node, from branching on x_{15}, and an adjacent leaf on the same level, from branching on x_8).

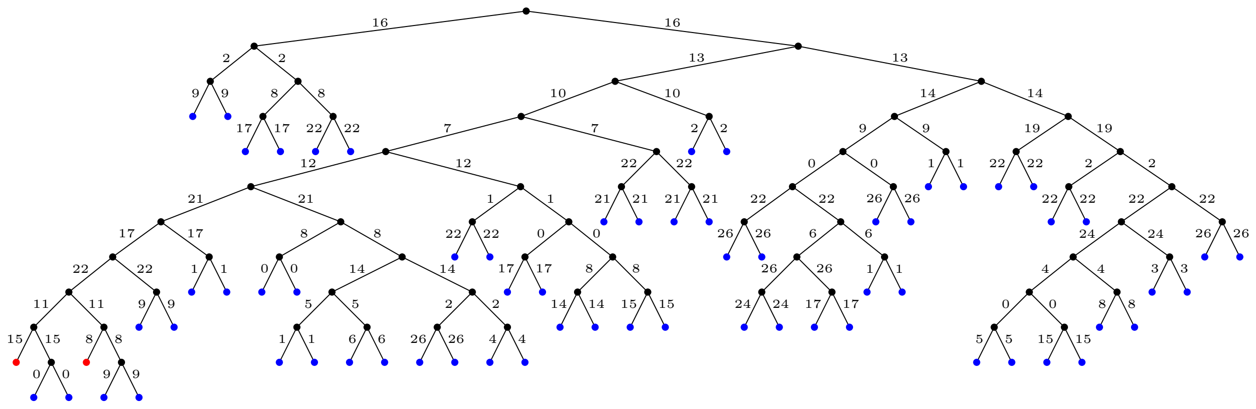


Figure D.2: Partial branch-and-bound tree with 64 leaf nodes for instance `bm23`.

D.3 Parameters to consider for VPC implementations

There are many choices that could be influential in an implementation of VPCs. In this section we list, mostly without discussion, some of the more important parameters.

1. Which disjunction(s) to use? Some options to consider are:
   (a) Split disjunctions: $(x_j \le \lfloor \bar{x}_j \rfloor) \vee (x_j \ge \lceil \bar{x}_j \rceil)$ for all $j \in I$ with $\bar{x}_j \notin \mathbb{Z}$.
   (b) Disjunctions based on the integrality of two integer variables fractional in $\bar{x}$.
   (c) Disjunctions with more terms, such as from the leaf nodes of a partial branch-and-bound tree. This raises the questions of:
      i. which strategy to use to generate the partial tree (how to choose the next subproblem, number of strong branching candidates to test, variable to branch on, and direction to branch in),
      ii. how large the partial tree should be, and
      iii. how else we can use the structural information gained while constructing the partial tree.
2. How to choose the $\mathcal{V}$-polyhedral relaxation? We showed that using the type 1 point-ray collection may significantly weaken the possible cuts that can be generated. On the

other hand, using a more involved relaxation may be computationally expensive, not only when choosing and constructing the proper refinement, but also by increasing the size of the resulting *point-ray linear program* (PRLP), since this LP is solved many times.

3. Which cutting planes to generate? A few of the relevant characteristics are:
   (a) Number of cuts.
   (b) Number of rounds of cuts.
   (c) Minimum orthogonality of the cuts (with respect to each other).
   (d) Density of the cuts.
   (e) Strengthening of the cuts (or disjunctions).
   (f) Requirement that $\bar{x}$ is cut away.

4. Which objectives to try in (PRLP)? The two perspectives are finding cuts that have maximum violation with respect to some point not in P_D, or cuts that have minimum violation with respect to some point in P_D. As described earlier, the three objectives we use are as follows.
   (a) All-ones objective, $w = e$.
   (b) The optimum after adding *Gomory mixed-integer cuts* (GMICs) to P.
   (c) The point determining the disjunctive lower-bound. First, we use $w = \underline{p}$, where $\underline{p} \in \arg\min\{c^\mathsf{T} p : p \in \mathcal{P}\}$. Then, use $w = r$ for all rays $r \in \mathcal{R}$ that were not tight for the cut from the objective $w = \underline{p}$.

5. How to set up the branch-and-bound experiments? There is a trade-off between maintaining an experimental setting that closely resembles reality and controlling for the vast amount of variability in optimization solvers that can overshadow any of the computational results.

D.4 Discarded instances

Of the instances with at most 5,000 rows and columns, Table D.1 lists the 75 that were removed from consideration and the reason for removal. As the table shows, one of the most common reasons for discarding an instance was that no cuts were generated for that instance due either to $\max_{p \in \mathcal{P}} c^\mathsf{T} p = c^\mathsf{T} \bar{x}$ (the optimal value over the best leaf node equals the optimal value of the LP) or (PRLP) being primal infeasible. The situations are treated together because the former anyway typically results in (PRLP) being infeasible, though it also implies that our metric of gap closed is not reasonable for that instance. There is potential for these instances to either generate inequalities that do not cut away $\bar{x}$ or to try to find a different partial branch-and-bound tree, but that has been beyond the scope of our investigation.

Table D.1: Instances that were not considered with reason for being discarded.

Instance	Set	Reason
22433	coral	Integer-optimal solution found during partial tree construction
air01	miplib2	Integer-optimal solution found during partial tree construction
air02	miplib2	Integer-optimal solution found during partial tree construction
air03	miplib3	Integer-optimal solution found during partial tree construction
air06	miplib2	Integer-optimal solution found during partial tree construction
berlin_5_8_0	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
bnatt400	miplib2010	Max leaf value = LP optimal value
csched008	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
egout	miplib3	Integer-optimal solution found during partial tree construction
enlight13	miplib2010	Solved by presolve
enlight15	miplib2010	Solved by presolve
fixnet3	miplib2	Solved at root by Gurobi
fixnet4	miplib2	Solved at root by Gurobi
flugpl	miplib3	Optimal solution to presolved LP is integer-feasible
gen	miplib3	Solved at root by Gurobi
ic97_potential	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
lp4l	miplib2	Integer-optimal solution found during partial tree construction
markshare1	miplib2003	Max leaf value = LP optimal value
markshare2	miplib2003	Max leaf value = LP optimal value
markshare_5_0	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
mcf2	coral	Same as danoint
misc01	miplib2	Integer-optimal solution found during partial tree construction
misc02	miplib2	Integer-optimal solution found during partial tree construction
misc04	mipib2	Integer-optimal solution found during partial tree construction
misc05	miplib2	Integer-optimal solution found during partial tree construction
mitre	miplib3	Presolved instance has no integrality gap
mkc1	coral	Max leaf value = LP optimal value or PRLP primal infeasible
neos-1200887	coral	Max leaf value = LP optimal value or PRLP primal infeasible
neos-1211578	coral	Max leaf value = LP optimal value or PRLP primal infeasible
neos-1228986	coral	Max leaf value = LP optimal value or PRLP primal infeasible
neos-1337489	coral	Max leaf value = LP optimal value or PRLP primal infeasible
neos-1425699	coral	Integer-optimal solution found during partial tree construction
neos-1426635	miplib2010	Max leaf value = LP optimal value
neos-1426662	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
neos-1436709	miplib2010	Max leaf value = LP optimal value
neos-1437164	coral	Max leaf value = LP optimal value or numerical trouble
neos-1440447	coral	Max leaf value = LP optimal value or PRLP primal infeasible
neos-1440460	miplib2010	Max leaf value = LP optimal value
neos-1441553	coral	Max leaf value = LP optimal value or PRLP primal infeasible
neos-1442119	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
neos-1442657	miplib2010	Max leaf value = LP optimal value
neos-1516309	coral	Integer-optimal solution found during partial tree construction
neos-1605075	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
neos-1620770	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible

Instance	Set	Reason
neos-430149	coral	Max leaf value = LP optimal value or PRLP primal infeasible
neos-530627	coral	Solved at root by Gurobi
neos-555298	coral	Presolved instance has no integrality gap
neos-555424	miplib2010	Presolved instance has no integrality gap
neos-555694	coral	Max leaf value = LP optimal value or PRLP primal infeasible
neos-555927	coral	Max leaf value = LP optimal value
neos-631694	coral	Max leaf value = LP optimal value or PRLP primal infeasible
neos-847302	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
neos-955215	coral	Max leaf value = LP optimal value or PRLP primal infeasible
noswot	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
ns1208400	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
opt1217	miplib2003	Max leaf value = LP optimal value or PRLP primal infeasible
p0033	miplib3	Integer-optimal solution found during partial tree construction
p0040	miplib2	Optimal solution to presolved LP is integer-feasible
p0201	miplib3	Integer-optimal solution found during partial tree construction
p0291	miplib2	Integer-optimal solution found during partial tree construction
pigeon-10	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
pigeon-11	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
pigeon-12	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
pigeon-13	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
pigeon-19	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
pk1	miplib2003	Max leaf value = LP optimal value or PRLP primal infeasible
pw-myciel4	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
rlp2	neos	Integer-optimal solution found during partial tree construction
sample2	miplib2	Integer-optimal solution found during partial tree construction
set1cl	miplib2	Solved at root by Gurobi
stein09_nocard	miplib2	Integer-optimal solution found during partial tree construction
stein15_nocard	miplib2	Integer-optimal solution found during partial tree construction
usAbbrv-8-25_70	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible
vpm1	miplib3	Solved at root by Gurobi
wachplan	miplib2010	Max leaf value = LP optimal value or PRLP primal infeasible

D.5 Tables for partial branch-and-bound tree experiments

This section contains full tables left out of the main text due to length.

Table D.2 shows the number of rows and columns for each of the 184 instances with $c^{\mathsf{T}}\underline{p} > c^{\mathsf{T}}\bar{x}$, the number of cuts produced to yield the "best" VPC objective value (across partial tree sizes), and the percent gap closed for each instance. Columns 2 and 3 give, for each instance, the number of constraints and variables after preprocessing. The next two columns show the number of cuts generated. Column 6 is the percent gap closed by GMICs when they are added to the LP relaxation. Column 7 is the percent gap closed as implied by the disjunctive lower bound from the partial tree with 64 leaf nodes. Column 8 is the percent gap closed by VPCs; the subsequent column is the percent gap closed when GMICs and VPCs are used together. Columns 10 and 11 show the percent gap closed by `Gurobi` cuts from one round at the root, first without and then with VPCs added as user cuts. Columns 12 and 13 show the same, but after the last round of cuts at the root. For columns 10 through 13, the values are the maximum percent gap closed across seven random seeds. The last two rows of Table D.2 reproduce the summary data in Table 4.1.

Table D.2: Percent gap closed by instance for GMICs (G), VPCs (V), both VPCs and GMICs used together, and the bound implied by the partial branch-and-bound tree with 64 leaf nodes (DB). Also shown are the sizes of the instances, the number of cuts added, and the percent gap closed by `Gurobi` at the root (after one round (GurF) and after the last round (GurL)). Entries in which DB appears to be 0.00 are actually small strictly positive numbers.

			# cuts		% gap closed							
Instance	Rows	Cols	G	V	G	DB	V	V+G	GurF	V+GurF	GurL	V+GurL
10teams	210	1600	158	2	100.00	85.71	28.57	100.00	100.00	100.00	100.00	100.00
23588	137	237	75	75	5.77	72.18	71.83	71.83	12.13	12.13	21.39	71.60
30n20b8	387	4197	188	98	11.42	1.64	0.04	11.42	1.10	1.10	10.78	10.78
50v-10	233	2013	29	29	45.75	18.01	6.92	46.01	43.40	42.86	66.19	57.32
a1c1s1	2141	2489	155	7	24.27	4.86	1.51	24.27	42.17	44.43	89.16	88.89
aflow30a	449	812	25	25	16.54	17.17	15.93	19.19	24.75	27.05	63.24	62.72
aflow40b	1401	2687	36	36	12.00	14.59	13.33	15.65	22.87	25.40	57.04	57.22
aligninq	337	1831	182	58	11.59	64.30	39.52	40.38	23.41	44.49	48.51	53.66
arki001	693	957	63	12	35.17	13.92	6.24	35.30	27.90	28.01	40.93	40.73
b2c1s1	2546	2677	246	8	16.45	1.51	0.13	16.45	12.85	17.10	70.71	71.23
bc1	1338	1044	5	5	0.59	15.50	14.23	14.23	30.60	30.94	41.06	40.36
beasleyC3	1153	1704	148	148	14.47	2.81	1.04	14.54	48.28	51.31	97.33	97.33
bell3a	63	82	7	4	36.98	53.83	40.67	43.65	38.39	45.54	45.54	45.54
bell3b	73	91	24	21	31.17	86.10	83.93	84.65	32.61	86.41	44.29	95.03
bell4	73	88	27	7	25.62	21.72	14.22	26.39	19.47	23.48	28.91	28.91
bell5	34	56	10	10	13.84	84.75	77.17	77.78	13.58	30.08	22.77	79.65
bg512142	898	757	204	9	0.81	1.82	1.03	1.69	3.45	3.62	10.84	11.12
bienst1	520	449	26	8	11.95	43.61	43.61	43.61	9.65	43.61	12.07	43.61
bienst2	520	449	33	8	9.77	35.63	35.63	35.63	9.89	35.63	10.56	35.63
binkar10_1	815	1423	38	24	13.01	9.96	9.63	14.24	7.62	35.48	72.72	76.47
blend2	154	302	13	13	5.46	29.37	22.80	23.04	3.21	15.26	21.86	26.77
bm23	20	27	6	6	16.81	71.47	70.92	70.92	19.85	70.25	35.93	71.22
cap6000	1725	4596	2	2	42.44	69.28	64.70	65.75	0.54	0.54	44.61	66.06
cov1075	637	120	120	1	1.47	10.49	10.49	10.49	30.00	30.00	30.00	30.00
csched007	274	1680	136	59	5.17	7.71	4.69	7.44	18.06	18.15	36.58	37.42
csched010	272	1678	124	124	4.17	6.05	3.07	4.86	13.48	13.55	36.37	36.37
d10200	898	689	152	37	0.60	14.48	10.54	10.54	0.00	0.00	0.00	0.00
danoint	656	513	34	10	0.26	3.50	1.68	1.77	1.04	2.03	3.28	3.28
dcmulti	271	529	46	46	38.90	27.04	13.84	42.04	40.20	52.25	74.19	77.62
dfn-gwin-UUM	156	936	45	45	45.85	33.45	29.03	47.09	34.53	32.86	61.26	58.85
dg012142	1987	1899	397	16	0.48	0.09	0.05	0.49	0.66	0.66	1.24	1.24
eilB101	100	2815	89	89	2.47	13.46	0.50	2.59	9.69	11.98	41.71	42.74
eild76	75	1893	63	63	1.09	8.84	0.56	1.13	7.34	9.73	54.62	54.62
fiber	267	998	42	42	68.21	10.19	9.38	68.21	46.44	53.63	91.31	91.57
fixnet6	477	877	12	12	24.15	67.53	33.07	45.73	41.23	49.70	73.81	84.01
g200x740i	940	1480	171	25	3.90	2.78	0.20	4.00	68.96	71.91	88.34	88.12
gesa2	1344	1176	42	10	23.86	36.17	6.67	28.17	53.05	66.24	98.44	98.38

			# cuts		% gap closed							
Instance	Rows	Cols	G	V	G	DB	V	V+G	GurF	V+GurF	GurL	V+GurL
gesa2_o	1200	1176	73	12	27.12	15.71	8.74	31.54	33.18	36.09	89.66	94.04
gesa2-o	1200	1176	73	12	27.12	15.71	8.74	31.54	33.18	36.09	89.66	94.04
gesa3	1296	1080	58	58	7.71	48.28	27.90	30.24	42.61	49.43	68.35	74.61
gesa3_o	1152	1080	100	9	44.14	39.57	18.45	45.39	60.46	63.09	72.67	77.75
glass4	392	317	72	72	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gmu-35-40	356	651	11	11	0.07	0.00	0.00	0.07	0.63	0.63	4.87	4.87
gmu-35-50	358	953	16	16	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00
go19	361	361	357	175	2.00	15.58	13.12	13.12	2.60	12.64	6.69	13.35
gt2	28	173	14	7	87.06	2.90	1.67	87.06	42.27	51.11	100.00	100.00
harp2	92	1013	27	27	17.28	15.51	12.23	18.47	13.70	15.72	35.24	37.28
iis-100-0-cov	3831	100	100	100	1.15	35.60	35.55	35.55	3.80	35.39	9.03	43.24
iis-bupa-cov	4796	337	153	98	0.85	31.11	25.35	25.35	4.37	24.33	6.19	25.46
janos-us-DDM	755	2179	81	81	11.98	3.89	3.41	12.20	23.84	24.07	25.62	25.78
k16x240	256	480	14	14	11.40	23.34	21.48	22.47	66.92	69.71	77.90	78.55
khb05250	100	1299	19	14	74.91	57.82	44.64	76.06	67.08	76.61	99.89	99.89
l152lav	97	1987	52	52	12.57	42.42	9.99	16.11	1.12	7.82	3.36	10.08
lotsize	1920	2985	520	4	13.70	2.14	0.27	13.81	19.66	22.43	83.31	83.32
lrn	4968	4399	278	43	19.85	18.01	0.06	19.86	21.77	21.59	28.35	28.29
lseu	28	79	9	9	5.84	22.79	17.26	19.23	38.92	40.73	62.75	60.67
macrophage	2708	1889	535	13	8.52	3.92	1.97	9.84	71.59	72.06	99.71	99.53
mas074	13	148	12	12	6.67	13.51	13.34	13.36	1.51	13.34	7.64	13.44
mas076	12	148	11	11	6.42	13.30	12.96	12.96	3.66	12.34	13.81	13.97
mas284	68	148	20	4	0.87	34.87	33.81	33.81	0.51	33.75	7.67	33.88
maxgasflow	4288	4526	720	7	22.38	0.00	0.00	22.38	25.77	25.96	33.43	32.60
mc11	1917	3035	363	363	30.66	0.54	0.14	30.66	75.89	78.36	97.70	97.16
mcsched	1853	1495	1259	416	0.03	3.45	0.51	0.51	0.06	0.43	0.38	0.53
mik-250-1-100-1	100	251	100	100	53.52	12.03	7.23	53.52	46.61	52.12	74.14	74.14
mine-90-10	4118	778	346	103	6.25	34.42	31.68	32.01	19.99	35.99	25.28	37.10
misc03	95	138	18	18	0.00	50.69	44.32	44.32	0.00	40.51	0.00	46.46
misc07	211	232	16	16	0.72	6.09	5.49	5.49	0.00	0.40	6.09	6.77
mkc	961	2933	122	20	5.74	5.47	3.66	6.23	4.08	9.87	46.88	46.85
mod008	6	319	6	6	24.37	26.26	24.96	27.75	15.07	20.94	75.67	78.53
mod013	62	96	5	5	5.88	52.81	47.39	47.39	30.61	31.94	74.70	77.21
modglob	286	354	29	29	18.09	25.31	3.95	19.58	49.77	55.14	95.37	97.35
n4-3	884	2950	39	39	8.18	23.31	12.39	14.86	18.21	18.21	61.98	64.11
neos-1058477	966	769	28	19	0.75	25.42	6.69	6.69	0.05	6.45	99.21	99.21
neos-1112782	2070	4050	44	6	23.10	66.36	2.58	24.00	13.75	16.31	38.04	38.04
neos-1112787	1640	3200	39	6	23.33	74.82	3.18	24.34	14.16	17.32	38.22	38.22
neos-1215259	1179	1494	145	51	3.46	33.22	22.47	22.47	10.92	25.46	36.80	40.06
neos-1225589	650	1250	25	12	3.94	8.92	4.72	5.51	85.07	85.15	98.95	98.96
neos-1281048	459	685	127	10	0.00	7.18	7.16	7.16	66.23	66.23	99.34	99.34
neos-1396125	1437	1158	63	63	0.00	2.73	0.09	0.10	3.13	3.13	7.40	7.40
neos-1413153	2500	2451	269	1	14.37	26.85	1.48	14.40	15.51	15.51	32.82	32.87
neos-1415183	2809	2757	253	253	11.13	29.29	3.90	11.26	24.40	24.40	29.08	29.08

			# cuts		% gap closed							
Instance	Rows	Cols	G	V	G	DB	V	V+G	GurF	V+GurF	GurL	V+GurL
neos-1420205	341	231	44	1	0.00	14.67	14.67	14.67	20.00	20.00	20.00	20.00
neos-1489999	998	484	484	106	0.41	36.51	17.92	18.06	0.48	14.19	1.66	18.24
neos-1582420	2487	2407	293	91	2.30	31.61	14.77	14.77	12.77	12.77	58.44	58.44
neos-1595230	677	490	120	-	2.40	10.00	0.00	2.40	0.00	0.00	0.00	0.00
neos-1601936	3088	4022	628	-	100.00	100.00	0.00	100.00	100.00	100.00	100.00	100.00
neos-1605061	3447	4023	773	-	39.04	34.19	0.00	39.04	38.37	38.37	91.45	91.45
neos-1616732	1026	200	200	117	1.47	25.79	25.76	25.76	1.85	25.74	7.33	25.74
neos-1620807	405	231	62	1	0.00	16.67	16.67	16.67	0.00	16.67	0.00	16.67
neos-480878	1141	622	23	23	1.84	21.61	14.16	14.69	1.59	11.04	7.80	16.77
neos-501453	13	52	1	1	70.00	0.08	0.08	70.00	70.00	70.00	72.12	72.12
neos-504674	1128	634	148	148	7.09	12.11	5.21	7.97	10.07	10.47	17.19	18.44
neos-504815	887	500	116	116	4.53	18.55	12.92	13.33	10.59	11.95	15.88	18.60
neos-512201	1108	627	143	56	5.68	18.12	12.73	13.87	15.98	18.02	25.01	25.41
neos-522351	1705	1524	106	106	52.74	29.37	29.37	52.75	79.77	79.77	99.22	99.51
neos-538867	1042	666	47	47	0.00	4.60	2.32	2.32	0.00	10.60	0.00	11.11
neos-538916	1154	704	53	53	0.00	4.66	4.49	4.49	0.00	6.32	0.00	10.87
neos-547911	357	2352	86	-	0.91	11.57	0.00	0.91	1.65	1.65	10.88	10.88
neos-570431	925	495	217	54	1.48	43.75	43.38	43.38	9.48	43.42	33.17	43.52
neos-584851	661	441	274	-	0.56	7.64	0.00	0.56	2.33	2.33	7.91	8.55
neos-593853	1202	2030	9	9	20.88	49.15	41.97	43.81	24.41	39.11	35.64	45.35
neos-598183	452	916	19	19	19.39	24.89	16.62	22.46	4.26	14.74	95.70	98.85
neos-603073	452	974	20	20	4.35	4.23	3.29	5.52	1.78	3.33	31.29	31.61
neos-686190	3658	3660	95	80	4.38	62.51	23.45	23.45	4.56	4.56	11.17	24.58
neos-717614	811	3049	145	15	66.87	3.98	2.07	67.65	72.36	87.24	97.87	98.79
neos-775946	1543	2880	159	19	36.93	33.39	31.88	36.93	36.93	34.40	61.65	52.62
neos-801834	3300	3260	481	54	2.95	62.23	17.46	17.63	1.77	1.77	10.18	20.84
neos-803219	621	360	20	20	13.00	65.81	59.02	59.68	18.57	55.03	19.26	59.13
neos-803220	611	350	20	20	14.88	55.68	54.34	54.44	21.92	54.01	22.79	54.55
neos-806323	1070	689	159	31	25.77	0.43	0.43	25.77	21.52	21.62	26.86	36.33
neos-807639	1037	616	116	116	19.99	78.81	72.34	72.52	19.99	68.57	44.43	72.78
neos-807705	1073	643	142	142	10.62	35.75	32.77	33.89	18.13	31.70	20.67	38.75
neos-810326	1730	1701	198	166	3.80	64.03	21.27	21.27	18.66	18.66	28.33	29.51
neos-825075	328	800	148	6	0.00	11.11	11.11	11.11	0.00	10.71	0.00	50.00
neos-839859	2626	1950	144	16	5.10	33.12	10.92	11.93	1.71	11.39	9.59	14.35
neos-860300	568	1232	95	95	1.54	37.62	9.59	9.59	13.97	15.91	56.83	57.48
neos-862348	953	1833	121	30	53.29	4.42	2.01	53.29	54.22	55.66	70.34	62.57
neos-880324	182	135	45	11	0.00	36.33	36.33	36.33	14.42	36.33	24.04	36.33
neos-886822	1057	1025	376	158	3.18	3.16	0.09	3.22	2.04	2.06	2.75	2.75
neos-892255	1675	1521	186	-	0.00	33.33	0.00	0.00	33.33	33.33	33.33	33.33
neos-906865	1562	1160	34	34	17.47	39.37	29.73	30.71	15.50	28.14	27.37	30.31
neos-916792	1413	1465	89	20	4.06	15.72	9.37	9.67	0.43	9.42	6.15	10.90
neos-942830	589	831	262	1	6.25	25.00	12.50	12.50	25.00	25.00	50.00	50.00
neos14	437	676	130	14	55.47	7.14	4.29	56.00	55.55	55.84	80.17	81.18
neos15	460	699	153	16	48.80	6.30	4.23	49.26	48.87	49.12	75.31	78.37

			# cuts		% gap closed							
Instance	Rows	Cols	G	V	G	DB	V	V+G	GurF	V+GurF	GurL	V+GurL
neos17	486	511	171	171	4.19	1.16	0.47	4.58	70.31	70.60	87.80	87.26
neos18	3058	760	532	4	11.30	2.61	2.61	13.91	54.35	54.35	62.94	61.26
neos3	1146	2160	28	28	2.49	20.37	11.83	13.33	3.07	14.17	18.16	25.92
neos5	63	63	35	1	11.11	37.50	37.50	37.50	4.17	37.50	20.85	37.50
neos7	1698	1390	22	13	29.12	30.06	19.44	29.12	35.06	35.06	46.56	46.56
newdano	520	449	52	2	7.76	28.32	28.32	28.32	7.92	28.32	10.60	28.32
ns1606230	3459	3261	966	-	4.11	1.70	0.00	4.11	16.76	16.76	58.55	58.55
ns1688347	2352	1242	317	5	10.38	29.56	16.98	16.98	27.04	27.32	39.62	39.62
ns1830653	1309	585	237	113	9.57	26.28	9.90	11.60	19.32	19.32	28.84	28.84
ns894788	923	2012	376	-	100.00	9.04	0.00	100.00	100.00	100.00	100.00	100.00
nsrand-ipx	510	3713	69	9	14.03	13.75	5.41	15.69	11.10	15.21	61.52	60.55
p0282	160	200	24	24	3.19	13.47	9.84	10.58	74.91	75.93	97.94	99.63
p0548	117	365	31	4	48.62	10.78	7.42	48.93	76.86	79.01	99.86	99.86
p100x588b	688	1176	74	74	10.57	5.43	2.23	11.91	75.33	79.67	91.18	90.36
p2756	591	2143	96	85	93.51	11.61	4.81	93.51	78.76	79.38	98.25	98.58
p6000	1725	4596	2	2	41.65	67.98	63.49	64.52	0.53	0.53	43.78	64.82
p6b	502	451	375	49	0.91	12.92	10.34	10.34	0.64	9.93	1.10	10.40
p80x400b	474	792	52	5	12.27	7.70	2.34	12.75	76.04	78.27	89.49	90.63
pg	125	2700	92	3	24.30	9.18	1.41	24.87	82.28	82.28	82.65	82.65
pg5_34	225	2600	88	20	23.45	6.70	0.66	23.93	94.67	95.35	98.80	98.80
pipex	25	48	6	6	35.56	33.47	33.27	36.66	14.11	31.80	96.90	98.68
pp08a	133	234	53	4	54.53	22.35	9.90	54.53	60.60	59.91	95.38	95.50
pp08aCUTS	239	235	46	46	33.79	21.04	12.71	35.72	49.38	54.18	91.78	91.20
probportfolio	302	320	105	48	4.61	18.70	9.89	11.30	1.98	2.86	14.12	16.31
prod1	75	117	40	40	4.66	40.54	36.12	36.30	9.96	32.51	25.36	37.72
protfold	2110	1835	570	-	4.35	6.56	0.00	4.35	5.83	5.83	10.78	12.89
qiu	1192	840	36	36	0.65	36.61	17.63	18.02	1.57	16.52	6.43	17.74
qnet1	360	1417	47	47	12.61	23.89	8.45	17.11	26.14	30.71	86.21	87.05
qnet1_o	237	1314	10	10	25.04	63.67	61.48	62.68	46.87	63.66	93.45	84.56
queens-30	900	900	900	1	2.29	2.98	0.17	2.31	0.27	0.48	2.24	2.24
r80x800	880	1600	65	65	31.62	20.13	2.01	32.41	67.79	72.71	87.68	86.94
ran14x18	284	504	18	18	7.75	10.58	10.23	10.48	22.86	24.14	52.15	52.09
ran14x18-disj-8	447	504	84	24	0.14	10.29	8.19	8.19	0.74	1.64	6.29	9.22
ran16x16	288	512	20	20	10.17	20.19	19.12	19.59	35.22	37.05	62.42	61.20
reblock67	1928	585	442	72	10.68	31.20	20.55	22.24	15.80	21.43	35.09	39.21
rgn	24	180	18	3	8.03	64.07	42.44	45.39	35.33	49.51	57.49	67.22
rococoB10-011000	720	3510	261	98	9.68	1.10	0.79	9.68	22.30	22.30	31.84	31.95
rococoC10-001000	576	2442	128	128	29.16	30.83	20.85	32.35	35.59	35.71	64.66	66.55
roll3000	1036	897	185	57	5.13	1.34	0.44	5.25	25.82	29.00	89.29	89.33
rout	290	555	40	40	1.47	17.07	16.66	17.02	2.79	16.05	6.36	16.66
roy	147	139	11	11	4.53	43.14	29.82	30.30	25.57	37.61	80.64	86.89
sentoy	30	60	8	8	19.31	61.36	59.51	59.79	9.02	12.10	35.30	59.96
set1al	432	652	197	5	98.85	3.66	1.86	98.85	99.15	99.15	99.95	99.95
set1ch	423	643	129	129	29.41	2.19	1.92	29.41	67.21	67.21	99.84	99.75

			# cuts		% gap closed							
Instance	Rows	Cols	G	V	G	DB	V	V+G	GurF	V+GurF	GurL	V+GurL
set3-10	2481	2677	176	5	17.62	3.77	0.00	18.81	8.63	8.79	39.55	39.69
set3-15	2537	2677	174	9	7.36	1.77	0.16	7.37	6.98	7.51	29.53	29.62
set3-20	2537	2677	176	8	10.55	1.09	0.00	11.04	8.15	8.59	33.50	33.31
seymour	4369	893	472	7	3.84	15.92	4.17	5.26	20.55	20.67	44.19	43.98
seymour-disj-10	4770	1022	604	103	0.20	5.42	0.50	0.51	0.27	0.28	0.49	0.80
sp98ir	1395	1576	102	99	3.69	31.37	11.54	12.04	12.74	12.74	50.19	50.65
stein27_nocard	117	27	27	1	8.30	59.26	59.26	59.26	10.05	66.67	33.33	66.67
stein45_nocard	330	45	45	1	4.69	53.48	53.48	53.48	7.62	59.09	22.80	59.09
timtab1	165	365	128	4	24.08	11.49	4.48	24.08	30.31	29.79	62.70	59.11
timtab2	285	625	214	8	13.66	9.25	2.61	13.66	22.81	23.17	42.75	47.53
toll-like	4038	2570	551	11	3.86	2.61	2.13	5.39	57.23	58.20	91.52	91.50
tr12-30	710	1028	321	4	58.36	1.74	0.78	58.60	60.22	60.39	99.58	99.65
uct-subprob	1725	2062	827	5	2.23	10.48	6.30	6.88	10.60	11.75	33.27	33.82
umts	1749	1648	281	75	0.97	0.17	0.07	0.97	1.69	1.69	5.01	5.01
vpm2	127	187	24	24	18.20	13.62	11.24	20.06	43.29	48.35	72.77	71.44
Average					17.28	24.03	15.60	26.95	25.99	33.03	46.48	52.07
Wins						114	91	156		143		116

Table D.3: Time (in seconds) and number nodes taken to solve each instance. The table is sorted by column 4 (“V” under “Time (s)”). “Gur1” indicates `Gurobi` run with one random seed. “Gur7” indicates the minimum from seven runs of `Gurobi` with different random seeds.

	Time (s)					Nodes		
Instance	Gur1	Gur7	V	V+PRLP	V Total	Gur1	Gur7	V
neos-796608	0.01	0.01	0.01	0.01	0.01	22	20	22
neos-501453	0.02	0.02	0.02	0.02	0.02	176	176	176
gt2	0.02	0.02	0.02	0.02	0.02	2	2	2
roy	0.03	0.03	0.03	0.03	0.03	18	18	1
pipex	0.03	0.03	0.03	0.03	0.03	4	4	3
p0548	0.04	0.04	0.04	0.04	0.04	2	2	2
bm23	0.09	0.09	0.04	0.09	0.09	391	391	167
mod013	0.06	0.06	0.04	0.06	0.06	61	61	43
set1al	0.05	0.04	0.05	0.05	0.05	8	8	8
lseu	0.09	0.08	0.09	0.09	0.09	227	227	227
sentoy	0.19	0.18	0.10	0.19	0.19	127	127	127
modglob	0.13	0.13	0.11	0.13	0.13	55	55	10
mod008	0.12	0.12	0.12	0.12	0.12	45	45	33
bell5	0.19	0.19	0.12	0.13	0.13	2332	2332	1255
khb05250	0.14	0.14	0.13	0.14	0.14	7	7	6
p0282	0.15	0.14	0.13	0.15	0.15	90	90	3
neos-1599274	0.16	0.14	0.16	0.16	0.16	1	1	1

	Time (s)					Nodes		
Instance	Gur1	Gur7	V	V+PRLP	V Total	Gur1	Gur7	V
bell4	0.39	0.38	0.28	0.36	0.36	2537	2537	1623
gesa2	0.34	0.33	0.31	0.34	0.34	17	17	17
neos-1225589	0.39	0.31	0.31	0.39	0.39	26	26	23
stein27_nocard	0.38	0.37	0.33	0.38	0.38	2703	2701	2703
vpm2	0.34	0.34	0.34	0.34	0.34	457	457	457
qnet1_o	0.42	0.36	0.35	0.42	0.42	88	21	27
set1ch	0.37	0.34	0.35	0.37	0.37	317	317	317
fixnet6	0.35	0.34	0.35	0.35	0.35	42	42	35
neos-1058477	0.37	0.30	0.36	0.37	0.37	10	7	10
dcmulti	0.38	0.37	0.37	0.38	0.38	114	78	57
neos-880324	0.37	0.35	0.37	0.37	0.37	482	431	482
blend2	0.89	0.47	0.39	0.77	0.78	1455	1048	947
fiber	1.07	0.33	0.41	0.54	0.56	584	172	222
neos-1480121	0.41	0.28	0.41	0.41	0.41	1015	816	1015
rgn	0.44	0.36	0.41	0.44	0.44	1440	1350	1440
bell3b	0.59	0.36	0.46	0.47	0.47	4251	2691	2621
gesa3_o	0.63	0.62	0.47	0.63	0.63	66	66	44
misc03	0.83	0.61	0.51	0.57	0.58	1686	1221	863
gesa3	0.64	0.63	0.56	0.64	0.64	40	40	38
neos-522351	1.00	0.89	0.61	1.00	1.00	152	150	34
bell3a	0.93	0.93	0.69	0.72	0.73	8123	8123	5948
neos7	1.58	0.95	0.71	1.58	1.58	684	280	355
p2756	1.45	1.25	0.75	1.45	1.45	461	461	132
neos-1420205	3.79	0.64	0.76	1.08	1.10	5511	1752	1872
pp08a	1.27	0.85	0.79	0.83	0.83	1574	947	873
pp08aCUTS	1.18	1.14	0.80	0.93	0.94	1105	1105	868
neos-598183	0.87	0.87	0.87	0.87	0.87	114	114	33
gesa2_o	1.45	1.44	0.96	1.45	1.45	565	565	555
gesa2-o	1.77	1.67	0.96	1.72	1.76	565	565	555
neos-775946	1.95	1.57	0.99	1.95	1.95	146	41	36
qnet1	1.26	0.88	1.00	1.26	1.26	44	23	26
23588	1.62	1.30	1.41	1.62	1.62	1181	744	961
l152lav	2.35	1.56	1.49	2.35	2.35	631	328	319
lectsched-4-obj	2.48	1.24	1.52	2.48	2.48	53	1	53
macrophage	1.95	1.93	1.60	1.95	1.95	21	21	9
neos-862348	1.78	1.78	1.78	1.78	1.78	313	154	313
neos-1112787	3.04	2.99	1.96	3.04	3.04	801	801	504
neos20	2.06	1.20	1.97	2.06	2.06	497	342	497
neos2	2.40	2.40	2.40	2.40	2.40	624	601	624
neos-555771	4.92	1.87	2.59	4.92	4.92	517	128	212
10teams	19.24	8.72	2.81	19.24	19.24	591	189	31
neos17	2.92	2.38	2.92	2.92	2.92	2305	1877	2305

	Time (s)					Nodes		
Instance	Gur1	Gur7	V	V+PRLP	V Total	Gur1	Gur7	V
neos-825075	4.13	2.56	4.13	4.13	4.13	543	543	543
neos-584851	23.27	3.49	4.31	4.99	5.13	3728	663	688
neos-717614	13.09	11.22	4.62	13.02	13.09	12024	12024	8816
neos-1281048	5.82	5.82	5.44	5.82	5.82	102	69	102
neos-1489999	5.57	5.26	5.57	5.57	5.57	88	73	80
aligninq	5.82	5.82	5.82	5.82	5.82	490	490	490
neos18	5.88	5.15	5.88	5.88	5.88	2191	1819	2191
mas284	8.63	8.11	6.10	8.63	8.63	14405	13942	14405
neos3	7.01	5.46	6.28	7.01	7.01	2097	1653	1665
neos16	7.33	0.68	6.36	6.43	6.48	3635	523	3635
neos-603073	6.76	6.63	6.41	6.76	6.76	770	770	770
eild76	6.47	6.21	6.47	6.47	6.47	152	152	152
neos-512201	12.83	5.58	6.57	12.83	12.83	2254	1439	1602
neos-803219	7.58	6.40	7.58	7.58	7.58	5415	4719	5195
neos-803220	9.50	8.79	8.05	9.50	9.50	8260	7561	7161
neos-504815	8.53	8.53	8.53	8.53	8.53	3459	3459	3138
neos-801834	9.97	9.89	9.33	9.97	9.97	195	182	158
pg	10.44	10.38	9.67	10.44	10.44	1016	1016	1016
neos-806323	25.71	16.41	9.91	12.54	13.15	6714	4041	1696
neos-807639	12.53	9.86	10.46	12.53	12.53	3400	3284	3043
neos-807705	10.64	8.95	10.64	10.64	10.64	2217	1898	2217
neos-1112782	10.78	10.72	10.78	10.78	10.78	1895	1895	1392
neos-593853	110.01	21.46	12.54	14.93	15.15	27195	9343	4694
neos-570431	13.65	13.65	13.65	13.65	13.65	1193	1161	1193
misc07	74.12	29.06	14.10	14.19	14.22	62889	25415	16054
binkar10_1	15.53	15.53	14.92	15.53	15.53	6414	5288	5744
aflow30a	18.96	10.68	15.14	15.33	15.39	4528	2423	2894
p6000	15.84	15.80	15.84	15.84	15.84	3948	3948	3306
cap6000	15.93	15.77	15.93	15.93	15.93	3948	3948	3306
stein45_nocard	16.28	14.31	16.19	16.28	16.28	46555	43499	38857
neos-860300	20.95	20.95	17.78	20.95	20.95	654	654	485
arki001	18.18	8.35	18.18	18.18	18.18	6489	2198	6489
neos-1415183	19.52	5.30	19.52	19.52	19.52	1244	735	900
neos-1413153	19.94	17.85	19.94	19.94	19.94	1157	1143	1149
beasleyC3	28.69	14.95	21.66	23.52	23.87	910	659	615
mc11	30.34	26.08	21.67	30.34	30.34	636	628	590
roll3000	32.71	21.32	27.08	27.83	28.18	2126	1547	1587
rout	28.15	28.15	28.15	28.15	28.15	19067	16967	11779
neos-1582420	28.94	3.86	28.94	28.94	28.94	1047	155	1047
ns1688347	54.90	27.02	30.33	54.90	54.90	2152	1113	1517
bienst1	46.14	34.20	32.46	42.84	42.98	11102	10784	8872
neos-480878	52.07	31.22	35.85	42.75	42.87	7962	6836	5891

	Time (s)					Nodes		
Instance	Gur1	Gur7	V	V+PRLP	V Total	Gur1	Gur7	V
neos-538916	36.13	36.13	36.13	36.13	36.13	19766	19766	19766
neos-906865	38.18	37.23	38.18	38.18	38.18	7315	6707	7315
neos-504674	66.24	38.61	40.45	42.48	43.22	8281	5109	5453
neos-911880	3600.00	19.21	41.53	41.83	42.03	4303328	10741	32373
neos-538867	47.76	47.76	47.76	47.76	47.76	33208	33208	33208
neos-892255	49.08	20.54	49.08	49.08	49.08	680	464	680
neos-1215259	54.28	43.70	49.61	54.28	54.28	1308	979	1107
prod1	62.72	56.37	50.91	55.52	56.99	44431	41529	33157
qiu	62.65	49.55	52.41	62.65	62.65	10065	8814	9216
neos-839859	59.02	43.62	56.20	59.02	59.02	4621	3058	4621
neos-1396125	157.30	28.99	58.39	76.44	77.08	6501	1148	2311
pg5_34	109.63	109.63	59.31	59.84	59.88	15750	15750	8833
mik-250-1-100-1	85.85	64.02	59.81	61.81	61.91	30820	25082	24685
lrn	162.87	42.50	63.75	162.87	162.87	5145	1111	1518
sp98ir	68.95	44.26	68.95	68.95	68.95	1269	1269	1206
bc1	72.98	72.98	72.98	72.98	72.98	2771	2771	2771
neos-810326	159.47	40.99	76.52	101.22	101.87	4005	860	2170
neos-686190	101.64	71.97	80.58	101.64	101.64	10071	5581	6536
ran16x16	189.96	114.96	90.14	91.35	91.51	38847	29498	18434
mas076	131.06	123.68	90.73	90.83	90.84	219321	219321	182305
harp2	216.56	133.74	92.73	93.50	93.54	115912	85376	76767
30n20b8	165.72	133.03	106.97	165.72	165.72	785	716	785
mine-90-10	793.30	269.93	119.46	406.70	413.23	208784	62561	34388
nsrand-ipx	143.16	132.33	123.19	127.88	128.04	6860	5502	4792
bienst2	145.79	117.76	138.51	143.89	144.02	82657	72249	76823
neos-886822	141.96	108.31	141.96	141.96	141.96	12818	10610	12818
neos14	162.75	156.85	157.38	157.42	157.44	58288	58288	58288
timtab1	308.79	235.68	227.72	227.79	227.82	58028	36560	36976
glass4	354.27	211.68	259.66	259.86	259.96	404317	156966	140261
mcsched	278.94	138.59	278.94	278.94	278.94	24029	13421	23045
tr12-30	366.49	283.71	296.44	299.07	299.77	155870	110839	115114
ns1830653	650.26	189.63	297.15	373.44	384.05	14002	7119	9065
eilB101	356.06	337.29	308.44	356.06	356.06	5049	5049	4937
neos-1620807	401.95	401.95	348.95	349.28	349.40	170114	170114	146470
neos-916792	419.24	419.24	414.67	419.24	419.24	80347	75915	77289
ns1606230	437.61	406.49	437.61	437.61	437.61	539	537	539
ns2081729	1164.14	325.98	444.50	446.65	447.71	856594	265549	297466
reblock67	669.56	490.69	454.50	669.56	669.56	149731	94609	102046
neos-1601936	2887.71	129.24	490.70	1349.39	1426.28	10225	457	855
rococoC10-001000	945.54	922.11	665.86	668.82	668.93	55535	51122	55439
neos-1595230	1103.66	719.61	748.57	751.06	751.17	181361	114144	129236
neos5	1239.90	617.23	789.47	790.31	791.26	905747	510755	370106

	Time (s)					Nodes		
Instance	Gur1	Gur7	V	V+PRLP	V Total	Gur1	Gur7	V
neos-547911	858.75	234.31	858.75	858.75	858.75	13946	5279	13946
n4-3	961.28	760.15	961.28	961.28	961.28	16189	13521	15961
aflow40b	1098.90	1098.90	1098.90	1098.90	1098.90	50709	50709	50709
neos-1605061	1209.36	1209.36	1209.36	1209.36	1209.36	6283	6277	6283
dfn-gwin-UUM	1373.28	964.96	1230.43	1239.19	1239.75	276913	209309	208185
umts	2344.82	974.07	1234.36	1337.32	1337.85	866745	398798	471882
k16x240	1493.09	1060.81	1399.93	1400.15	1400.19	517769	307378	397159
mas074	1913.29	1540.80	1428.56	1428.76	1428.79	2704801	2457271	2647865
gmu-35-40	3600.00	575.93	1497.91	1501.60	1502.05	1763647	213513	1204007
newdano	1658.79	1658.79	1658.79	1658.79	1658.79	1583316	1577872	148850
iis-100-0-cov	2286.74	2286.74	2286.74	2286.74	2286.74	113846	113846	113846
csched007	2525.97	2391.25	2525.97	2525.97	2525.97	114745	93447	114745
a1c1s1	2698.94	2698.94	2545.58	2555.06	2556.30	46312	43057	43584
csched010	3600.00	2442.29	3028.19	3040.81	3040.97	95668	67054	85407
p80x400b	3156.20	2182.93	3156.20	3156.20	3156.20	363870	284292	299703
gmu-35-50	3517.04	3517.04	3517.04	3517.04	3517.04	1789962	1096140	1789962
Gmean	81.48	58.94	63.79	68.37	68.50	6069	4475	4549
Wins1		87	89	45	45		105	109
Wins7			41	12	12			64

Table D.4: Time (in seconds) and number nodes taken to solve each of the instances for which all six branch-and-bound trees successfully yielded VPCs. The table is sorted by column 4 (“V” under “Time (s)”). “Gur1” indicates `Gurobi` run with one random seed. “Gur7” indicates the minimum from seven runs of `Gurobi` with different random seeds.

	Time (s)					Nodes		
Instance	Gur1	Gur7	V	V+PRLP	V Total	Gur1	Gur7	V
gt2	0.02	0.02	0.02	0.02	0.02	2	2	2
roy	0.03	0.03	0.03	0.03	0.03	18	18	1
pipex	0.03	0.03	0.03	0.03	0.03	4	4	3
p0548	0.04	0.04	0.04	0.04	0.04	2	2	2
bm23	0.09	0.09	0.04	0.09	0.09	391	391	167
mod013	0.06	0.06	0.04	0.06	0.06	61	61	43
set1al	0.05	0.04	0.05	0.05	0.05	8	8	8
lseu	0.09	0.08	0.09	0.09	0.09	227	227	227
sentoy	0.19	0.18	0.10	0.19	0.19	127	127	127
modglob	0.13	0.13	0.11	0.13	0.13	55	55	10
mod008	0.12	0.12	0.12	0.12	0.12	45	45	33
bell5	0.19	0.19	0.12	0.13	0.13	2332	2332	1255
khb05250	0.14	0.14	0.13	0.14	0.14	7	7	6

	Time (s)					Nodes		
Instance	Gur1	Gur7	V	V+PRLP	V Total	Gur1	Gur7	V
p0282	0.15	0.14	0.13	0.15	0.15	90	90	3
bell4	0.39	0.38	0.28	0.36	0.36	2537	2537	1623
gesa2	0.34	0.33	0.31	0.34	0.34	17	17	17
stein27_nocard	0.38	0.37	0.33	0.38	0.38	2703	2701	2703
vpm2	0.34	0.34	0.34	0.34	0.34	457	457	457
qnet1_o	0.42	0.36	0.35	0.42	0.42	88	21	27
set1ch	0.37	0.34	0.35	0.37	0.37	317	317	317
fixnet6	0.35	0.34	0.35	0.35	0.35	42	42	35
dcmulti	0.38	0.37	0.37	0.38	0.38	114	78	57
blend2	0.89	0.47	0.39	0.77	0.78	1455	1048	947
fiber	1.07	0.33	0.41	0.54	0.56	584	172	222
rgn	0.44	0.36	0.41	0.44	0.44	1440	1350	1440
bell3b	0.59	0.36	0.46	0.47	0.47	4251	2691	2621
gesa3_o	0.63	0.62	0.47	0.63	0.63	66	66	44
misc03	0.83	0.61	0.51	0.57	0.58	1686	1221	863
gesa3	0.64	0.63	0.56	0.64	0.64	40	40	38
neos-522351	1.00	0.89	0.61	1.00	1.00	152	150	34
bell3a	0.93	0.93	0.69	0.72	0.73	8123	8123	5948
p2756	1.45	1.25	0.75	1.45	1.45	461	461	132
pp08a	1.27	0.85	0.79	0.83	0.83	1574	947	873
pp08aCUTS	1.18	1.14	0.80	0.93	0.94	1105	1105	868
neos-598183	0.87	0.87	0.87	0.87	0.87	114	114	33
gesa2_o	1.45	1.44	0.96	1.45	1.45	565	565	555
gesa2-o	1.77	1.67	0.96	1.72	1.76	565	565	555
23588	1.62	1.30	1.41	1.62	1.62	1181	744	961
macrophage	1.95	1.93	1.60	1.95	1.95	21	21	9
neos-862348	1.78	1.78	1.78	1.78	1.78	313	154	313
neos17	2.92	2.38	2.92	2.92	2.92	2305	1877	2305
neos-717614	13.09	11.22	4.62	13.02	13.09	12024	12024	8816
neos-1489999	5.57	5.26	5.57	5.57	5.57	88	73	80
neos18	5.88	5.15	5.88	5.88	5.88	2191	1819	2191
mas284	8.63	8.11	6.10	8.63	8.63	14405	13942	14405
neos3	7.01	5.46	6.28	7.01	7.01	2097	1653	1665
neos-603073	6.76	6.63	6.41	6.76	6.76	770	770	770
neos-512201	12.83	5.58	6.57	12.83	12.83	2254	1439	1602
neos-803219	7.58	6.40	7.58	7.58	7.58	5415	4719	5195
neos-803220	9.50	8.79	8.05	9.50	9.50	8260	7561	7161
neos-504815	8.53	8.53	8.53	8.53	8.53	3459	3459	3138
pg	10.44	10.38	9.67	10.44	10.44	1016	1016	1016
neos-806323	25.71	16.41	9.91	12.54	13.15	6714	4041	1696
neos-807639	12.53	9.86	10.46	12.53	12.53	3400	3284	3043
neos-807705	10.64	8.95	10.64	10.64	10.64	2217	1898	2217

	Time (s)					Nodes		
Instance	Gur1	Gur7	V	V+PRLP	V Total	Gur1	Gur7	V
neos-593853	110.01	21.46	12.54	14.93	15.15	27195	9343	4694
neos-570431	13.65	13.65	13.65	13.65	13.65	1193	1161	1193
misc07	74.12	29.06	14.10	14.19	14.22	62889	25415	16054
binkar10_1	15.53	15.53	14.92	15.53	15.53	6414	5288	5744
aflow30a	18.96	10.68	15.14	15.33	15.39	4528	2423	2894
p6000	15.84	15.80	15.84	15.84	15.84	3948	3948	3306
cap6000	15.93	15.77	15.93	15.93	15.93	3948	3948	3306
stein45_nocard	16.28	14.31	16.19	16.28	16.28	46555	43499	38857
arki001	18.18	8.35	18.18	18.18	18.18	6489	2198	6489
mc11	30.34	26.08	21.67	30.34	30.34	636	628	590
roll3000	32.71	21.32	27.08	27.83	28.18	2126	1547	1587
rout	28.15	28.15	28.15	28.15	28.15	19067	16967	11779
ns1688347	54.90	27.02	30.33	54.90	54.90	2152	1113	1517
bienst1	46.14	34.20	32.46	42.84	42.98	11102	10784	8872
neos-480878	52.07	31.22	35.85	42.75	42.87	7962	6836	5891
neos-504674	66.24	38.61	40.45	42.48	43.22	8281	5109	5453
prod1	62.72	56.37	50.91	55.52	56.99	44431	41529	33157
pg5_34	109.63	109.63	59.31	59.84	59.88	15750	15750	8833
mik-250-1-100-1	85.85	64.02	59.81	61.81	61.91	30820	25082	24685
bc1	72.98	72.98	72.98	72.98	72.98	2771	2771	2771
ran16x16	189.96	114.96	90.14	91.35	91.51	38847	29498	18434
mas076	131.06	123.68	90.73	90.83	90.84	219321	219321	182305
harp2	216.56	133.74	92.73	93.50	93.54	115912	85376	76767
mine-90-10	793.30	269.93	119.46	406.70	413.23	208784	62561	34388
nsrand-ipx	143.16	132.33	123.19	127.88	128.04	6860	5502	4792
bienst2	145.79	117.76	138.51	143.89	144.02	82657	72249	76823
neos14	162.75	156.85	157.38	157.42	157.44	58288	58288	58288
timtab1	308.79	235.68	227.72	227.79	227.82	58028	36560	36976
tr12-30	366.49	283.71	296.44	299.07	299.77	155870	110839	115114
neos-916792	419.24	419.24	414.67	419.24	419.24	80347	75915	77289
rococoC10-001000	945.54	922.11	665.86	668.82	668.93	55535	51122	55439
neos5	1239.90	617.23	789.47	790.31	791.26	905747	510755	370106
aflow40b	1098.90	1098.90	1098.90	1098.90	1098.90	50709	50709	50709
dfn-gwin-UUM	1373.28	964.96	1230.43	1239.19	1239.75	276913	209309	208185
k16x240	1493.09	1060.81	1399.93	1400.15	1400.19	517769	307378	397159
mas074	1913.29	1540.80	1428.56	1428.76	1428.79	2704801	2457271	2647865
gmu-35-40	3600.00	575.93	1497.91	1501.60	1502.05	1763647	213513	1204007
newdano	1658.79	1658.79	1658.79	1658.79	1658.79	1583316	1577872	148850
iis-100-0-cov	2286.74	2286.74	2286.74	2286.74	2286.74	113846	113846	113846
a1c1s1	2698.94	2698.94	2545.58	2555.06	2556.30	46312	43057	43584
p80x400b	3156.20	2182.93	3156.20	3156.20	3156.20	363870	284292	299703

	Time (s)					Nodes		
Instance	Gur1	Gur7	V	V+PRLP	V Total	Gur1	Gur7	V
gmu-35-50	3517.04	3517.04	3517.04	3517.04	3517.04	1789962	1096140	1789962
Gmean	65.56	54.28	54.04	56.59	56.67	6747	5588	5239
Wins1		44	57	30	30		57	73
Wins7			33	11	11			51

Table D.5: Branch-and-bound statistics by number leaf nodes for "6 trees" set of instances. Each set contains the instances for which VPCs were generated and `Gurobi` solves the problem in under an hour either with or without VPCs.

			Time (s)				Nodes	
Set	# inst		Gur1	V	w/PRLP	Total	Gur1	V
2 leaves [0,3600)	97	Gmean	65.56	70.29	70.98	71.11	6747	200625
		Wins1		26	20	20		44
2 leaves [10,3600)	48	Gmean	199.22	216.63	218.85	219.13	32828	33126
		Wins1		14	13	13		21
2 leaves [100,3600)	22	Gmean	785.11	888.62	890.16	890.31	182736	200625
		Wins1		7	7	7		10
2 leaves [1000,3600)	11	Gmean	2016.85	2225.14	2225.53	2225.73	455997	532363
		Wins1		3	3	3		3
4 leaves [0,3600)	97	Gmean	65.56	72.85	75.50	75.96	6747	187036
		Wins1		12	9	9		32
4 leaves [10,3600)	48	Gmean	199.22	228.71	238.11	239.54	32828	37405
		Wins1		8	7	7		14
4 leaves [100,3600)	22	Gmean	762.16	925.84	933.12	933.62	152198	187036
		Wins1		3	3	3		5
4 leaves [1000,3600)	11	Gmean	2016.85	2434.61	2436.20	2436.67	455997	572511
		Wins1		1	1	1		2
8 leaves [0,3600)	97	Gmean	65.56	69.71	77.01	77.94	6747	169522
		Wins1		22	12	12		41
8 leaves [10,3600)	48	Gmean	199.22	214.97	238.04	240.74	32828	33309
		Wins1		11	9	9		20
8 leaves [100,3600)	23	Gmean	728.12	792.90	814.31	815.97	164565	169522
		Wins1		6	6	6		10
8 leaves [1000,3600)	11	Gmean	2016.85	2350.61	2360.07	2361.38	455997	496207
		Wins1		2	2	2		5
16 leaves [0,3600)	97	Gmean	65.56	68.09	91.35	93.24	6747	156831
		Wins1		27	14	13		40
16 leaves [10,3600)	47	Gmean	206.32	215.40	279.14	284.70	34556	30524
		Wins1		12	10	10		20

			Time (s)				Nodes	
Set	# inst		Gur1	V	w/PRLP	Total	Gur1	V
16 leaves [100,3600)	22	Gmean	770.35	838.38	925.39	931.40	175639	156831
		Wins1		5	5	5		9
16 leaves [1000,3600)	10	Gmean	2116.48	2633.10	2662.14	2665.77	425732	374841
		Wins1		2	2	2		4
32 leaves [0,3600)	97	Gmean	65.56	72.79	125.62	129.90	6747	194536
		Wins1		27	8	8		41
32 leaves [10,3600)	46	Gmean	214.11	245.78	393.29	405.24	35721	37933
		Wins1		11	7	7		20
32 leaves [100,3600)	23	Gmean	728.12	890.85	1010.87	1020.74	164565	194536
		Wins1		4	4	4		8
32 leaves [1000,3600)	10	Gmean	2116.48	2422.80	2487.50	2501.11	425732	510264
		Wins1		2	2	2		4
64 leaves [0,3600)	97	Gmean	65.56	71.53	165.41	176.93	6747	145261
		Wins1		32	10	9		48
64 leaves [10,3600)	46	Gmean	213.16	238.46	516.36	553.50	36141	34778
		Wins1		14	9	9		25
64 leaves [100,3600)	23	Gmean	728.12	809.06	1050.44	1078.81	164565	145261
		Wins1		8	7	7		14
64 leaves [1000,3600)	11	Gmean	2016.85	2422.10	2526.42	2555.06	455997	401381
		Wins1		2	2	2		5

Table D.6: Information about objectives and time to generate cuts corresponding to the results in Table D.2.

	Objectives				Time (s)				
Instance	Obj	Succ	Fails	% fails	Tree	Setup	Solve	(s) / obj	(s) / cut
10teams	20	2	18	90.0	2.0	220.9	678.6	33.9	339.3
23588	75	75	0	0.0	1.2	3.4	219.5	2.9	2.9
30n20b8	105	98	7	6.7	1.5	28.6	847.1	8.1	8.6
50v-10	34	29	5	14.7	0.6	0.3	15.6	0.5	0.5
a1c1s1	8	7	1	12.5	1.2	0.1	1.2	0.1	0.2
aflow30a	26	25	1	3.8	1.2	3.3	42.2	1.6	1.7
aflow40b	37	36	1	2.7	5.5	46.8	770.1	20.8	21.4
aligninq	65	58	7	10.8	1.7	8.6	895.2	13.8	15.4
arki001	25	12	13	52.0	0.2	0.0	1.8	0.1	0.2
b2c1s1	29	8	21	72.4	1.6	5.5	119.0	4.1	14.9
bc1	6	5	1	16.7	39.5	10.5	77.0	12.8	15.4
beasleyC3	149	148	1	0.7	1.9	14.7	565.6	3.8	3.8
bell3a	8	4	4	50.0	0.0	0.0	0.0	0.0	0.0
bell3b	50	21	30	60.0	0.0	0.0	0.0	0.0	0.0

	Objectives				Time (s)				
Instance	Obj	Succ	Fails	% fails	Tree	Setup	Solve	(s) / obj	(s) / cut
bell4	25	7	18	72.0	0.0	0.0	0.0	0.0	0.0
bell5	11	10	1	9.1	0.0	0.0	0.0	0.0	0.0
bg512142	114	9	105	92.1	1.1	1.2	518.0	4.5	57.6
bienst1	54	8	46	85.2	2.0	101.9	218.2	4.0	27.3
bienst2	68	8	60	88.2	1.9	106.0	237.9	3.5	29.7
binkar10_1	78	24	54	69.2	1.4	2.2	170.6	2.2	7.1
blend2	15	13	2	13.3	0.3	0.5	3.1	0.2	0.2
bm23	7	6	1	14.3	0.1	0.0	0.1	0.0	0.0
cap6000	4	2	2	50.0	1.1	4.3	32.3	8.1	16.1
cov1075	21	1	20	95.2	12.8	5.1	13.3	0.6	13.3
csched007	80	59	21	26.3	1.1	20.9	881.1	11.0	14.9
csched010	138	124	14	10.1	0.6	4.9	645.1	4.7	5.2
d10200	38	37	1	2.6	11.4	15.9	897.8	23.6	24.3
danoint	46	10	36	78.3	5.2	52.9	850.8	18.5	85.1
dcmulti	50	46	4	8.0	0.4	0.2	7.1	0.1	0.2
dfn-gwin-UUM	46	45	1	2.2	0.4	2.3	40.0	0.9	0.9
dg012142	29	16	13	44.8	10.5	50.6	853.8	29.4	53.4
eilB101	91	89	2	2.2	1.2	10.9	624.1	6.9	7.0
eild76	65	63	2	3.1	0.9	10.6	376.4	5.8	6.0
fiber	49	42	7	14.3	0.5	1.1	12.1	0.2	0.3
fixnet6	13	12	1	7.7	0.6	1.1	5.4	0.4	0.4
g200x740i	26	25	1	3.8	0.3	0.1	3.1	0.1	0.1
gesa2	39	10	29	74.4	0.2	0.0	0.1	0.0	0.0
gesa2_o	38	12	28	73.7	0.1	0.0	0.0	0.0	0.0
gesa2-o	38	12	28	73.7	0.1	0.0	0.1	0.0	0.0
gesa3	97	58	39	40.2	0.7	0.3	14.9	0.2	0.3
gesa3_o	10	9	1	10.0	1.1	0.1	2.9	0.3	0.3
glass4	112	72	40	35.7	0.4	0.4	0.3	0.0	0.0
gmu-35-40	11	11	0	0.0	0.5	0.6	12.2	1.1	1.1
gmu-35-50	17	16	1	5.9	0.7	1.5	25.3	1.5	1.6
go19	181	175	6	3.3	4.1	20.4	879.9	4.9	5.0
gt2	30	7	23	76.7	0.0	0.0	0.1	0.0	0.0
harp2	29	27	2	6.9	0.4	0.9	26.5	0.9	1.0
iis-100-0-cov	102	100	2	2.0	58.6	8.0	160.3	1.6	1.6
iis-bupa-cov	98	98	0	0.0	44.0	23.0	880.9	9.0	9.0
janos-us-DDM	126	81	45	35.7	1.1	0.1	3.5	0.0	0.0
k16x240	16	14	2	12.5	0.2	0.2	2.3	0.1	0.2
khb05250	15	14	1	6.7	0.3	0.1	4.5	0.3	0.3
l152lav	54	52	2	3.7	0.2	0.8	76.0	1.4	1.5
lotsize	6	4	2	33.3	0.3	0.0	0.0	0.0	0.0
lrn	51	43	9	17.6	12.0	32.5	858.5	16.8	20.0
lseu	11	9	2	18.2	0.1	0.0	0.2	0.0	0.0

	Objectives				Time (s)				
Instance	Obj	Succ	Fails	% fails	Tree	Setup	Solve	(s) / obj	(s) / cut
macrophage	53	13	40	75.5	2.0	2.6	4.7	0.1	0.4
mas074	13	12	1	7.7	0.2	0.3	3.8	0.3	0.3
mas076	12	11	1	8.3	0.2	0.3	3.6	0.3	0.3
mas284	42	4	38	90.5	0.6	1.0	50.1	1.2	12.5
maxgasflow	7	7	2	28.6	0.3	0.0	0.0	0.0	0.0
mc11	380	363	17	4.5	1.0	0.3	57.6	0.2	0.2
mcsched	430	416	14	3.3	4.0	4.3	893.8	2.1	2.1
mik-250-1-100-1	141	100	41	29.1	0.2	0.0	8.8	0.1	0.1
mine-90-10	138	103	35	25.4	6.5	54.4	843.6	6.1	8.2
misc03	20	18	2	10.0	0.3	1.4	6.7	0.3	0.4
misc07	17	16	1	5.9	1.0	7.7	23.9	1.4	1.5
mkc	23	20	3	13.0	0.2	1.3	0.3	0.0	0.0
mod008	8	6	2	25.0	0.1	0.2	2.4	0.3	0.4
mod013	5	5	0	0.0	0.1	0.1	0.1	0.0	0.0
modglob	45	29	16	35.6	0.1	0.0	0.9	0.0	0.0
n4-3	39	39	0	0.0	1.2	32.1	241.4	6.2	6.2
neos-1058477	28	19	9	32.1	0.8	37.1	516.3	18.4	27.2
neos-1112782	19	6	13	68.4	0.2	0.0	4.8	0.3	0.8
neos-1112787	33	6	27	81.8	0.1	0.0	12.6	0.4	2.1
neos-1215259	52	51	1	1.9	4.7	39.8	864.2	16.6	16.9
neos-1225589	52	12	40	76.9	0.9	2.7	82.5	1.6	6.9
neos-1281048	28	10	18	64.3	5.1	61.0	853.3	30.5	85.3
neos-1396125	64	63	1	1.6	2.1	23.3	191.4	3.0	3.0
neos-1413153	21	1	20	95.2	0.6	11.0	29.5	1.4	29.5
neos-1415183	257	253	4	1.6	0.7	13.0	157.2	0.6	0.6
neos-1420205	21	1	20	95.2	0.5	3.2	2.9	0.1	2.9
neos-1489999	107	106	1	0.9	2.2	17.4	886.4	8.3	8.4
neos-1582420	106	91	15	14.2	3.2	13.1	878.4	8.3	9.7
neos-1616732	266	117	149	56.0	10.6	43.7	860.1	3.2	7.4
neos-1620807	21	1	20	95.2	4.0	70.7	36.4	1.7	36.4
neos-480878	24	23	1	4.2	3.4	2.4	27.6	1.2	1.2
neos-501453	1	1	0	0.0	0.0	0.0	0.0	0.0	0.0
neos-504674	176	148	28	15.9	4.2	9.8	501.9	2.9	3.4
neos-504815	128	116	12	9.4	3.4	9.7	294.7	2.3	2.5
neos-512201	131	56	75	57.3	4.2	4.0	550.1	4.2	9.8
neos-522351	184	106	78	42.4	2.5	116.4	598.1	3.3	5.6
neos-538867	85	47	38	44.7	1.5	13.2	67.7	0.8	1.4
neos-538916	60	53	7	11.7	2.3	14.8	80.4	1.3	1.5
neos-570431	72	54	18	25.0	10.3	67.2	832.0	11.6	15.4
neos-593853	11	9	2	18.2	1.1	0.6	10.9	1.0	1.2
neos-598183	22	19	3	13.6	0.3	0.8	57.7	2.6	3.0
neos-603073	24	20	4	16.7	0.3	0.6	9.1	0.4	0.5

	Objectives				Time (s)				
Instance	Obj	Succ	Fails	% fails	Tree	Setup	Solve	(s) / obj	(s) / cut
neos-686190	82	80	2	2.4	1.5	7.1	882.8	10.8	11.0
neos-717614	17	15	2	11.8	0.1	0.0	0.1	0.0	0.0
neos-775946	75	19	57	76.0	0.6	2.7	82.5	1.1	4.3
neos-801834	58	54	4	6.9	4.9	25.5	873.9	15.1	16.2
neos-803219	22	20	2	9.1	19.0	8.4	213.3	9.7	10.7
neos-803220	21	20	1	4.8	5.5	4.3	124.1	5.9	6.2
neos-806323	66	31	35	53.0	0.4	0.0	0.4	0.0	0.0
neos-807639	132	116	16	12.1	13.4	5.2	597.4	4.5	5.2
neos-807705	162	142	20	12.3	2.9	3.8	880.2	5.4	6.2
neos-810326	171	166	5	2.9	2.5	7.6	887.4	5.2	5.3
neos-825075	121	6	115	95.0	2.6	57.3	808.9	6.7	134.8
neos-839859	180	16	164	91.1	0.9	4.7	892.3	5.0	55.8
neos-860300	98	95	3	3.1	13.0	4.3	241.2	2.5	2.5
neos-862348	79	30	49	62.0	2.9	64.8	832.1	10.5	27.7
neos-880324	92	11	81	88.0	2.8	6.0	11.0	0.1	1.0
neos-886822	163	158	5	3.1	0.3	0.7	26.0	0.2	0.2
neos-906865	39	34	5	12.8	2.5	33.9	174.7	4.5	5.1
neos-916792	21	20	1	4.8	7.9	34.0	899.5	42.8	45.0
neos-942830	20	1	19	95.0	3.3	318.9	600.1	30.0	600.1
neos14	32	14	18	56.3	0.4	0.0	0.1	0.0	0.0
neos15	34	16	18	52.9	0.4	0.0	0.1	0.0	0.0
neos17	172	171	1	0.6	1.5	2.4	83.2	0.5	0.5
neos18	41	4	39	95.1	2.0	5.3	38.6	0.9	9.7
neos3	42	28	14	33.3	18.8	49.8	16.1	0.4	0.6
neos5	21	1	20	95.2	0.6	2.6	1.5	0.1	1.5
neos7	46	13	33	71.7	0.9	1.5	150.6	3.3	11.6
newdano	41	2	39	95.1	3.9	351.2	434.9	10.6	217.5
ns1688347	7	5	3	42.9	11.9	6.5	13.7	2.0	2.7
ns1830653	162	113	49	30.2	8.8	36.2	867.4	5.4	7.7
nsrand-ipx	21	9	12	57.1	0.4	0.6	25.1	1.2	2.8
p0282	25	24	1	4.0	0.2	0.1	1.1	0.0	0.0
p0548	5	4	1	20.0	0.1	0.0	0.0	0.0	0.0
p100x588b	77	74	3	3.9	0.6	0.3	14.2	0.2	0.2
p2756	97	85	15	15.5	0.2	0.2	4.6	0.0	0.1
p6000	4	2	2	50.0	1.2	4.4	32.9	8.2	16.5
p6b	50	49	1	2.0	7.8	47.2	858.6	17.2	17.5
p80x400b	7	5	2	28.6	0.1	0.0	0.1	0.0	0.0
pg	5	3	2	40.0	0.8	0.2	1.8	0.4	0.6
pg5_34	26	20	6	23.1	0.0	0.0	0.1	0.0	0.0
pipex	7	6	1	14.3	0.1	0.0	0.1	0.0	0.0
pp08a	5	4	1	20.0	0.1	0.0	0.1	0.0	0.0
pp08aCUTS	47	46	1	2.1	0.4	0.9	15.8	0.3	0.3

	Objectives				Time (s)				
Instance	Obj	Succ	Fails	% fails	Tree	Setup	Solve	(s) / obj	(s) / cut
probportfolio	49	48	1	2.0	2.3	35.6	863.0	17.6	18.0
prod1	41	40	1	2.4	1.9	0.7	8.9	0.2	0.2
qiu	37	36	1	2.7	3.0	48.3	616.6	16.7	17.1
qnet1	53	47	6	11.3	0.3	1.7	49.3	0.9	1.0
qnet1_o	11	10	1	9.1	0.8	55.4	116.4	10.6	11.6
queens-30	15	1	14	93.3	93.4	60.8	863.5	57.6	863.5
r80x800	69	65	4	5.8	0.8	0.8	125.6	1.8	1.9
ran14x18	18	18	0	0.0	0.4	0.5	7.1	0.4	0.4
ran14x18-disj-8	25	24	1	4.0	8.7	40.5	860.0	34.4	35.8
ran16x16	21	20	1	4.8	0.4	0.8	8.8	0.4	0.4
reblock67	74	72	2	2.7	4.0	46.2	865.3	11.7	12.0
rgn	38	3	35	92.1	0.1	0.3	2.0	0.1	0.7
rococoB10-011000	100	98	2	2.0	1.8	28.1	859.9	8.6	8.8
rococoC10-001000	131	128	3	2.3	1.3	27.8	691.5	5.3	5.4
roll3000	66	57	9	13.6	3.0	0.7	56.9	0.9	1.0
rout	47	40	7	14.9	1.0	24.0	344.6	7.3	8.6
roy	12	11	1	8.3	0.3	0.0	0.1	0.0	0.0
sentoy	9	8	1	11.1	0.2	0.1	0.5	0.1	0.1
set1al	7	5	2	28.6	0.3	0.0	0.0	0.0	0.0
set1ch	160	129	31	19.4	0.5	0.1	11.4	0.1	0.1
set3-10	6	5	1	16.7	0.5	0.1	0.5	0.1	0.1
set3-15	12	9	3	25.0	0.2	0.1	1.2	0.1	0.1
set3-20	8	8	0	0.0	0.6	0.1	1.9	0.2	0.2
seymour	9	7	3	33.3	7.9	1.3	28.9	3.2	4.1
seymour-disj-10	117	103	15	12.8	28.4	14.5	877.2	7.5	8.5
sp98ir	102	99	3	2.9	3.9	16.0	886.6	8.7	9.0
stein27_nocard	21	1	20	95.2	0.2	0.1	0.3	0.0	0.3
stein45_nocard	21	1	20	95.2	1.0	0.3	0.9	0.0	0.9
timtab1	6	4	2	33.3	0.1	0.0	0.0	0.0	0.0
timtab2	9	8	1	11.1	0.1	0.0	0.0	0.0	0.0
toll-like	40	11	29	72.5	3.2	2.9	3.2	0.1	0.3
tr12-30	4	4	0	0.0	0.3	0.0	0.1	0.0	0.0
uct-subprob	36	5	31	86.1	12.4	52.8	324.2	9.0	64.8
umts	105	75	30	28.6	1.4	7.5	895.5	8.5	11.9
vpm2	27	24	3	11.1	0.2	0.0	0.4	0.0	0.0
Average				30.7	3.7	17.6	257.6	4.8	19.7

D.6 Tables for experiments with other cut-generating sets

The tables in this section show that, overall, using partial branch-and-bound trees leads to comparable or better percent gap closed than generating cuts from multiple split or cross disjunctions (Table D.7), in less time (Table D.8) and with fewer cuts (Table D.9). A summary of these tables appeared in Table 4.7.

The columns of Table D.7 give the following information for each instance. Columns 2 and 3 give the dimensions of the instance. Column 4 gives the number of GMICs generated (one for each elementary split on an integer variable fractional at $\bar{x}$). Column 5 contains the number of VPCs generated, while the next column (6) specifies the number of cuts that are active, i.e., tight, at the optimum of the LP after adding the cuts. Finally, columns 7 and 8 give the percentage of the integrality gap closed by the GMICs and the VPCs. The last column (9) gives the time used for generating the cuts.

Table D.7: Comparison of percent gap closed by VPCs from partial branch-and-bound trees to using multiple split or cross disjunctions.

Instance	G	V+G (2)	V+G (4)	V+G (8)	V+G (16)	V+G (32)	V+G (64)	V+G (best)	V+G (splits)	V+G (crosses)
23588	5.8	17.3	28.7	35.7	47.8	60.9	71.8	71.8	28.7	39.8
bell3a	37.0	37.0	37.0	37.0	43.6	43.6	43.6	43.6	37.0	43.9
bell3b	31.2	84.1	84.1	84.1	84.5	84.5	84.7	84.7	51.4	84.3
bell4	25.6	25.6	25.8	26.1	26.0	26.1	26.4	26.4	25.6	29.9
bell5	13.8	13.8	13.8	25.7	25.6	77.8	62.9	77.8	14.8	22.7
blend2	5.5	5.5	7.5	12.1	13.1	15.5	23.0	23.0	7.1	8.9
bm23	16.8	18.0	19.8	19.8	39.5	56.2	70.9	70.9	18.0	20.2
glass4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
go19	2.0	2.5	3.7	5.6	9.8	13.1	2.0	13.1	8.5	5.8
gt2	87.1	87.1	87.1	87.1	87.1	87.1	87.1	87.1	87.1	87.1
k16x240	11.4	11.4	12.9	12.8	16.9	17.7	22.5	22.5	13.5	16.4
lseu	5.8	6.1	6.1	6.1	10.1	11.4	19.2	19.2	6.2	9.1
mas074	6.7	6.7	7.2	7.9	9.0	11.1	13.4	13.4	6.9	7.2
mas076	6.4	6.4	6.5	6.4	6.8	8.9	13.0	13.0	6.4	6.5
mas284	0.9	1.8	8.7	12.4	15.6	25.3	33.8	33.8	3.3	10.6
mik-250-1-100-1	53.5	53.5	53.5	53.5	53.5	53.5	53.5	53.5	53.5	53.5
misc03	0.0	0.0	0.0	5.4	10.5	17.4	44.3	44.3	0.0	4.1
misc07	0.7	0.7	0.7	0.7	0.9	1.9	5.5	5.5	0.7	0.7
mod008	24.4	24.4	24.4	25.8	25.5	26.7	27.8	27.8	24.4	24.4
mod013	5.9	9.0	9.9	20.6	21.1	36.9	47.4	47.4	9.0	11.6
modglob	18.1	18.5	18.8	19.6	18.9	18.4	18.6	19.6	30.3	44.8
neos-1420205	0.0	0.0	0.0	0.0	0.0	11.4	14.7	14.7	0.0	0.4
neos5	11.1	11.1	11.1	18.8	21.9	29.2	37.5	37.5	11.2	15.7
neos-880324	0.0	0.0	0.0	0.0	0.0	0.0	36.3	36.3	8.5	16.9
p0282	3.2	3.2	3.2	3.4	6.4	7.3	10.6	10.6	65.2	77.9
pipex	35.6	35.6	35.7	35.7	35.6	35.6	36.7	36.7	35.6	35.6
pp08aCUTS	33.8	33.8	34.9	34.9	35.7	34.6	33.8	35.7	39.9	51.9
pp08a	54.5	54.5	54.5	54.5	54.5	54.5	54.5	54.5	56.1	70.3
probportfolio	4.6	4.8	5.4	5.7	7.0	8.4	11.3	11.3	14.7	13.5
prod1	4.7	5.1	8.3	12.2	18.6	29.6	36.3	36.3	9.7	22.4
rgn	8.0	8.0	8.0	8.0	21.6	33.8	45.4	45.4	8.0	8.1
roy	4.5	7.1	10.4	11.3	15.8	24.3	30.3	30.3	7.4	9.3
sentoy	19.3	19.3	20.0	27.5	45.9	55.0	59.8	59.8	21.3	22.9
stein27_nocard	8.3	13.1	17.9	29.5	44.4	50.0	59.3	59.3	28.5	36.4
timtab1	24.1	24.1	24.1	24.1	24.1	24.1	24.1	24.1	27.2	34.1
vpm1	15.7	15.7	15.7	15.7	27.6	26.2	22.8	27.6	16.5	17.3
vpm2	18.2	18.6	18.2	18.6	19.0	19.3	20.1	20.1	23.8	31.8
Average	16.33	18.48	19.55	21.73	25.52	30.74	35.26	36.17	21.79	26.92

Table D.8: Time to generate VPCs for each of the different partial branch-and-bound tree sizes and for multiple split and cross disjunctions.

Instance	V (2)	V (4)	V (8)	V (16)	V (32)	V (64)	V (splits)	V (crosses)
23588	0.5	2.3	8.8	23.9	77.5	224.8	36.2	912.9
bell3a	0.0	0.0	0.0	0.0	0.1	0.2	0.0	0.1
bell3b	0.0	0.0	0.0	0.1	0.1	0.2	0.1	2.4
bell4	0.0	0.0	0.0	0.1	0.1	0.2	0.1	2.6
bell5	0.0	0.0	0.0	0.0	0.1	0.1	0.1	0.3
blend2	0.1	0.2	0.2	0.5	1.6	4.3	0.6	6.9
bm23	0.0	0.0	0.0	0.1	0.1	0.3	0.1	0.2
glass4	0.0	0.1	0.1	0.3	0.5	1.6	1.5	100.8
go19	1.6	7.5	52.1	314.4	906.3	71.7	619.5	926.5
gt2	0.0	0.0	0.0	0.2	0.4	0.9	0.2	2.9
k16x240	0.1	0.1	0.3	0.7	1.6	3.5	0.7	8.8
lseu	0.0	0.0	0.0	0.1	0.2	0.4	0.1	1.0
mas074	0.0	0.1	0.2	0.6	1.4	4.4	0.4	4.8
mas076	0.0	0.1	0.3	0.8	1.9	4.2	0.4	4.8
mas284	0.1	0.3	0.8	4.1	13.5	52.2	1.7	48.8
mik-250-1-100-1	0.1	0.1	0.2	0.4	2.1	9.3	3.8	327.5
misc03	0.1	0.1	0.5	1.1	3.2	8.7	0.9	20.1
misc07	0.1	0.3	0.9	3.0	13.0	33.3	1.3	29.5
mod008	0.1	0.1	0.2	0.4	1.2	2.9	0.2	1.2
mod013	0.0	0.0	0.0	0.1	0.2	0.4	0.1	0.2
modglob	0.1	0.2	0.4	1.2	1.9	3.4	0.8	18.6
neos-1420205	0.2	0.3	0.3	0.9	4.1	7.1	2.4	170.8
neos5	0.1	0.2	0.4	0.7	1.8	4.9	1.4	114.6
neos-880324	0.1	0.6	0.4	1.3	15.3	20.1	2.2	267.1
p0282	0.0	0.1	0.1	0.4	0.7	1.6	0.5	10.9
pipex	0.0	0.0	0.0	0.1	0.1	0.2	0.0	0.2
pp08aCUTS	0.1	0.1	0.2	0.8	2.9	17.6	3.7	184.6
pp08a	0.0	0.0	0.1	0.2	0.4	1.1	0.9	39.1
probportfolio	2.3	7.4	41.2	264.7	682.7	903.3	88.5	907.2
prod1	0.1	0.3	1.2	2.7	6.1	11.8	2.0	218.3
rgn	0.0	0.1	0.2	0.3	1.4	2.5	0.4	7.4
roy	0.0	0.0	0.1	0.2	0.3	0.6	0.2	1.4
sentoy	0.0	0.0	0.1	0.2	0.4	0.9	0.1	0.7
stein27_nocard	0.0	0.1	0.1	0.2	0.4	0.5	0.4	19.2
timtab1	0.0	0.0	0.1	0.2	0.3	0.7	2.3	339.6
vpm1	0.0	0.0	0.1	0.2	0.3	0.7	0.1	1.1
vpm2	0.0	0.0	0.1	0.2	0.4	0.8	0.5	8.9
Geomean	0.16	0.54	2.34	7.24	13.16	14.43	7.38	53.40

Table D.9: Number of rows, columns, GMICs, and VPCs for small instances used to test multiple split and cross disjunctions. The last row gives the average ratio of number of VPCs as a fraction of the number of GMICs.

			# cuts								
Instance	Rows	Cols	G	V (2)	V (4)	V (8)	V (16)	V (32)	V (64)	V (splits)	V (crosses)
23588	137	237	75	36	75	75	75	75	75	4890	32920
bell3a	63	82	7	1	5	7	4	5	7	4	19
bell3b	73	91	24	3	3	7	21	25	25	63	328
bell4	73	88	27	3	4	8	5	7	9	22	178
bell5	34	56	10	3	2	5	4	10	10	19	82
blend2	154	302	13	13	13	13	13	13	13	117	840
bm23	20	27	6	6	6	6	6	6	6	36	90
glass4	392	317	72	2	4	10	14	0	72	142	906
go19	361	361	357	97	228	357	357	175	0	35760	22557
gt2	28	173	14	3	4	3	7	2	1	21	242
k16x240	256	480	14	14	9	14	14	14	14	166	977
lseu	28	79	9	9	9	9	9	9	9	81	280
mas074	13	148	12	12	12	12	12	12	12	144	779
mas076	12	148	11	11	8	4	10	11	11	112	497
mas284	68	148	20	20	20	20	6	9	4	387	3607
mik-250-1-100-1	100	251	100	1	2	8	13	74	100	1	100
misc03	95	138	18	18	18	18	18	18	18	310	2616
misc07	211	232	16	16	16	16	16	16	16	256	1901
mod008	6	319	6	6	6	3	6	6	6	30	88
mod013	62	96	5	5	5	5	5	5	5	23	50
modglob	286	354	29	21	25	29	29	29	29	209	1049
neos-1420205	341	231	44	40	44	0	0	2	1	613	9756
neos5	63	63	35	25	35	2	1	1	1	719	8989
neos-880324	182	135	45	10	7	0	0	0	11	543	17738
p0282	160	200	24	13	10	24	24	24	24	181	1697
pipex	25	48	6	6	6	6	6	6	6	19	42
pp08aCUTS	239	235	46	6	7	3	5	11	46	924	12766
pp08a	133	234	53	4	5	4	4	2	2	246	1145
probportfolio	302	320	105	105	105	105	105	105	48	6157	9432
prod1	75	117	40	18	40	40	40	40	40	1184	30508
rgn	24	180	18	16	18	18	18	18	3	222	1606
roy	147	139	11	7	6	11	11	11	11	104	368
sentoy	30	60	8	8	8	8	8	8	8	64	224
stein27_nocard	117	27	27	7	11	27	1	2	1	163	6102
timtab1	165	365	128	5	2	3	4	2	2	557	13566
vpm1	128	188	11	11	0	5	11	9	11	56	104
vpm2	127	187	24	24	7	13	24	24	24	247	1209
Avg (cut ratio)				0.63	0.64	0.66	0.69	0.71	0.72	14.27	108.23

Bibliography

[1] T. Achterberg. “Conflict analysis in mixed integer programming”. *Discrete Optim.* 4.1 (2007), 4–20.

[2] T. Achterberg, T. Berthold, T. Koch, and K. Wolter. “Constraint integer programming: a new approach to integrate CP and MIP”. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems: 5th International Conference, CPAIOR 2008 Paris, France, May 20-23, 2008 Proceedings.* Ed. by L. Perron and M. A. Trick. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, 6–20.

[3] T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. “Presolve reductions in mixed integer programming”. Tech. rep. 16-44. Takustr.7, 14195 Berlin: ZIB, 2016.

[4] T. Achterberg, T. Koch, and A. Martin. “Branching rules revisited”. *Oper. Res. Lett.* 33.1 (2005), 42–54.

[5] T. Achterberg, T. Koch, and A. Martin. “MIPLIB 2003”. *Oper. Res. Lett.* 34.4 (2006), 361–372.

[6] T. Achterberg and R. Wunderling. “Mixed integer programming: analyzing 12 years of progress”. *Facets of Combinatorial Optimization.* Springer, Heidelberg, 2013, 449–481.

[7] K. Andersen, G. Cornuéjols, and Y. Li. “Split closure and intersection cuts”. *Integer Programming and Combinatorial Optimization.* Vol. 2337. Lecture Notes in Comput. Sci. Springer, Berlin, 2002, 127–144.

[8] K. Andersen, G. Cornuéjols, and Y. Li. “Split closure and intersection cuts”. *Math. Program.* 102.3, Ser. A (2005), 457–493.

[9] K. Andersen, Q. Louveaux, R. Weismantel, and L. A. Wolsey. “Inequalities from two rows of a simplex tableau”. *Integer Programming and Combinatorial Optimization.* Vol. 4513. Lecture Notes in Comput. Sci. Springer, Berlin, 2007, 1–15.

[10] K. Andersen and Y. Pochet. “Coefficient strengthening: a tool for reformulating mixed-integer programs”. *Math. Program.* 122.1, Ser. A (2010), 121–154.

[11] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. “Finding cuts in the TSP (a preliminary report)”. Tech. rep. Center for Discrete Mathematics & Theoretical Computer Science, 1995.

[12] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. "TSP cuts which do not conform to the template paradigm". *Computational Combinatorial Optimization (Schloß Dagstuhl, 2000).* Vol. 2241. Lecture Notes in Comput. Sci. Springer, Berlin, 2001, 261–303.

[13] A. Atamtürk, G. L. Nemhauser, and M. W. P. Savelsbergh. "Conflict graphs in solving integer programming problems". *European J. Oper. Res.* 121.1 (2000), 40–55.

[14] A. Atamtürk, G. L. Nemhauser, and M. W. P. Savelsbergh. "The mixed vertex packing problem". *Math. Program.* 89.1, Ser. A (2000), 35–53.

[15] G. Averkov, C. Wagner, and R. Weismantel. "Maximal lattice-free polyhedra: finiteness and an explicit description in dimension three". *Math. Oper. Res.* 36.4 (2011), 721–742.

[16] Y. Awate, G. Cornuéjols, B. Guenin, and L. Tunçel. "On the relative strength of families of intersection cuts arising from pairs of tableau constraints in mixed integer programs". *Math. Program.* 150.2, Ser. A (2015), 459–489.

[17] E. Balas. "Disjunctive programming". *Ann. Discrete Math.* 5 (1979), 3–51.

[18] E. Balas. "Disjunctive programming: cutting planes from logical conditions". Ed. by O. Mangasarian, R. Meyer, and S. Robinson. Academic Press, New York, 1975, 279–312.

[19] E. Balas. "Disjunctive programming: properties of the convex hull of feasible points". *Discrete Appl. Math.* 89.1-3 (1998), 3–44.

[20] E. Balas. "Facets of the knapsack polytope". *Math. Program.* 8 (1975), 146–164.

[21] E. Balas. "Intersection cuts—a new type of cutting planes for integer programming". *Oper. Res.* 19.1 (1971), 19–39.

[22] E. Balas and P. Bonami. "Generating lift-and-project cuts from the LP simplex tableau: open source implementation and testing of new variants". *Math. Program. Comput.* 1.2-3 (2009), 165–199.

[23] E. Balas and P. Bonami. "New variants of lift-and-project cut generation from the LP tableau: open source implementation and testing". *Integer Programming and Combinatorial Optimization.* Vol. 4513. Lecture Notes in Comput. Sci. Springer, Berlin, 2007, 89–103.

[24] E. Balas, S. Ceria, and G. Cornuéjols. "A lift-and-project cutting plane algorithm for mixed 0-1 programs". *Math. Program.* 58.3, Ser. A (1993), 295–324.

[25] E. Balas, S. Ceria, and G. Cornuéjols. "Mixed 0-1 programming by lift-and-project in a branch-and-cut framework". *Man. Sci.* 42.9 (1996), 1229–1246.

[26] E. Balas, S. Ceria, G. Cornuéjols, and N. Natraj. "Gomory cuts revisited". *Oper. Res. Lett.* 19.1 (1996), 1–9.

[27] E. Balas, M. Fischetti, and A. Zanette. "On the enumerative nature of Gomory's dual cutting plane method". *Math. Program.* 125.2, Ser. B (2010), 325–351.

[28] E. Balas and R. G. Jeroslow. "Strengthening cuts for mixed integer programs". *European J. Oper. Res.* 4.4 (1980), 224–234.

[29] E. Balas and T. Kis. “Intersection cuts—standard versus restricted”. *Discrete Optim.* 18 (2015), 189–192.

[30] E. Balas and T. Kis. “On the relationship between standard intersection cuts, lift-and-project cuts and generalized intersection cuts”. *Math. Progam.* (2016), 1–30.

[31] E. Balas and F. Margot. “Generalized intersection cuts and a new cut generating paradigm”. *Math. Program.* 137.1-2, Ser. A (2013), 19–35.

[32] 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, Ser. B (2003). The Aussois 2000 Workshop in Combinatorial Optimization, 221–245.

[33] E. Balas and M. Perregaard. “Lift-and-project for mixed 0-1 programming: recent progress”. *Discrete Appl. Math.* 123.1-3 (2002). Workshop on Discrete Optimization, DO’99 (Piscataway, NJ), 129–154.

[34] E. Balas and A. Qualizza. “Intersection cuts from multiple rows: a disjunctive programming approach”. *EURO J. Computat. Optim.* 1.1 (2013), 3–49.

[35] E. Balas and A. Saxena. “Optimizing over the split closure”. *Math. Progam.* 113.2, Ser. A (2008), 219–240.

[36] A. Barvinok. *A course in convexity.* Vol. 54. Graduate Studies in Mathematics. American Mathematical Society, Providence, RI, 2002, x+366.

[37] A. Basu, P. Bonami, G. Cornuéjols, and F. Margot. “Experiments with two-row cuts from degenerate tableaux”. *INFORMS J. Comput.* 23.4 (2011), 578–590.

[38] A. Basu, P. Bonami, G. Cornuéjols, and F. Margot. “On the relative strength of split, triangle and quadrilateral cuts”. *Math. Program.* 126.2, Ser. A (2011), 281–314.

[39] A. Basu, G. Cornuéjols, and M. Molinaro. “A probabilistic analysis of the strength of the split and triangle closures”. *Integer Programming and Combinatorial Optimization.* Vol. 6655. Lecture Notes in Comput. Sci. Springer, Heidelberg, 2011, 27–38.

[40] D. E. Bell and M. L. Fisher. “Improved integer programming bounds using intersections of corner polyhedra”. *Math. Program.* 8 (1975), 345–368.

[41] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan. “Mixed-integer nonlinear optimization”. *Acta Numer.* 22 (2013), 1–131.

[42] D. Bienstock, C. Chen, and G. Muñoz. “Outer-product-free sets for polynomial optimization and oracle-based cuts”. 2018.

[43] R. E. Bixby, E. A. Boyd, and R. R. Indovina. “MIPLIB: a test set of mixed integer programming problems”. *SIAM News* 25 (2 1992), 16.

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

[45] R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. “MIP: theory and practice—closing the gap”. *System Modelling and Optimization (Cambridge, 1999).* Kluwer Acad. Publ., Boston, MA, 2000, 19–49.

[46] R. Bixby and E. Rothberg. “Progress in computational mixed integer programming—a look back from the other side of the tipping point”. *Ann. Oper. Res.* 149 (2007). History of integer programming: distinguished personal notes and reminiscences, 37–41.

[47] P. Bonami. “On optimizing over lift-and-project closures”. *Math. Program. Comput.* 4.2 (2012), 151–179.

[48] P. Bonami, A. Lodi, A. Tramontani, and S. Wiese. “Cutting planes from wide split disjunctions”. *Integer Programming and Combinatorial Optimization.* Ed. by F. Eisenbrand and J. Könemann. Vol. 10328. Lecture Notes in Comput. Sci. Springer, Cham, 2017, 99–110.

[49] V. Borozan and G. Cornuéjols. “Minimal valid inequalities for integer constraints”. *Math. Oper. Res.* 34.3 (2009), 538–546.

[50] G. H. Bradley, P. L. Hammer, and L. Wolsey. “Coefficient reduction for inequalities in 0-1 variables”. *Math. Program.* 7 (1974), 263–282.

[51] C. Buchheim, F. Liers, and M. Oswald. “Local cuts revisited”. *Oper. Res. Lett.* 36.4 (2008), 430–433.

[52] B. Chen, S. Küçükyavuz, and S. Sen. “A computational study of the cutting plane tree algorithm for general mixed-integer linear programs”. *Oper. Res. Lett.* 40.1 (2012), 15–19.

[53] B. Chen, S. Küçükyavuz, and S. Sen. “Finite disjunctive programming characterizations for general mixed-integer linear programs”. *Oper. Res.* 59.1 (2011), 202–210.

[54] V. Chvátal, W. Cook, and D. Espinoza. “Local cuts for mixed-integer programming”. *Math. Program. Comput.* 5.2 (2013), 171–200.

[55] “COIN-OR Cut Generation Library”. `https://projects.coin-or.org/Clp/`.

[56] “Computational Optimization Research at Lehigh. MIP instances”. `http://coral.ise.lehigh.edu/data-sets/mixed-integer-instances/`. Accessed September 2017.

[57] M. Conforti, G. Cornuéjols, and G. Zambelli. “Corner polyhedron and intersection cuts”. *Surveys in Operations Research and Management Science* 16.2 (2011), 105–120.

[58] M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer programming.* Vol. 271. Graduate Texts in Mathematics. Springer, Cham, 2014, xii+456.

[59] M. Conforti and L. A. Wolsey. "“Facet” separation with one linear program”. CORE Discussion Papers 2016021. Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), 2016.

[60] W. Cook, S. Dash, R. Fukasawa, and M. Goycoolea. “Numerically safe gomory mixed-integer cuts”. *INFORMS J. Comput.* 21.4 (2009), 641–649.

[61] W. Cook, R. Kannan, and A. Schrijver. “Chvátal closures for mixed integer programming problems”. *Math. Program.* 47.2, (Ser. A) (1990), 155–174.

[62] G. Cornuéjols and C. Lemaréchal. “A convex-analysis perspective on disjunctive cuts”. *Math. Program.* 106.3, Ser. A (2006), 567–586.

[63] G. Cornuéjols, L. Liberti, and G. Nannicini. “Improved strategies for branching on general disjunctions”. *Math. Program.* 130.2, Ser. A (2011), 225–247.

[64] G. Cornuéjols. “The ongoing story of Gomory cuts”. *Extra Volume: Optimization Stories.* Documenta Mathematica, 2012, 221–226.

[65] H. Crowder, E. L. Johnson, and M. Padberg. “Solving large-scale zero-one linear programming problems”. *Oper. Res.* 31.5 (1983), 803–834.

[66] S. Dash and O. Günlük. “On t-branch split cuts for mixed-integer programs”. *Math. Program.* 141.1-2, Ser. A (2013), 591–599.

[67] S. Dash, O. Günlük, and A. Lodi. “MIR closures of polyhedral sets”. *Math. Program.* 121.1, Ser. A (2010), 33–60.

[68] S. Dash, O. Günlük, and M. Molinaro. “On the relative strength of different generalizations of split cuts”. *Discrete Optim.* 16 (2015), 36–50.

[69] S. Dash, O. Günlük, and D. A. Morán R. “On the polyhedrality of cross and quadrilateral closures”. *Math. Program.* (2016), 1–26.

[70] S. Dash, O. Günlük, and J. P. Vielma. “Computational experiments with cross and crooked cross cuts”. *INFORMS J. Comput.* 26.4 (2014), 780–797.

[71] A. Del Pia, C. Wagner, and R. Weismantel. “A probabilistic comparison of the strength of split, triangle, and quadrilateral cuts”. *Oper. Res. Lett.* 39.4 (2011), 234–240.

[72] S. S. Dey, A. Lodi, A. Tramontani, and L. A. Wolsey. “On the practical strength of two-row tableau cuts”. *INFORMS J. Comput.* 26.2 (2014), 222–237.

[73] S. S. Dey and S. Pokutta. “Design and verify: a new scheme for generating cutting-planes”. *Math. Program.* 145.1-2, Ser. A (2014), 199–222.

[74] S. S. Dey and J.-P. Richard. “Linear-programming-based lifting and its application to primal cutting-plane algorithms”. *INFORMS J. Comput.* 21.1 (2009), 137–150.

[75] S. S. Dey and L. A. Wolsey. “Two row mixed-integer cuts via lifting”. *Math. Program.* 124.1-2, Ser. B (2010), 143–174.

[76] B. L. Dietrich and L. F. Escudero. “Coefficient reduction for knapsack-like constraints in 0-1 programs with variable upper bounds”. *Oper. Res. Lett.* 9.1 (1990), 9–14.

[77] B. L. Dietrich, L. F. Escudero, and F. Chance. “Efficient reformulation for 0-1 programs—methods and computational results”. *Discrete Appl. Math.* 42.2-3 (1993), 147–175.

[78] L. F. Escudero, S. Martello, and P. Toth. “On tightening 0-1 programs based on extensions of pure 0-1 knapsack and subset-sum problems”. *Ann. Oper. Res.* 81 (1998). Applied mathematical programming and modeling, III (APMOD95) (Uxbridge), 379–404.

[79] D. G. Espinoza. “Computing with multi-row Gomory cuts”. *Oper. Res. Lett.* 38.2 (2010), 115–120.

[80] D. Espinoza, R. Fukasawa, and M. Goycoolea. “Lifting, tilting and fractional programming revisited”. *Oper. Res. Lett.* 38.6 (2010), 559–563.

[81] J. Farkas. “Theorie der einfachen ungleichungen”. *J. Reine Angew. Math.* 124 (1902), 1–27.

[82] M. C. Ferris, G. Pataki, and S. Schmieta. “Solving the *seymour* problem”. *Optima* 66 (2001), 2–6.

[83] T. Fischer and M. E. Pfetsch. “Monoidal cut strengthening and generalized mixed-integer rounding for disjunctions and complementarity constraints”. *Oper. Res. Lett.* 45.6 (2017), 556–560.

[84] M. Fischetti, A. Lodi, and A. Tramontani. “On the separation of disjunctive cuts”. *Math. Program.* 128.1-2, Ser. A (2011), 205–230.

[85] M. Fischetti and M. Monaci. “Exploiting erraticism in search”. *Oper. Res.* 62.1 (2014), 114–122.

[86] D. R. Fulkerson, G. L. Nemhauser, and L. E. Trotter. “Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems”. *Math. Program. Stud.* 2 (1974), 72–81.

[87] D. Gade, S. Küçükyavuz, and S. Sen. “Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs”. *Math. Program.* 144.1-2, Ser. A (2014), 39–64.

[88] R. E. Gomory. “Outline of an algorithm for integer solutions to linear programs”. *Bull. Amer. Math. Soc.* 64 (1958), 275–278.

[89] R. E. Gomory. “Some polyhedra related to combinatorial problems”. *Linear Algebra Appl.* 2.4 (1969), 451–558.

[90] R. E. Gomory and E. L. Johnson. “Some continuous functions related to corner polyhedra”. *Math. Program.* 3.1 (1972), 23–85.

[91] R. E. Gomory and E. L. Johnson. “Some continuous functions related to corner polyhedra. II”. *Math. Program.* 3.1 (1972), 359–389.

[92] Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. “Lifted cover inequalities for 0-1 integer programs: computation”. *INFORMS J. Comput.* 10.4 (1998), 427–437.

[93] Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh. “Sequence independent lifting in mixed integer programming”. *J. Comb. Optim.* 4.1 (2000), 109–129.

[94] Y. Guan, S. Ahmed, and G. L. Nemhauser. “Sequential pairing of mixed integer inequalities”. *Discrete Optim.* 4.1 (2007), 21–39.

[95] O. Günlük and Y. Pochet. “Mixing mixed-integer inequalities”. *Math. Program.* 90.3, Ser. A (2001), 429–457.

[96] Gurobi Optimization, Inc. “Gurobi Optimizer Reference Manual”. `http://www.gurobi.com`. Version 7.5.1. 2017.

[97] Q. He, S. Ahmed, and G. L. Nemhauser. “A probabilistic comparison of split and type 1 triangle cuts for two-row mixed-integer programs”. *SIAM J. Optim.* 21.3 (2011), 617–632.

[98] J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of convex analysis.* Grundlehren Text Editions. Springer-Verlag, Berlin, 2001, x+259.

[99] R. Jeroslow. "The principles of cutting-plane theory: part i". Tech. rep. AD0779955. Pittsburgh, PA: GSIA, Carnegie-Mellon University, Mar. 1974.

[100] E. L. Johnson. "On the group problem for mixed integer programming". *Math. Program. Stud.* 2 (1974), 137–179.

[101] E. L. Johnson, G. L. Nemhauser, and M. W. P. Savelsbergh. "Progress in linear programming-based algorithms for integer programming: an exposition". *INFORMS J. Comput.* 12.1 (2000), 2–23.

[102] M. Jörg. "k-disjunctive cuts and cutting plane algorithms for general mixed integer linear programs". Ph.D. Dissertation. Technischen Universität München, Aug. 2008.

[103] M. Karamanov. "Branch and cut: an empirical study". Ph.D. Dissertation. Carnegie Mellon University, Sept. 2006.

[104] M. Karamanov and G. Cornuéjols. "Branching on general disjunctions". *Math. Program.* 128.1-2, Ser. A (2011), 403–436.

[105] R. M. Karp. "Reducibility among combinatorial problems" (1972), 85–103.

[106] A. Kazachkov. "Cutting planes by tilting". Poster presented at the Mixed Integer Programing Workshop 2016. 2016.

[107] F. Kianfar. "Stronger inequalities for 0, 1 integer programming using knapsack functions". *Operations Res.* 19 (1971), 1374–1392.

[108] M. Kılınç, J. Linderoth, J. Luedtke, and A. Miller. "Strong-branching inequalities for convex mixed integer nonlinear programs". *Comput. Optim. Appl.* 59.3 (2014), 639–665.

[109] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. "MIPLIB 2010: mixed integer programming library version 5". *Math. Program. Comput.* 3.2 (2011), 103–163.

[110] Y. Li and J.-P. P. Richard. "Cook, Kannan and Schrijver's example revisited". *Discrete Optim.* 5.4 (2008), 724–734.

[111] A. Lodi. "Mixed integer programming computation". *50 Years of Integer Programming 1958–2008.* Ed. by M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2010, 619–645.

[112] R. Lougee-Heimer. "The Common Optimization INterface for Operations Research: promoting open-source software in the operations research community". *IBM Journal of Research and Development* 47 (2003).

[113] Q. Louveaux and L. Poirrier. "An algorithm for the separation of two-row cuts". *Math. Program.* 143.1-2, Ser. A (2014), 111–146.

[114] Q. Louveaux, L. Poirrier, and D. Salvagnin. "The strength of multi-row models". *Math. Program. Comput.* 7.2 (2015), 113–148.

[115] L. Lovász. “Geometry of numbers and integer programming”. *Mathematical programming (Tokyo, 1988).* Vol. 6. Math. Appl. (Japanese Ser.) SCIPRESS, Tokyo, 1989, 177–201.

[116] O. L. Mangasarian. *Nonlinear programming.* Vol. 10. Classics in Applied Mathematics. Corrected reprint of the 1969 original. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994, xvi+220.

[117] R. R. Meyer. “On the existence of optimal solutions to integer and mixed-integer programming problems”. *Math. Program.* 7 (1974), 223–235.

[118] M. Miltenberger, T. Ralphs, and D. Steffy. “Exploring the numerics of branch-and-cut for mixed integer linear optimization”. Tech. rep. 17-43. Takustr.7, 14195 Berlin: ZIB, 2017.

[119] G. L. Nemhauser and L. E. Trotter Jr. “Properties of vertex packing and independence system polyhedra”. *Math. Program.* 6 (1974), 48–61.

[120] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization.* Wiley-Interscience Series in Discrete Mathematics and Optimization. A Wiley-Interscience Publication. John Wiley & Sons, Inc., New York, 1988, xvi+763.

[121] G. Owen. “A cutting plane approach to the fixed charge problem”. Unpublished report for Mathematica. Princeton, NJ, 1965.

[122] M. Padberg. “Classical cuts for mixed-integer programming and branch-and-cut”. *Ann. Oper. Res.* 139 (2005), 321–352.

[123] M. W. Padberg. “A note on zero-one programming”. *Oper. Res.* 23.4 (1975), 833.

[124] M. W. Padberg. “On the facial structure of set packing polyhedra”. *Math. Program.* 5 (1973), 199–215.

[125] M. Padberg and G. Rinaldi. “A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems”. *SIAM Rev.* 33.1 (1991), 60–100.

[126] M. Perregaard. “Generating disjunctive cuts for mixed integer programs”. Ph.D. Dissertation. Carnegie Mellon University, Sept. 2003.

[127] M. Perregaard and E. Balas. “Generating cuts from multiple-term disjunctions”. *Integer Programming and Combinatorial Optimization.* Vol. 2081. Lecture Notes in Comput. Sci. Springer, Berlin, 2001, 348–360.

[128] Y. Pochet and L. A. Wolsey. “Lot-sizing with constant batches: formulation and valid inequalities”. *Math. Oper. Res.* 18.4 (1993), 767–785.

[129] Y. Qi and S. Sen. “The ancestral Benders’ cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming”. *Math. Program.* 161.1-2, Ser. A (2017), 193–235.

[130] J.-P. P. Richard. “Lifting techniques for mixed integer programming”. *Wiley Encyclopedia of Operations Research and Management Science.* Ed. by J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith. John Wiley & Sons, Inc., 2010.

[131] M. W. P. Savelsbergh. “Preprocessing and probing techniques for mixed integer programming problems”. *ORSA J. Comput.* 6.4 (1994), 445–454.

[132] S. Sen and H. D. Sherali. "Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming". 106.2 (2006), 203–223.

[133] T. Serra. "Reformulating the disjunctive cut generating linear program". Poster presented at the Mixed Integer Programing Workshop 2016. 2016.

[134] H. D. Sherali, Y. Lee, and Y. Kim. "Partial convexification cuts for 0-1 mixed-integer programs". *European J. Oper. Res.* 165.3 (2005), 625–648.

[135] H. D. Sherali and J. C. Smith. "Higher-level RLT or disjunctive cuts based on a partial enumeration strategy for 0-1 mixed-integer programs". *Optim. Lett.* 6.1 (2012), 127–139.

[136] J. L. C. Soares. "Disjunctive convex optimization". Ph.D. Dissertation. Columbia University, 1998.

[137] L. A. Wolsey. "Facets and strong valid inequalities for integer programs". *Oper. Res.* 24.2 (1976), 367–372.

[138] Y. Yuan and S. Sen. "Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs". *INFORMS J. Comput.* 21.3 (2009), 480–487.

[139] A. Zanette, M. Fischetti, and E. Balas. "Lexicography and degeneracy: can a pure cutting plane algorithm work?" *Math. Program.* 130.1, Ser. A (2011), 153–176.

[140] G. M. Ziegler. *Lectures on polytopes*. Vol. 152. Graduate Texts in Mathematics. Springer-Verlag, New York, 1995, x+370.