University of Liège

Montefiore Institute

Multi-row Approaches to Cutting Plane Generation

PhD thesis

Laurent Poirrier

Advisor:	Committee:
Prof. Quentin Louveaux	**Prof. Bernard Boigelot**
	Prof. Gérard Cornuéjols
	Prof. Yves Crama
	Prof. Santanu Dey
	Prof. Domenico Salvagnin
	Prof. Rodolphe Sepulchre (president)

October 2012
(Updated May 21, 2014)

Contents

Abstract

This thesis focuses on the use of cutting-plane techniques to improve general-purpose mixed-integer linear programming solvers.

The first topic covered here is a fast separation method for two-row cuts. Two-row cuts are intersection cuts from two rows of a simplex tableau describing the LP relaxation of the problem. These types of cuts recently garnered a lot of attention from the scientific community following a paper by Andersen, Louveaux, Weismantel and Wolsey describing the facets of the underlying two-row model and providing an intuitive geometric classification the derived cuts. The specificity of the approach adopted here is that it does not rely on an “infinite relaxation” point of view and generate intersection cuts from fixed lattice-free sets. Instead, given a fractional point x^*, it aims at always finding a most violated facet-defining inequality for the two-row model. This can be achieved by optimizing over the polar set of the integer hull of the model. A fast way of performing this is provided, by means of a polyhedron that is equivalent to the polar for that purpose, but has a more compact representation. Moreover, a row-generation algorithm is developed in order to avoid the costly computations of integer hulls of two-dimensional cones. An implementation of the resulting algorithm performs separation of two-row cuts in a few milliseconds on average, on the standard MIPLIB 3 and 2003 testsets.

While this two-row separator is quick, the measurements of the computational usefulness of the cuts do not yield satisfactory results. Since all the cuts generated are facet-defining, this might suggest that the underlying two-row models are too weak. This observation prompted the second part of this thesis, an attempt to evaluate the strength of various multi-row relaxations, on small instances, using a generic separator. To that end, a separator is developed, which is able to compute facet-defining inequalities from arbitrary (yet reasonably small) mixed-integer sets. A row-generation approach is again adopted, but this time the slave part consists in the resolution of a mixed-integer problem instead of a closed-form oracle. Some interesting computational tricks are developed, in order to speedup the inherently hard computations.

The first topic, the separation of two-row cuts, is joint work with my advisor Prof. Quentin Louveaux. The second, the separation of facets from arbitrary MIPs, is joint work with Prof. Quentin Louveaux and Prof. Domenico Salvagnin from the Università degli Studi di Padova.

Acknowledgements

I would like to thank my advisor, Quentin Louveaux, for the confidence he has placed in me, and for his investment in our work. Although I started with none of the necessary background, he guided me all along the path that led to this thesis, showing impressive skill, patience and gentleness. At work, his intuition is remarkably (and almost frustratingly) accurate at indicating the right way, especially when mine leans in the opposite direction. As a person, his kindness is evident; he sincerely cares about the people around him.

Domenico Salvagnin gave me higher expectations for quality and practicality in my research. He was always willing to share his unique knowledge of the computational issues in our field. It is something that could not be found in books or journals.

My family provided an incredible support during these four years, just as they always have. We share passion and a special way of thinking. For me, they are a joy and a constant motivation to go forwards.

Doors were always open in the Montefiore Institute, and the rooms filled with passionate researchers. Professor Boigelot showed me that research is not distinguishable from fun. Many of my colleagues are former classmates and long time friends. Their company is always a pleasure.

I would like to thank Carla for her cheerful presence, Philippe, Marc, Daniel, Didier and Laurent for their precious advice, François for our atypical conversations, and Denis and Julien who, without knowing it, showed tremendous skill at understanding my research.

Chapter 1

Introduction

Mathematical programming is a subfield of applied mathematics that deals with the solution of optimization problems. In particular, we focus here on *discrete* optimization problems, i.e. problems where some variables are constrained to take integer values. This covers a very broad class of problems with a wide range of applications, particularly in the field of operations research. Some special subclasses of discrete problems have enjoyed sizable fame both in terms of detailed theoretical study, and in terms of successful applications. Let us cite the *traveling salesman problem*, *set packing/cover/partitioning* problems (and their graph derivatives *matching*, *vertex packing*, *maximum clique*, *node covering*, etc.), *knapsack*, *Steiner tree*, *facility location*, *lot-sizing* and *bin packing*, among plenty of others.

To better illustrate the concept of discrete optimization, let us start with an example that is tightly linked to real-life applications.

Problem 1 (Boolean logic minimization)**.** *Let f be a Boolean function of n Boolean variables $x_1, \ldots, x_n$, given by its truth table. A disjunctive normal form (DNF) is a sum-of-products formula where variables may appear complemented, e.g. $x_1\bar{x}_2 + x_1\bar{x}_3x_4 + x_2\bar{x}_5$. Find the lightest DNF describing $f(x)$, if the weight is given by the sum of the weights of its product terms, and the weight $w(g)$ of any product term g is given.*

One straightforward way to tackle Problem 1 is to first list all *implicants* of f. An implicant of f is a product-term $g(x)$ that is true only if $f(x)$ is true, e.g. $x_2\bar{x}_5$. In other words, "g is an implicant of f" means that "$g(x)$ is true" implies "$f(x)$ is true". We denote by G the set of all such implicants, which has cardinality at most 3^n, and by $G(x)$ the subset of them that are true for x, i.e. $G(x) := \{g \in G : g(x) \text{ is true}\}$. We can now look for the lightest subset of G such that at least one of its elements is true whenever $f(x)$ is true.

$$\begin{aligned} \min \quad & \sum_{g \in G} z_g w(g) \\ \text{s.t.} \quad & \sum_{g \in G(x)} z_g \geq 1 \quad \text{for all } x : f(x) \text{ is true} \\ & z \in \{0,1\}^{|G|} \end{aligned} \tag{1.1}$$

Letting z^* be an optimal solution to (1.1), a solution to the problem is given by taking in the sum every implicant g such that $z^*_g = 1$.

The process of building (1.1) is known as *modeling*, and (1.1) is called a *formulation* of Problem 1. There are three main approaches to solving this formulation.

First, one can study the theory of the problem in depth and design a custom algorithm for solving it from scratch. In particular, (1.1) is known as a *set covering* formulation, and one can find a substantial body of theory on how to solve it approximately or exactly. As a side note, this provides strong clues suggesting that Problem 1 is hard, as set covering is one of Karp's 21 $\mathcal{NP}$-complete problems, while the size of the formulation (1.1) is already exponential in n.

A second possibility is to rewrite the formulation (1.1) using *constraint programming* (CP) techniques. Constraint programming lets us describe a feasible region using an extensive set of high-level constraints such as `alldifferent`, `atmost` or `element`. For example, a constraint `alldifferent`(y_1, y_2, y_3) means that the three variables y_1, y_2 and y_3 must take distinct (typically integer) values. The description is then fed into a CP solver which essentially searches for solutions by enumeration on the space of the variables. Each type of CP constraint comes with dedicated algorithms for efficiently pruning parts of the search space, in order to speedup the enumeration. Because of this approach, constraint programming is especially suited for highly combinatorial feasibility problems. Indeed, although any optimization problem can be reduced to a feasibility problem in polynomial time through dichotomy on the objective function value, in practice, obtaining optimality guarantees is often a tedious task for CP solvers. Note that from a computer science perspective, the CP description can be seen as the actual implementation of the solution-finding algorithm, in a declarative language that is then interpreted by the CP solver. One important characteristic of constraint programming is that plenty of different CP formulations can be valid for a single problem, and writing an efficient one requires some insight into the problem structure, although less than with an ad-hoc method as exposed previously. We will not delve deeper into the constraint programming paradigm as it extends far beyond the scope of this text, instead referring the interested reader to [5, 70, 62] for a comprehensive introduction.

The third approach consists in feeding (1.1) directly into a general-purpose solver, whose mission is to find an optimal solution. More precisely, one passes only the data A, b, c, l, u and J to a solver that solves a generic-form problem such as

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & A\,x \geq b \\ & l \leq x \leq u \\ & x \;\in \mathbb{R}^n \\ & x_j \in \mathbb{Z} \qquad \forall j \in J. \end{array} \tag{1.2}$$

In our case, $J = \{1, \ldots, |G|\}$, $l = (0, \ldots, 0)$ and $u = (1, \ldots, 1)$, i.e. all variables are integer constrained and in the range $[0, 1]$; $c = \big(w(G_1), \ldots, w(G_{|G|})\big)^T$ gives the costs associated to the variables, each variable representing one implicant, $b = (1, \ldots, 1)^T$ and A is the node-node adjacency matrix of a bipartite graph (X, G). The node set X contains one node for every x such that $f(x)$ is true, and the node set G contains a node for every implicant, with an edge between an implicant g and an input x if $g(x)$ is true.

This third approach is often referred to as *black-box* optimization, because no information about the structure of the problem is available to the solver, apart from A, b, c, l, u and J. For example, the fact that all variables are binary is only represented by the conjunction of the integrality of the variables, the lower bound 0 and the upper bound 1. Less trivially, the simple fact that

for every x such that $f(x)$ is true, at least one selected implicant must be true

is represented by a complex combination of constraints, namely, (a) all variables are binary and (b), letting a_i be a row of A, $a_i^T x \geq 1$ where a_i is a binary vector representing $G(x)$.

At first glance, the black-box approach may seem too general to be useful or efficient. Yet on a significant portion of common instances, it nevertheless outperforms other approaches at finding optimal solutions. This is mostly due to the particular methods adopted by black-box solvers, based on solving easier relaxations of the feasible region. Roughly, if such relaxations are "meaningful" for the original problem, then the black-box approach has a chance to be successful. Also, the lack of a deep analysis of the problem (and the efficient customized techniques it can yield) is often offset by the raw amount of effort that is put into perfecting the efficiency of general-purpose solvers. Moreover, black-box solvers have grown increasingly capable of exploiting (and sometimes even detecting) some forms of problem structure, bridging the gap with tailor-made algorithms.

The modeling approach adopted to write the formulation (1.1) is central to the field of *mathematical programming.* The covering problem is a typical subject studied in the subfield of *discrete optimization*, which enjoys a privileged position at the crossroads of numerical analysis, polyhedral theory, and graph theory.

This thesis covers specific techniques that aim at improving the performances of general-purpose black-box solvers for problems in a generic form such as (1.2). These techniques are known as *cutting planes* methods.

The next few sections serve the dual purpose of laying out some historical context, while formally introducing the basic theory needed for this discussion.

1.1 Optimization

Optimization problems deal with minimizing or maximizing a function of many variables, subject to many constraints on these variables. Let $f : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}$ be scalar functions of the variable vector x for $i \in \{1, \dots, m\}$, a general form of an optimization problem is

$$\begin{array}{rll} \max & f(x) & \\ \text{s.t.} & g_1(x) & \geq 0 \\ & \dots & \\ & g_m(x) & \geq 0 \\ & x \in \mathbb{R}^n. & \end{array} \tag{1.3}$$

The set $P := \{x \in \mathbb{R}^n \, : \, g_i(x) \geq 0, \, \forall i\}$ is called the feasible region of the problem, and the function $f(x)$ its objective function. A feasible solution $\bar{x}$ of (1.3) is an element of its feasible region P. A feasible solution $x^* \in P$ is optimal if for all $\bar{x} \in P$, $f(\bar{x}) \leq f(x^*)$. The optimal set $P^* := \{x \in P \, : \, f(\bar{x}) \leq f(x), \, \forall \bar{x} \in P\}$ denotes the set of all optimal solutions. When P^* is a singleton, the optimal solution is said to be unique.

The problem (1.3) is feasible if $P \neq \emptyset$, infeasible otherwise. A feasible problem is called unbounded when, for any $K > 0$, there exists $\bar{x} \in P$ such that $f(\bar{x}) > K$, which can be roughly interpreted as the "optimal" objective function value being $+\infty$.

It is important to note that, unless further hypotheses are made, optimization problems are, in general, *unsolvable* [68]. Indeed, one can easily construct an optimization problem of the form (1.3) that is arbitrarily hard to solve. This leads us to consider special subsets of the problems of the form (1.3) that are tractable computationally and for which one can prove complexity bounds.

1.2 Linear Programming

A linear optimization problem is one in which both the objective function and the constraints are linear in the variables. It can thus be written in matrix form as

$$\begin{array}{rl} \max & c^T x \\ \text{s.t.} & A\,x \geq b \\ & x \in \mathbb{R}^n \end{array} \tag{1.4}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. For practical reasons, we will further assume throughout this text that all variables and coefficients are rational, i.e. $x_j, c_j, b_i, A_{ij} \in \mathbb{Q}$, for all i, j.

One very important property of the feasible region $P := \{x \in \mathbb{R}^n : Ax \geq b\}$ of (1.4) is that it is a polyhedron in $\mathbb{R}^n$, hence the strong ties between linear optimization and polyhedral theory. Before we delve into more detail (see Section 1.5 for the basics of polyhedral theory), we can already state one interesting and fairly intuitive consequence of these ties: If the polyhedron P has one or more vertices, the optimal set P^* of (1.4) contains at least one such vertex. In other words, there exists an optimal solution to (1.4) that is a vertex of its feasible region (note that through a simple reformulation, we can assume without loss of generality that $P \subseteq \mathbb{R}^n_+$, ensuring that P has at least one vertex, unless it is empty). We can thus restrict ourselves to these vertices in the search for $\max\{c^T x : x \in P\}$, and this idea is put into practice in the simplex method [32] which, albeit having exponential worst-case complexity, is still one of the fastest algorithms for solving linear optimization problems.

Moreover, linear optimization admits algorithms of weakly polynomial time complexity [56], i.e. algorithms which find an approximation $\bar{x}$ of an optimal solution x^* such that $||f(\bar{x}) - f(x^*)|| \leq \epsilon$ in $O(\log(\frac{1}{\epsilon})\ p(nE))$ iterations, where $p(nE)$ is a polynomial in the encoding length nE of the problem [68].

The foundations of linear optimization mostly originate from polyhedral theory, and were laid out in the late 19th century by Farkas and Minkowski (see, e.g., [40, 65]). In its contemporary form however, the linear optimization problem was formulated by Kantorovich in the 1939 paper "Mathematical methods in the organization and planning of production" [54]. The term "linear *programming*" spawns from this view of optimization as a tool for production and planning. The simplex method was published by Dantzig in 1951 [32] and stayed the most efficient method for solving linear problems, both in theory and in practice, until the late 1970's. The first polynomial-time algorithm for linear programming, based on the ellipsoid method, was published in 1979 by Khachiyan [56], and was quickly followed by the first practically competitive polynomial-time method by Karmarkar [55] in 1984. Nowadays, there is no algorithm for linear programming that clearly outperforms all others, even on classes of problems of a given size. Which algorithm is faster for solving a linear problem is thus highly dependent on the particular instance, and fully reliable methods for predicting it have not been developed yet [24]. We refer the reader to Todd [72] and Schrijver [71] for an excellent overview of the history and the current state of linear programming.

1.3 Mixed-Integer Linear Programming

Mixed-integer linear optimization is the main object of this thesis. Mixed-integer problems (MIPs) are a superset of linear optimization problems, in which we admit, in addition to linear constraints, integrality constraints on some or all of the variables. A general form is given by

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & A\,x \geq b \\ & x \ \in \mathbb{R}^n \\ & x_j \in \mathbb{Z} \qquad \forall j \in J \end{aligned} \tag{1.5}$$

where $J \subseteq \{1, \dots, n\}$. It is called a pure integer problem when $J = \{1, \dots, n\}$. Note that there is no known polynomial-time algorithm for solving (1.5). Mixed-integer programming is $\mathcal{NP}$-hard in general, and plenty of special cases have been proven to be $\mathcal{NP}$-complete (see e.g. [74]).

Note that the feasible region $P := \{x \in \mathbb{R}^n : Ax \geq b \text{ and } x_j \in \mathbb{Z} \text{ for all } j \in J\}$ is no longer a polyhedron in this case, nor even a convex set. We can however consider the convex hull $\text{conv}(P)$ of the feasible solutions which, by Meyer's theorem [64], is a polyhedron. Obviously, $P \subseteq \text{conv}(P)$ and the vertices of $\text{conv}(P)$ belong to P. Therefore, if we possess a description of $\text{conv}(P)$, e.g. $\text{conv}(P) = \{x \in \mathbb{R}^n : \bar{A}x \geq \bar{b}\}$, then the problem reduces to linear programming. Indeed, assuming as before that $P \subseteq \mathbb{R}^n_+$, there exists an optimal solution x^* to $\min\{c^T x : x \in \text{conv}(P)\}$ that is a vertex of $\text{conv}(P)$, and thus belongs to P. Since x^* is an optimal solution over $\text{conv}(P)$ which is a relaxation of P, it is also an optimal solution over P. Unfortunately, computing $\bar{A}$ and $\bar{b}$ from A and b is also $\mathcal{NP}$-hard in general.

Another set of interest in the context of MIPs is the linear relaxation P_{LP} of P, which is defined by $P_{LP} := \{x \in \mathbb{R}^n : Ax \geq b\}$ and consists in dropping the integrality constraints from P. As the term *relaxation* indicates, $P \subseteq P_{LP}$ (more precisely, $P = P_{LP} \cap \{x \in \mathbb{R}^n : x_j \in \mathbb{Z} \text{ for all } j \in J\}$), and we can easily show that $P \subseteq \text{conv}(P) \subseteq P_{LP}$.

The two fundamental approaches to solving (1.5) are *cutting planes* and *enumeration*. Cutting planes are inequalities of the form $\alpha^T x \geq \alpha_0$ that are valid for P, i.e. $\alpha^T x \geq \alpha_0$ is verified for every $x \in P$. The inequality is redundant if it is also verified for every $x \in P_{LP}$. Otherwise, it cuts off part of the polyhedron P_{LP} (hence the term), and we can construct $P^{(1)}_{LP} := P_{LP} \cap \{x \in \mathbb{R}^n : \alpha^T x \geq \alpha\}$. By linearity, any inequality that is valid for P is also valid for $\text{conv}(P)$, hence $P_{LP} \supset P^{(1)}_{LP} \supseteq \text{conv}(P)$. Letting $P^{(1)} := P^{(1)}_{LP} \cap \{x \in \mathbb{R}^n : x_j \in \mathbb{Z} \text{ for all } j \in J\}$, it is straightforward to observe that $P^{(1)} = P$, hence $P^{(1)}$ can be used as a new formulation of P. Iterating the process of finding a non-redundant valid inequality, we obtain a sequence $P_{LP} \supset P^{(1)}_{LP} \supset P^{(2)}_{LP} \supset \ldots \supseteq \text{conv}(P)$. We will see in Section 1.6 that under some assumptions, specific families of valid inequalities yield sequences $\{P_{LP}, P^{(1)}_{LP}, \dots, P^{(k)}_{LP}\}$ such that $P^{(k)}_{LP} = \text{conv}(P)$ for some finite integer k. This method of iteratively approaching $\text{conv}(P)$ is called a *pure* cutting-plane algorithm. Note that the property of non-redundancy, which yields the strict inclusion relationship $P_{LP}^{(i)} \supset P_{LP}^{(i+1)}$, can be enforced by considering at step (i) a valid inequality that is violated by one specific point $\tilde{x}$ of $P_{LP}^{(i)}$. Such an inequality is said to *separate* $\tilde{x}$, and we will see in Section 1.6 that it is common for cutting-plane generation techniques to guarantee that the cuts separate, at least, an optimal solution of the current LP relaxation $\min\{c^T x : x \in P_{LP}^{(i)}\}$.

The most typical example of the enumeration approach, is the *branch and bound* algorithm, which proceeds as follows. Let $\tilde{x}$ be an optimal solution of $\min\{c^T x : x \in P_{LP}\}$. If $\tilde{x}_j \in \mathbb{Z}$ for all $j \in J$, then it is also an optimal solution to (1.5), and the algorithm terminates. Otherwise, assume without loss of generality that $\tilde{x}_1 \notin \mathbb{Z}$ and $1 \in J$. We know that for x to be in P, either $x_1 \leq \lfloor \tilde{x}_1 \rfloor$ or $x_1 \geq \lceil \tilde{x}_1 \rceil$. We thus create two subproblems whose feasible regions are $P^l := P_{LP} \cap \{x \in \mathbb{R}^n : x_1 \leq \lfloor \tilde{x}_1 \rfloor\}$ and $P^r := P_{LP} \cap \{x \in \mathbb{R}^n : x_1 \geq \lceil \tilde{x}_1 \rceil\}$, respectively. This operation is called branching on x_1. We now have the relation $P_{LP} \supset P^l \cup P^r \supseteq P$, and look for an optimal solution to (1.5) in both P^l and P^r. This leads us to explore a so-called *branch and bound tree*, creating for example P^{lr} and P^{ll} if an optimal solution $\tilde{x}^l$ to $\min\{c^T x : x \in P^l\}$ has $\tilde{x}_2^l \notin \mathbb{Z}$ while $2 \in J$. The recursive subdivision of problems in the branch and bound tree stops (i.e. we reach leaf nodes) when one of three conditions is met. First, if a subproblem is infeasible, its feasible region is empty and we do not need to further consider that node. Secondly, if an optimal solution $\tilde{x}$ of a subproblem has $\tilde{x}_j \in \mathbb{Z}$ for all $j \in J$, then it is feasible for (1.5) and no further branching is necessary. If it is the first such solution found, it becomes the so-called *incumbent* solution $\underline{x}^*$. Otherwise, it is compared to the incumbent solution, whose place it takes if it has a better objective function value, i.e. if $c^T\tilde{x} > c^T\underline{x}^*$. At termination of the algorithm, when all nodes of the tree have been explored, the incumbent solution is proven optimal for the problem. Thirdly, if the objective function value of an optimal solution $\tilde{x}$ to the subproblem is no better than that of the incumbent solution, i.e. if $c^T\tilde{x} \leq c^T\underline{x}^*$, then the node is discarded. In that case indeed, as any further branching amounts to restricting the feasible region of the current node, the optimal objective function value of any problem in the subtree of that node can only reach $c^T\tilde{x}$, at best. Occurrences of this third condition and the consequent action of discarding the corresponding node is commonly referred to as *pruning* of the branch and bound tree. In practice, pruning is extremely important for the efficiency of branch and bound, which motivates our search for good incumbents (with high $c^T\underline{x}^*$) and strong LP relaxations (yielding low $c^T\tilde{x}$).

The pure cutting plane approach was first devised by Gomory in 1960 [46, 47, 48], along with a type of cuts that ensures finiteness of the algorithm in some important specific cases, now known as the Gomory fractional cuts. Despite the theoretical elegance of the approach, initial implementations of pure cutting planes techniques were plagued with numerical difficulties. As a consequence, general-purpose cutting planes such as the Gomory cuts were widely believed through the seventies and eighties to be hopelessly useless for solving MIPs. Only custom-tailored families of cutting planes were used successfully for solving particular problems such as the traveling salesman problem, exploiting knowledge about the special polyhedral structure of the problem.

The branch and bound method on the other hand was developed also in 1960 by Land and Doig [57], and has since been the basic method of choice for solving large-scale general MIPs. In the mid-nineties however, Balas, Ceria and Cornuéjols [9] achieved tremendous improvements to the method by incorporating Gomory cutting planes in a branch and bound framework. Their approach differed from previous attempts in two main aspects. First, instead of adding one cut at a time, they included all the easily computable Gomory cuts to the formulation altogether. Secondly they switched to branch and bound after few pure cutting plane iterations, thus limiting the adverse effect of floating-point computations. In this perspective, cutting planes serve the purpose of strengthening the initial linear programming relaxation, before branch and bound is initiated. Further refinements were later brought to this hybrid approach, called *branch and cut*, such as the use of cuts within the branch and bound tree [10]. These discoveries have led to a renewed interest in general-purpose cutting planes with, as a broad objective, generating different cuts, in addition to Gomory cuts, to further strengthen the linear programming relaxations. We refer to [30] for an interesting read on this line of research and its recent history.

1.4 Subject of this thesis

The topics of this thesis are about the generation of cutting planes, with the objective of further improving the performances of branch and cut solvers. All state-of-the-art solvers already incorporate a number of cutting-plane generation techniques, such as Gomory's method. In practice, the most effective ones are computed in closed-form from a single row (i.e. one constraint) of the formulation, considering also the integrality constraints on the variables. This thesis explores more complex approaches (in particular multi-row approaches) that yield stronger inequalities, but present several important limitations. The increased complexity leads to slower cut generation and, more importantly, introduces new parameters that are not yet fully understood. Contrary to what the intuition suggests, these additional degrees of freedom pose a challenge, both from a theoretical and a practical standpoint. Indeed, while the use of cutting planes may decrease the size of a branch and bound tree by making the LP formulation stronger, it also increases the computational effort required to solve each node of the tree, as the LP formulation also grows bigger in terms of number of variables and constraints. This means that we can not just add cuts massively, but instead need to choose the right parameters and pick the cuts carefully. So far, computational experience has shown that simple cuts like Gomory's mixed-integer (GMI) cuts provide a good compromise. GMIs are computed with a closed-form formula, and have acceptable numerical properties. Furthermore, they are effective at reducing the size of the enumeration tree, and only one cut can be computed from any given linear constraint.

However, if multi-row cuts have not proved useful in practice yet, this may be due to our limited understanding about how to apply them. In this thesis, we tackle the problem in two ways. The first is straightforward, as we develop a fast generator for multi-row (specifically, two-row) cuts. The second is to implement a slow but general-purpose computational tool that we use to test the efficacy of any cut generating framework, on small benchmark MIP instances. Our objective is to predict, extrapolating from our testset, which frameworks are computationally promising, before we try to develop fast algorithms for them.

In order to describe more specifically the issues that are tackled here, we need to provide some theoretical context. In the next section, we review very briefly a few elements of polyhedral theory that are necessary for our developments. Then, in Section 1.6 we describe two simple cut generation frameworks, and in Section 1.7, we introduce a central concept in this thesis, the intersection cut.

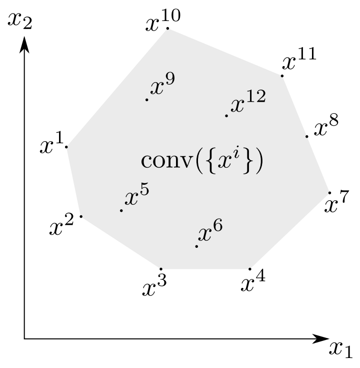

Figure 1.1: $\text{conv}(\{x^i\})$

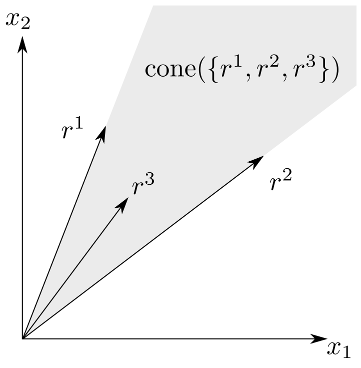

Figure 1.2: $\text{cone}(\{r^1, r^2, r^3\})$

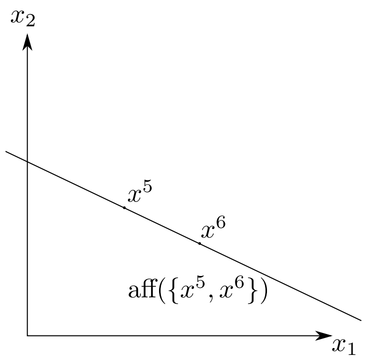

Figure 1.3: $\text{aff}(\{x^5, x^6\})$.

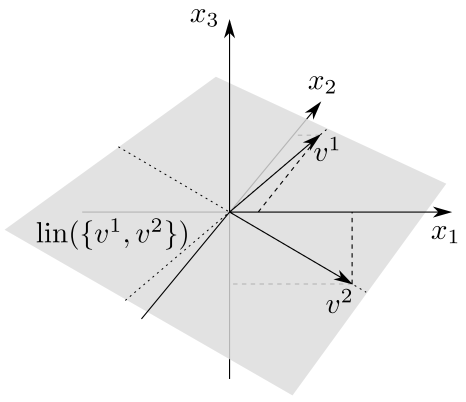

Figure 1.4: $\text{lin}(\{v^1, v^2\})$.

1.5 Polyhedral theory

While this text assumes some basic familiarity with polyhedral theory, we recall here some of its basic elements. We refer the reader to any of [21, 22, 66, 74] for a more complete introduction.

The convex, conical, affine and linear hulls are essential tools to describe the geometry of polyhedra (see Figures 1.1, 1.2, 1.3 and 1.4). They are defined here algebraically.

Definition 1. *Let X be any subset of $\mathbb{R}^n$. The convex hull* $\text{conv}(X)$ *of X is given by*

$$\text{conv}(X) := \left\{ \sum_{x \in X} \lambda_x x \;:\; \lambda_x \geq 0 \textit{ for all } x \in X, \textit{ and } \sum_{x \in X} \lambda_x = 1 \right\}.$$

Definition 2. *Let R be any subset of $\mathbb{R}^n$. The conical hull* $\text{cone}(R)$ *of R is given by*

$$\text{cone}(R) := \left\{ \sum_{r \in R} \lambda_r r \;:\; \lambda_r \geq 0 \textit{ for all } r \in R \right\}.$$

Definition 3. *Let X be any subset of $\mathbb{R}^n$. The affine hull* $\text{aff}(X)$ *of X is given by*

$$\text{aff}(X) := \left\{ \sum_{x \in X} \lambda_x x \;:\; \lambda_x \in \mathbb{R} \textit{ for all } x \in X, \textit{ and } \sum_{x \in X} \lambda_x = 1 \right\}.$$

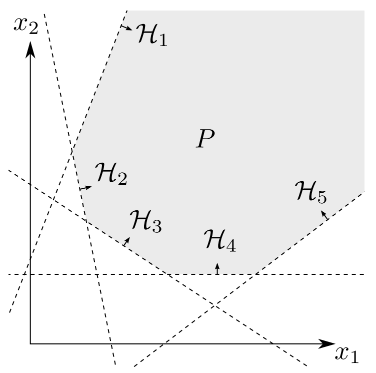

Figure 1.5: $P = \bigcap_i \mathcal{H}_i$

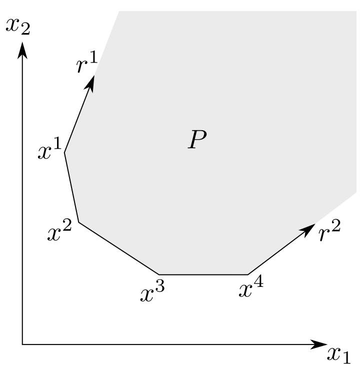

Figure 1.6: $P = \text{conv}(\{x^i\}) + \text{cone}(\{r^i\})$

Definition 4. *Let $R = \{r^1, r^2, \dots, r^k\}$ be any subset of $\mathbb{R}^n$. The linear hull (or linear span) $\text{lin}(R)$ of R is given by*

$$\text{lin}(R) := \left\{ \sum_{i=1}^{k} \lambda_i r^i \; : \; \lambda_i \in \mathbb{R}^n \textit{ for all } i \right\}.$$

The independence properties characterize the relationship between points in a set.

Definition 5 ([66])**.** *A set of points $x^1, \dots, x^k \in \mathbb{R}^n$ is linearly independent if the unique solution of $\sum_{i=1}^k \lambda_i x^i = 0$ is $\lambda_i = 0$ for $i = 1, \dots, k$.*

Definition 6 ([66])**.** *A set of points $x^1, \dots, x^k \in \mathbb{R}^n$ is affinely independent if the unique solution of $\sum_{i=1}^k \lambda_i x^i = 0$ such that $\sum_{i=1}^k \lambda_i = 0$ is $\lambda_i = 0$ for $i = 1, \dots, k$.*

We now define the concept of polyhedron. A description of a polyhedron, as provided here, in terms of half-spaces is called an *outer* description (Figure 1.5).

Definition 7. *A half-space $\mathcal{H} := \{x \in \mathbb{R}^n : \alpha^T x \geq \alpha_0\}$ is the feasible region of one inequality. If $\mathcal{H}$ is not empty, its boundary is the hyperplane $\{x \in \mathbb{R}^n : \alpha^T x = \alpha_0\}$.*

Definition 8. *A polyhedron $P := \{x \in \mathbb{R}^n : Ax \geq b\}$ is the intersection of a finite number of half-spaces.*

Definition 9. *A polytope is a bounded polyhedron, i.e. a polyhedron P such that there exists $U \in \mathbb{R}$ finite such that $|x_i| \leq U$ for all i and for all $x \in P$.*

The notion of dimension is introduced next, for any set in $\mathbb{R}^n$.

Definition 10. *A set $P \subseteq \mathbb{R}^n$ has dimension $\dim(P) = k$ if the maximal number of affinely independent points one can find in P is $k + 1$. It is comprised between -1 and n.*

In order to fully characterize a polyhedron, one needs to describe its minimal faces, extreme rays, and facets. In particular, if a polyhedron does not contain a line, its minimal faces are extreme points.

Definition 11. *An extreme point x of a polyhedron P is a point $x \in P$ such that there does not exist $y, z \in P$ such that $x \neq y$, $x \neq z$, and $x \in \text{conv}(\{y, z\})$.*

Definition 12. *A point $x \in P$ is a vertex of the polyhedron P if there exists $c \in \mathbb{R}^n$ such that $c^T x < c^T y$ for any $y \in P$ different from x.*

Proposition 1. *A point $x \in P$ is a vertex of P if and only if it is an extreme point of P.*

See e.g. [21] for a proof.

Definition 13. *A vector $r \subseteq \mathbb{R}^n$ is a ray of a polyhedron $P \subseteq \mathbb{R}^n$ if for every point $x \in P$ and every scalar $\lambda \geq 0$, $x + \lambda r$ belongs to P.*

Definition 14. *An extreme ray r of a polyhedron P is a ray of P such that there do not exist two rays s and t of P such that $s \neq \lambda r$ and $t \neq \lambda r$ for all $\lambda > 0$, and $r \in \frac{1}{2}s + \frac{1}{2}t$.*

Definition 15. *The recession cone* $\text{recc}(P)$ *of a polyhedron P is the set of all its rays:* $\text{recc}(P) := \{r \in \mathbb{R}^n : x + \lambda r \in P \text{ for all } x \in P, \lambda \geq 0\}$.

Definition 16. *The lineality space* $\text{lin.space}(P)$ *of a polyhedron P is the set* $\text{lin.space}(P) := \{r \in \mathbb{R}^n : x + \lambda r \in P \text{ for all } x \in P, \lambda \in \mathbb{R}\}$.

Note that if P does not contain a line, then $\text{recc}(P) = \text{cone}(\{r^1, \ldots, r^q\})$, where $\{r^1, \ldots, r^q\}$ are the extreme rays of P. The lineality space is a subset of the recession cone, and the three following statements are equivalent [21]:

(a). the lineality space of the polyhedron P is empty,

(b). the polyhedron P does not contain a line,

(c). the polyhedron P has at least one vertex.

A polyhedron for which these statements are true is said to be *pointed*.

Minkowski's theorem lays the foundations for the so-called *inner* description of a polyhedron, i.e. a description in terms of vertices and rays (Figure 1.6).

Proposition 2 (Minkowski's theorem)**.** *Any polyhedron P can be represented as* $P = \text{conv}(\{x^1, \ldots, x^p\}) + \text{cone}(\{r^1, \ldots, r^q\}) + \text{lin}(\{l^1, \ldots, l^t\})$ *where p, q and t are finite. In particular, if P is pointed, a minimal description is obtained by choosing $\{x^1, \ldots, x^p\}$ to be the vertices of P, $\{r^1, \ldots, r^q\}$ the extreme rays of P, and the vectors $\{l^1, \ldots, l^t\}$ to form a basis of the lineality space of P.*

See e.g. [21] for a proof.

We now cover some basic nomenclature in the theory of valid inequalities. Note that the terms *cut*, *cutting plane*, and *valid inequality* are used interchangeably in this text.

Definition 17. *The inequality $\alpha^T x \geq \alpha_0$ is valid for P if it is verified by every point of P, i.e. $\alpha^T x \geq \alpha_0$ for all $x \in P$.*

Definition 18. *The set $F := \{x \in P : \alpha^T x = \alpha_0\}$ is a face of P if $\alpha^T x \geq \alpha_0$ is a valid inequality for P. It is called* proper *if it is not empty. The inequality $\alpha^T x \geq \alpha_0$ is said to* define *the face F.*

Definition 19. *A face F of P is a facet if $\dim(F) = \dim(P) - 1$.*

Remark that extreme points of P are zero-dimensional faces of P [66].

Definition 20. *Two inequalities $\alpha^T x \geq \alpha_0$ and $\beta^T x \geq \beta_0$ are* equivalent *if there exist $\lambda > 0$ such that $\alpha = \lambda\beta$ and $\alpha_0 = \lambda\beta_0$.*

Definition 21. *Let $\alpha^T x \geq \alpha_0$ and $\beta^T x \geq \beta_0$ be two valid inequalities for $P \subseteq \mathbb{R}^n_+$. $\alpha^T x \geq \alpha_0$* dominates *$\beta^T x \geq \beta_0$ if they are not equivalent and there exist $\lambda > 0$ such that $\alpha \leq \lambda\beta$ and $\alpha_0 \geq \lambda\beta$.*

Proposition 3. *An inequality $\alpha^T x \geq \alpha_0$ is valid for a mixed-integer set P if and only if it is valid for the convex hull $\operatorname{conv}(P)$ of its solutions.*

Proof. Since the half-space $\mathcal{H}$ associated to the inequality is a convex set, $P \subseteq \mathcal{H}$ if and only if $\operatorname{conv}(P) \subseteq \mathcal{H}$. □

The following proposition, known as the separating hyperplane theorem, is exploited extensively in the proofs of Chapter 2.

Proposition 4 (Separating hyperplane theorem)**.** *Let P be a closed convex set and z a point that does not belong to P. There exists a valid inequality $\alpha^T x \geq \alpha_0$ for P such that $\alpha^T z < \alpha_0$.*

See Section 4.7 in [21] for a proof.

1.6 Two examples of simple cutting planes

To illustrate how the diverse cutting plane generation techniques work, we present two fairly intuitive ones in this section. We start with the Chvátal-Gomory rounding method.

Example 1. *Assume that we want to solve the pure integer problem $\min\{c^T x : x \in P \cap \mathbb{Z}^n_+\}$, where $\mathbb{Z}^n_+$ is a shorthand for $\{x \in \mathbb{Z}^n : x \geq 0\}$. Now let*

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \geq a_0 \tag{1.6}$$

be a known valid inequality for the feasible region $P \cap \mathbb{Z}^n_+$. For example, if P is expressed as $P := \{x \in \mathbb{R}^n : Ax \geq b\}$, then (1.6) *could be one constraint of its description, or a linear combination, i.e. $a := u^T A$ and $a_0 := u^T b$ for any $u \in \mathbb{R}^m_+$. It is straightforward to observe that*

$$\lceil a_1 \rceil x_1 + \lceil a_2 \rceil x_2 + \cdots + \lceil a_n \rceil x_n \geq a_0 \tag{1.7}$$

is also a valid inequality for $P \cap \mathbb{Z}^n_+$, as $\lceil a_1 \rceil x_1 + \cdots + \lceil a_n \rceil x_n \geq a_1 x_1 + \cdots + a_n x_n \geq a_0$ for any $x \in \mathbb{R}^n_+$. The inequality (1.7) *is said to be* weaker *than* (1.6)*. Indeed, denoting by $\mathcal{H}_1$ the feasible region of* (1.6) *and by $\mathcal{H}_2$ the feasible region of* (1.7)*, we can observe that $\mathcal{H}_1 \subseteq \mathcal{H}_2$. Therefore $\mathcal{H}_1$ provides a better* *approximation of $P \cap \mathbb{Z}^n_+$, or more rigorously, a stronger relaxation of* $\mathrm{conv}(P \cap \mathbb{Z}^n_+)$*. However,* (1.7) *has the additional property that its left-hand side takes integer values for any $x \in \mathbb{Z}^n$. The right-hand side can thus be rounded up, yielding*

$$\lceil a_1 \rceil x_1 + \lceil a_2 \rceil x_2 + \cdots + \lceil a_n \rceil x_n \geq \lceil a_0 \rceil \tag{1.8}$$

which is again a valid inequality for $P \cap \mathbb{Z}^n_+$, yet stronger than (1.7)*.*

Compared to the initial inequality (1.6)*, our cutting plane* (1.8) *could be weaker, stronger, or neither of them, depending on the values of a and a_0. Chvátal proved however in [26] that by applying this procedure a finite number of times on all the inequalities describing P (i.e. applying a finite number of* rounds *of cuts), one eventually generates all valid inequalities for $P \cap \mathbb{Z}^n_+$ [66]. Note that the rationality of P is essential for this property to hold. The rounding method was developed by Chvátal in 1973 [26], and was later found out to be equivalent to the Gomory fractional cuts developed by Gomory as early as 1958 [46, 48], although with a radically different approach.*

Examining Gomory's factional cut is worthwhile as well, since one stronger variant, the Gomory mixed-integer cut (GMI), is among the most useful cutting planes in practice (the name is somewhat misleading as not only the GMI procedure applies to mixed-integer problems, but it also yields stronger cuts in the pure integer case).

Example 2. *The development of Gomory's fractional cut assumes a slightly different problem form,* $\min\{c^T x : x \in \mathbb{Z}^n_+,\ Ax = b\}$*. Note that the various expressions of integer linear problems can easily be transformed into each other through the addition of variables and constraints. Let*

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = a_0 \tag{1.9}$$

be a known valid equality for the feasible region. Typically, (1.9) *is obtained by extracting one row from $Ax = b$, or from a simplex tableau reformulation of $Ax = b$. By rounding down the a coefficients in* (1.9)*, we obtain*

$$\lfloor a_1 \rfloor x_1 + \lfloor a_2 \rfloor x_2 + \cdots + \lfloor a_n \rfloor x_n \leq a_0 \tag{1.10}$$

which is also a valid for our feasible region. Now, the left-hand side of (1.10) *is always integral for $x \in \mathbb{Z}^n$, and we can round down the right-hand side,*

$$\lfloor a_1 \rfloor x_1 + \lfloor a_2 \rfloor x_2 + \cdots + \lfloor a_n \rfloor x_n \leq \lfloor a_0 \rfloor. \tag{1.11}$$

Finally, subtracting (1.11) *from* (1.9)*, we obtain*

$$(a_1 - \lfloor a_1 \rfloor) x_1 + (a_2 - \lfloor a_2 \rfloor) x_2 + \cdots + (a_n - \lfloor a_n \rfloor) x_n \geq a_0 - \lfloor a_0 \rfloor. \tag{1.12}$$

The inequality (1.12) *is the formula for the Gomory fractional cut. It can be applied to any valid constraint $a^T x = a_0$ for a pure integer problem, and yields a new inequality for that problem.*

There are too many methods for generating valid inequalities to detail them all here. In the next section however, we expose one more, as it is the fundamental framework that underlies all the developments in Chapter 2 and Chapter 3.

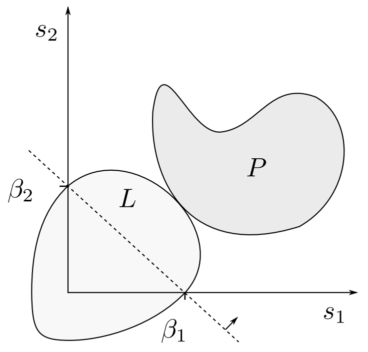

Figure 1.7: The set L in the intersection cut framework

1.7 Intersection Cuts

The concept of intersection cut, introduced by Balas in [7], is a general framework for generating cutting planes. Its nice geometric intuitiveness is better introduced with a simple example.

Problem 2. *Consider an optimization problem with a feasible region $\{s \in \mathbb{R}^n_+ : s \in P\}$, where P is an arbitrary closed set in $\mathbb{R}^n_+$ such that $0 \notin P$. Find a cutting plane* separating 0 *from P, i.e. a valid inequality $\alpha^T s \geq \alpha_0$ such that $\alpha^T 0 < \alpha_0$.*

The last statement implies that we want $\alpha_0 > 0$. Since any valid inequality can be multiplied by a positive scalar, yielding an equivalent valid inequality, we can simply fix $\alpha_0 = 1$ and look for α such that $\alpha^T s \geq 1$ for all $s \in \mathbb{R}^n_+ \cap P$.

Since $0 \notin P$ and P is closed, there must exist a bounded, closed convex set L whose interior contains 0 but no point of P, i.e. $0 \in \text{interior}(L)$ and $\text{interior}(L) \cap P = \emptyset$. Figure 1.7 illustrates such a set. The particular choice for the set L can be seen as a parameter in the intersection cut framework. Different families of sets will yield different families of intersection cuts.

Let v^i be the intersection of the ith axis with the boundary of L in the first orthant, for all i. Since v^i is on the ith axis, there exist $\beta \in \mathbb{R}^n_+$ such that $v^i = \beta_i e_i$. Consider the simplex L' defined by the convex hull of 0 and these intersection points, $L' = \text{conv}(\{0, \beta_1 e_1, \ldots, \beta_n e_n\})$. Since L' is the convex hull of points that belong to L, $L' \subseteq L$, thus L' has the same property that $\text{interior}(L') \cap P = \emptyset$. Note that L' can also be expressed as $L' = \{s \in \mathbb{R}^n_+ : \frac{1}{\beta_1} s_1 + \cdots + \frac{1}{\beta_n} s_n \leq 1\}$.

Finally, consider the open half-space $L'' := \{s \in \mathbb{R}^n : \frac{1}{\beta_1} s_1 + \cdots + \frac{1}{\beta_n} s_n < 1\}$. One can observe that all points of L'' belong to at least one of two sets:

(a) the complement set of the first orthant $\mathbb{R}^n \setminus \mathbb{R}^n_+$,

(b) the interior of L.

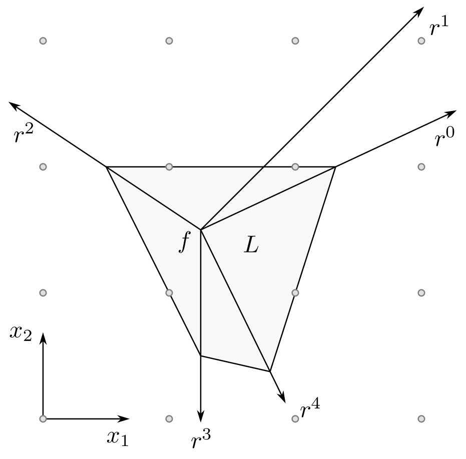

Figure 1.8: Example set L; $\alpha_i > 0$ for all i.

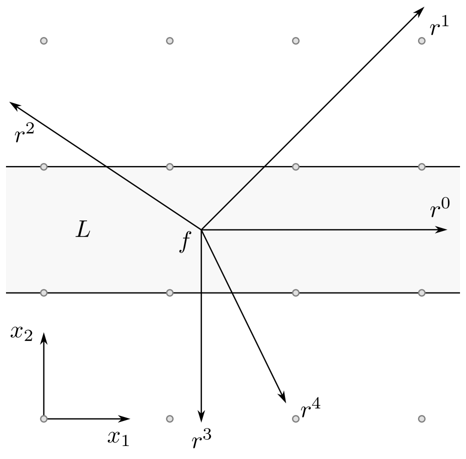

Figure 1.9: Example set L; $\alpha_0 = 0$.

Therefore, none of the points of L'' belong to P, and we conclude that

$$\frac{1}{\beta_1}s_1 + \cdots + \frac{1}{\beta_n}s_n \geq 1 \tag{1.13}$$

is a valid inequality for P.

We now adopt a more rigorous approach and develop the intersection cut in the algebraic context of integer programming. Let our feasible region be the integer set $P := \{y \in \mathbb{Z}^p_+ : Ay = b\}$, with $A \in \mathbb{R}^{m\times p}$, $b \in \mathbb{R}^m$, and $p > m$. Assume that we can construct the square nonsingular matrix A_x from a subset of m columns of A. For convenience, we write $A = [A_x|A_s]$ as if A_x covers the m *first* columns of A, although it is not necessarily the case. According to the same partition, we denote by x and s the original variables, i.e. $y^T = [x^T|s^T]$. With this notation, the feasible region can be rewritten $P = \{(x,s) \in \mathbb{Z}^m_+ \times \mathbb{Z}^n_+ : A_x x + A_s s = b\}$, with $n := p - m$. Letting $R := -A_x^{-1}A_s$ and $f = A_x^{-1}b$, and pre-multiplying both terms of the vector equality by A_x^{-1}, we finally obtain

$$P = \{(x,s) \in \mathbb{Z}^m_+ \times \mathbb{Z}^n_+ : x = f + Rs\}. \tag{1.14}$$

Although it may seem contrived, the reformulation (1.14) is perfectly natural in the context of linear programming, especially if one uses the simplex algorithm to optimize over the linear relaxation P_{LP} of P. Indeed, A_x is called a *basis* of the system $Ay = b$, and we know from linear programming theory that any vertex y^* of P_{LP} can be represented through at least one such basis by fixing $s = 0$, yielding $y^* = (f, 0)$. The x variables are thus called *basic* variables and s *nonbasic* variables. We refer to [21] for a more detailed exposition of the linear programming context.

Motivated by the form of the cut (1.13), we will prove the following theorem.

Theorem 1 (Balas [7]). *Let $L \subseteq \mathbb{R}^m$ be a closed convex set whose interior contains f but no integer point, i.e. $f \in$ interior(L) and interior$(L) \cap \mathbb{Z}^m = \emptyset$. If $\alpha \in \mathbb{R}^n_+$ is defined such that*

$$\alpha_i = \inf\{t > 0 : f + \frac{1}{t}r^i \in L\}$$

with r^i being the ith column of R, then $\alpha^T s \geq 1$ is a valid inequality for P.

Proof [28]. By the definition of α, the point $f + \frac{1}{\alpha_i} r^i$ is the intersection of the ray $f + \text{cone}(r^i)$ with the boundary of L, if such intersection exists (Figure 1.8). Otherwise, $\alpha_i = 0$ and r^i belongs to the recession cone of L (Figure 1.9). Without loss of generality, assume that $\alpha_i > 0$ for $i \in \{1, \dots, k\}$ and that $\alpha_i = 0$ for $i \in \{k+1, \dots, n\}$. Let

$$L' := \{x \in \mathbb{R}^m : x = f + r^1 s_1 + \cdots + r^n s_n,\ s \in \mathbb{R}^n_+,\ \alpha_1 s_1 + \cdots + \alpha_k s_k < 1\} \tag{1.15}$$

be the projection on the space of the x variables of the points $(x, s) \in \mathbb{R}^m \times \mathbb{R}^n_+$ that are cut off by $\alpha^T s \geq 1$. We now show that L' is a convex combination of the points $\{f + \frac{\mu}{\alpha_i} r^i : 0 \leq \mu < 1\}$ for $i \in \{1, \dots, k\}$, plus a conic combination of r^i for $i \in \{k+1, \dots, n\}$. Indeed, let $\bar{L}$ be such a set,

$$\begin{aligned}\bar{L} = \{x \in \mathbb{R}^m : x = {}& f\lambda_0 + (f + \tfrac{\mu_1}{\alpha_1} r^1)\lambda_1 + \cdots + (f + \tfrac{\mu_k}{\alpha_k} r^k)\lambda_k + \text{cone}(r^{k+1}, \dots, r^n) \\ & \lambda_0 + \lambda_1 + \cdots + \lambda_k = 1 \\ & \lambda \in \mathbb{R}^k_+,\ 0 \leq \mu_i < 1 \ \} \\ = \{x \in \mathbb{R}^m : x = {}& f + \tfrac{1}{\alpha_1} r^1 \mu_1 \lambda_1 + \cdots + \tfrac{1}{\alpha_k} r^k \mu_k \lambda_k + \text{cone}(r^{k+1}, \dots, r^n) \\ & \lambda_1 + \cdots + \lambda_k \leq 1 \\ & \lambda \in \mathbb{R}^k_+,\ 0 \leq \mu_i < 1 \ \}.\end{aligned}$$

Now letting $s_i := \frac{1}{\alpha_i} \mu_i \lambda_i$ for $i \in \{1, \dots, k\}$ and $s_i \geq 0$ for $i \in \{k+1, \dots, n\}$, we obtain

$$\begin{aligned}\bar{L} = \{x \in \mathbb{R}^m : x = {}& f + r^1 s_1 + \cdots + r^k s_k + \text{cone}(r^{k+1}, \dots, r^n) \\ & \alpha_1 s_1 + \cdots + \alpha_k s_k < 1 \\ & s \in \mathbb{R}^k_+ \ \}\end{aligned} \tag{1.16}$$

We can observe that the right-hand sides of (1.16) and (1.15) are equivalent, implying that $\bar{L} = L'$. Since L is convex and the expression (1.16) shows that any point of L' is a convex combination of points in the interior of L, we have that $L' \subseteq \text{interior}(L)$. We thus know that $L' \cap \mathbb{Z}^m = \emptyset$, therefore no point $(x, s) \in P$ satisfies $\alpha_1 s_1 + \cdots + \alpha_n s_n < 1$. So $\alpha^T s \geq 1$ is a valid inequality for P. □

A set such as L that does not contain integer points in its interior is called a *lattice-free* set. Furthermore, a lattice-free set L is said to be *maximal* if there does not exist a lattice-free set K such that $L \subset K$. Given the geometric intuition, it is natural to look for maximal lattice-free sets in order to find strong cuts, but we will come back to this topic more precisely in a later discussion.

Obviously, if $f \in \mathbb{Z}^m$, no lattice-free set could contain it in its interior, and it is impossible to apply Theorem 1 to find a valid inequality. We thus assume that there exist i such that $f_i \notin \mathbb{Z}$.

Among the simplest lattice-free sets is the split set $L^\pi := \{x \in \mathbb{R}^m : \pi_0 \leq \pi^T x \leq \pi_0 + 1\}$ for $\pi \in \mathbb{Z}^m$ and $\pi_0 \in \mathbb{Z}$. Figure 1.9 illustrates such split set if $\pi = e_2$ and $\pi_0 = 1$. It is possible to generate an intersection cut based on a split set for any $\pi \in \mathbb{Z}^m$ such that $\pi^T f \notin \mathbb{Z}$. As any intersection cut, it then separates the point $(x, s) = (f, 0)$.

One could note that the proof of Theorem 1 uses neither the fact that the basic variables are nonnegative-constrained, nor the fact that the nonbasic variables are integer-constrained. Indeed, the intersection cut is also valid for a relaxation of P where these constraints are dropped. Recall that $P = \{(x, s) \in \mathbb{Z}^m_+ \times \mathbb{Z}^n_+ : x = f + Rs\}$. Intersection cuts are valid for the set $P_I := \{(x, s) \in \mathbb{Z}^m \times \mathbb{R}^n_+ : x = f + Rs\}$. The set P_I can be seen as a further relaxation of the *corner relaxation* introduced by Gomory and Johnson [50, 51, 52], which in our case is given by $\{(x,s) \in \mathbb{Z}^m \times \mathbb{Z}^n_+ : x = f + Rs\}$. Andersen, Louveaux, Weismantel and Wolsey [4] and Cornuéjols and Margot [29] studied the case $m = 2$ and they showed that all facets of $\mathrm{conv}(P_I)$ are intersection cuts obtained from lattice-free polyhedra in $\mathbb{R}^2$ with at most four sides. This result has been generalized to $m \geq 2$ by Borozan and Cornuéjols [25], showing that the facets of $\mathrm{conv}(P_I)$ are defined by intersection cuts from lattice-free polyhedra in $\mathbb{R}^m$.

An additional type of constraint commonly occurs in mixed-integer problems: upper bounds on the variables, both basic and nonbasic. While we could reformulate problems featuring this type of constraints into the simpler form $\{y \in \mathbb{Z}^p_+ : Ay = b\}$, this is usually not done for computational reasons, as it would significantly increase the size of the problem. These upper bound constraints, too, are not taken into account when generating intersection cuts with Theorem 1.

A lot of recent research has been dedicated to strengthening intersection cuts from lattice-free sets, by reintroducing these dropped constraints. Andersen, Louveaux and Weismantel [2] considered upper bounds on nonbasic variables. Dey and Wolsey [38], Basu, Conforti, Cornuéjols and Zambelli [20] and Fukasawa and Günlük [45] considered bounds on basic variables. Dey and Wolsey [37, 33] and Conforti, Cornuéjols and Zambelli [27] considered the problem of exploiting the integrality of the nonbasic variables, the so-called lifting problem.

The latter works take place in an interesting setting introduced by Gomory and Johnson [51], the *infinite relaxation*

$$\begin{array}{rrcll} P_\infty := \{(x,s): & x & = & f + \sum_{r\in\mathbb{R}^m} r s_r & \\ & x & \in & \mathbb{Z}^m & \\ & s_r & \in & \mathbb{Z}_+ & \text{for all } r \in \mathbb{R}^m \\ & s & & & \text{has a finite support } \}, \end{array} \tag{1.17}$$

instead of P_I. Note that P_∞ is an integer set in an infinite-dimensional space. The corner relaxation $\{(x,s) \in \mathbb{Z}^m \times \mathbb{Z}^n_+ : x = f + Rs\}$ of P is given by intersecting P_∞ with $s_r = 0$ for any r that is not a column of R, then projecting onto the space of P. An intersection cut for P_∞ is typically given by a function $\psi(r) : \mathbb{R}^m \to \mathbb{R}_+$ representing the cut $\sum_{r\in\mathbb{R}^m} \psi(r) s_r \geq 1$. One property of P_∞ is that there is a direct correspondence between maximal lattice-free sets in $\mathbb{R}^m$ and undominated intersection cuts. We refer to [28] for a recent survey of this line of work.

Note that for $m = 1$, as was already pointed out by Balas [7], the lifting problem can be solved in closed form, yielding strengthened intersection cuts that turn out to be equivalent to Gomory's mixed-integer cuts. The interested reader can find more on the relationships between the major families of cutting planes in [31] and an interesting summary from a computational perspective in [73].

1.8 Two-row Cuts

Although Balas introduced the concept of intersection cut in 1971 [7], there has been a flurry of papers about related concepts in the recent years. This renewed interest was mostly spawned by a 2007 paper by Andersen, Louveaux, Weismantel and Wolsey [4] providing intuitive and elegant geometric insight into the intersection cut model for $m = 2$. Since this model is typically built from two rows of the simplex

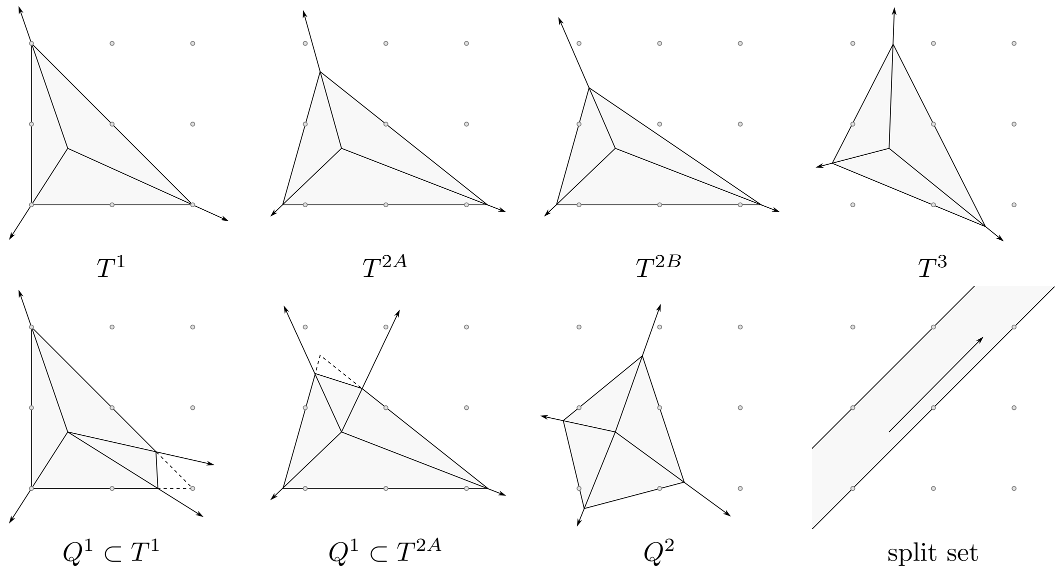

Figure 1.10: Different types of L_α

tableau, the resulting cuts are called two-row cuts. The intersection cuts originating from the same model for $m \geq 2$ are now also commonly referred to as a multi-row cuts. Albeit not strictly necessary for our further developments, we present here one result from [4] that enables most of the geometric intuition. We assume throughout this section that $m = 2$.

Let us define the polyhedron L_α in $\mathbb{R}^2$ as

$$L_\alpha := \{x \in \mathbb{R}^2 : \text{there exists } s \in \mathbb{R}^n_+ \text{ such that } x = f + Rs \text{ and } \alpha^T s \leq 1\}.$$

It corresponds to the closure of L' in the proof of Theorem 1, and is the minimal convex set yielding $\alpha^T s \geq 1$ as an intersection cut.

Theorem 2 (Andersen, Louveaux, Weismantel and Wolsey [4]). *Let $\alpha^T s \geq 1$ be a facet-defining inequality for* $\operatorname{conv}(P_I)$ *that satisfies $\alpha_j > 0$ for all j. Then L_α is a polygon with at most four vertices.*

Note that if L_α is a maximal lattice-free set and there exist j such that $\alpha_j = 0$, then L_α can only be a split set of the form $\pi_0 \leq \pi_1 x_1 + \pi_2 x_2 \leq \pi_0 + 1$ where $\pi_0 \in \mathbb{Z}$, π_1 and π_2 are coprime integers, as pictured in Figure 1.9.

Further refinements of Theorem 2 by Dey and Wolsey [33] and Dey and Louveaux [35] classify the maximal lattice-free polygons that correspond to facets of $\operatorname{conv}(P_I)$.

Proposition 5 (Dey and Louveaux [35]). *Let $\alpha^T s \geq 1$ be a facet-defining inequality for P_I. If the rays $\{r^i\}$ span $\mathbb{R}^2$ (i.e. if* $\operatorname{cone}\{r^1, \ldots, r^n\} = \mathbb{R}^2$*), then $f \in$* $\operatorname{interior}(L_\alpha)$ *and L_α is one of the following lattice-free sets:*

1. *Subset of a split set: $\{x \in \mathbb{R}^2 : \pi_0 \leq \pi_1 x_1 + \pi_2 x_2 \leq \pi_0 + 1\}$ where $\pi_1, \pi_2, \pi_0 \in \mathbb{Z}$.*

2. *Type 1 triangle (T^1): A triangle with integral vertices and exactly one integer point in the relative interior of each side.*

3. *Type 2 triangle (T^2): A triangle with at least one non-integral vertex v and the opposite side containing multiple integer points (not necessarily all in the relative interior). Let S^1 and S^2 be the two sides incident to v, and let S^3 be the third side. Then T^2 is further classified as:*

   (a) T^{2A}: S^1 and S^2 contain one integer point in their relative interior.

   (b) T^{2B}: S^1 contains one integer point in its relative interior and S^2 does not contain any integer point in its relative interior. This triangle is a subset of some triangle of type T^{2A}.

4. *Type 3 triangle(T^3): A triangle with non-integral vertices and exactly three integer points on its boundary, one in the relative interior of each side.*

5. *Type 1 quadrilateral (Q^1): A subset of T^1 or T^{2A} such that one side contains multiple integer points, two sides contain at least one integer point and the fourth side contains no integer point in its relative interior.*

6. *Type 2 quadrilateral (Q^2): A quadrilateral having non-integral vertices and containing exactly one integer point in the relative interior of each of its sides.*

The classification presented in Proposition 5 is illustrated in Figure 1.10.

1.9 Overview of the thesis

The first part of this thesis presents a fast intersection cut separator for the two-row model P_I described in Section 1.8. While other cut generators have been developed for that model [34, 18], our approach differs from these in that we perform exact separation. Specifically, given any point $x^* \in P_{LP}$, we find the facet-defining inequality for $\text{conv}(P_I)$ that is most violated at x^*, or prove that $x^* \in \text{conv}(P_I)$. This is a significant departure from previous works, which were not separators and relied on heuristics to find strong cuts. In particular, we do not adopt the "infinite relaxation" point of view and we do not generate intersection cuts from fixed lattice-free bodies.

We achieve this by optimizing over a specific polyhedron Q (the polar set of $\text{conv}(P_I)$), an approach that was suggested in [3]. Chapter 2 lays the theoretical bases for this, gathering the known results on the structure of $\text{conv}(P_I)$, and replacing them in our separation context. In Chapter 3, we first present an alternative polyhedron $\overline{Q}$ that is equivalent to Q for our purposes, but has a much more compact formulation. In practice, the size of the description of $\overline{Q}$ is linear in the number of variables, while the description of Q is quadratic. This reduction in complexity enables the implementation of a fast separator that operates by optimizing over small linear problems. The construction of these linear problems, however, requires the computation of the vertices of the integer hull of cones in $\mathbb{R}^2$. This type of computation is particularly difficult to implement safely in practice, as it mixes integer hull computations (involving large numbers and thus typically performed in exact arithmetic) with optimization models (performed in floating-point arithmetic, and featuring plenty of arbitrary tolerances). In the second part of Chapter 3, we show how to avoid this issue by performing row generation, i.e. constructing iteratively the description of $\overline{Q}$, adding the constraints only as they are needed. The main hurdle towards this is to develop a fast oracle for, given a partial description of $\overline{Q}$, finding a constraint that should be added to it. We present such an oracle in Section 3.2.

The computational results in Section 3.3 confirm that our separator is indeed fast. According to our measurements however, the resulting cuts do not strengthen the corresponding MIP formulations in a satisfactory manner. The fact that our separator is exact tends to indicate that it is the two-row model P_I itself that is weak.

There are a lot of ways to obtain stronger models, either by considering more rows, or by reinforcing the intersection cut model. The second part of this thesis tackles the question of determining which among these strengthened models present promising perspectives, computationally. In Chapter 4, we briefly survey some of the most important such models, and perform a first, very coarse evaluation of their potential. As this evaluation is not precise enough, we resort to developing a general-purpose cut separator, able to find facet-defining inequalities for arbitrary mixed-integer sets. This is the subject of Chapter 5. Obviously, such a separator could not be fast, and our objective is only to evaluate the strength of multi-row relaxations in reasonably small MIP instances. But a naive implementation would not let us conclude anything, even on the smallest problems. We present a series of computational tricks that enabled our separator to provide satisfactory results on most problems from the MIPLIB 3.0 library. Exploiting this tool, we test several variants of the intersection cut model, and are able to present a quantitative analysis of their strength.

Chapter 2

Polars of two-row models

2.1 Overview

In this chapter, we delve deeper into the polyhedral structure of the convex hull of the two-row model

$$P_I := \{(x, s) \in \mathbb{Z}^2 \times \mathbb{R}^n_+ : x = f + Rs\}$$

presented earlier. This helps us build a representation of the set of all valid inequalities for P_I. Then, we show how to exploit that expression to find facet-defining valid inequalities for P_I, and how to perform separation. Finally, we draw links between this set and the cut-generating LP (CGLP) from the lift-and-project method of Balas, Ceria and Cornuéjols [11].

While we do not present novel results in this chapter, we strive to make it a concise, self-contained introduction to the structure of the two-row model P_I, with the practical generation of cuts in mind. To that end, we define the concept of *radial polyhedron* and redevelop most of the results presented here with an intuitive perspective suiting that framework.

In our context, the set describing the valid inequalities for a polyhedron is called its *polar*. It is also a polyhedron, and we find a violated inequality by optimizing over it, with a linear objective function. Separation is thus a linear programming problem and, to perform it, we need an outer description of the polar, in terms of linear constraints. We show in Section 2.3 that we can obtain such a description from an inner description of the initial polyhedron, in our case $\text{conv}(P_I)$. To this end, we present a detailed characterization of the structure of $\text{conv}(P_I)$ in Section 2.2.

2.2 Structure of $\text{conv}(P_I)$

Let us denote by N the set $\{1, \ldots, n\}$. As noted earlier, we assume that $f \notin \mathbb{Z}^2$. Recall that we denote by r^i the ith column of R, for $i \in N$, and that we assume rational data for R and f, thus $r^i \in \mathbb{Q}^2$. Moreover, we assume without loss of generality that no two vectors r^i, r^j are parallel with the same direction. Indeed, if that were to be the case, we could aggregate the corresponding variables s_i and s_j into a single one, resulting in only one column of R.

A first basic characterization of the structure of $\text{conv}(P_I)$ is provided in Lemma 1.

Lemma 1 (Andersen, Louveaux, Weismantel and Wolsey [4]).

(i) The dimension of $\operatorname{conv}(P_I)$ *is* n.

(ii) The extreme rays of $\operatorname{conv}(P_I)$ *are* (r^i, e_i) *for all* $i \in N$.

(iii) The vertices (x^I, s^I) *of* $\operatorname{conv}(P_I)$ *take the following two forms:*

(a) $(x^I, s^I) = (x^I, s_i^I e_i)$, *where* $x^I = f + s_i^I r^i \in \mathbb{Z}^2$. x^I *is an integer point on the ray* $\{f + s_i r^i : s_i \geq 0\}$.

(b) $(x^I, s^I) = (x^I, s_i^I e_i + s_j^I e_j)$, *where* $x^I = f + s_i^I r^i + s_j^I r^j \in \mathbb{Z}^2$. x^I *is an integer point in the cone* $f + \operatorname{cone}(r^i, r^j)$.

Proof (Andersen, Louveaux, Weismantel and Wolsey [3]). Let $(\bar{x}, \bar{s})$ be an arbitrary point of P_I. For any $i \in N$, since $r^i \in \mathbb{Q}^2$, there exists a positive integer q_i such that $q_i r^i$ is integer. (i): Since x is determined unequivocally for any value of s, $\operatorname{conv}(P_I)$ is at most of dimension n. Moreover, the $(n+1)$ points $(\bar{x}, \bar{s})$ and $\{(\bar{x} + q_i r^i, \bar{s} + q_i e_i)\}$ for all $i \in N$ are in P_I and affinely independent. (ii): We have $f + \sum_{i \in N} \bar{s}_i r^i + k_i q_i r^i \in \mathbb{Z}^2$ if $k \in \mathbb{Z}_+^n$. This proves that (r^i, e_i) is a ray of $\operatorname{conv}(P_I)$, for all $i \in N$. In addition, since $\operatorname{conv}(P_I) \subseteq \mathbb{R}^2 \times \mathbb{R}_+^n$, every other ray of $\operatorname{conv}(P_I)$ can be expressed as a conic combination of these rays. (iii): If $(\bar{x}, \bar{s})$ is a vertex of $\operatorname{conv}(P_I)$, then $\bar{x}$ is integer, and $\bar{s}$ is a basic solution to the system $\{s \in \mathbb{R}_+^n : \sum_{i \in N} r^i s_i = \bar{x} - f\}$. □

Pushing further on the point (iii) of Lemma 1, we can establish a relationship between the vertices of $\operatorname{conv}(P_I) \subseteq \mathbb{R}^n$ and the vertices of the integer hull of some cones in $\mathbb{R}^2$. Lemma 2 establishes this relationship, enabling an intuitive representation of the vertices of $\operatorname{conv}(P_I)$ in the plane $(x_1, x_2) \in \mathbb{R}^2$.

Lemma 2. *The vertices* (x^I, s^I) *of* $\operatorname{conv}(P_I)$ *are of the form:*

(a) $(x^I, s^I) = (x^I, s_i^I e_i)$, *where* x^I *is the vertex of* $\operatorname{conv}(\mathbb{Z}^2 \cap f + \operatorname{cone}(r^i))$, *i.e. the integer point on the half-line* $\{f + s_i r^i : s_i \geq 0\}$ *that is closest to* f *(in terms of Euclidian distance).*

(b) $(x^I, s^I) = (x^I, s_i^I e_i + s_j^I e_j)$, *where* x^I *is a vertex of* $\operatorname{conv}(\mathbb{Z}^2 \cap f + \operatorname{cone}(r^i, r^j))$.

Proof. (a): If there exists $(x', s') = (x', s_i' e_i)$, where $x' = f + s_i' r^i \in \mathbb{Z}^2$ and $0 \leq s_i' < s_i^I$, then $(x^I, s^I) \in (x', s') + \operatorname{cone}(r^i)$, hence (x^I, s^I) is not a vertex of $\operatorname{conv}(P_I)$. (b): If x^I is not a vertex of $\operatorname{conv}(\mathbb{Z}^2 \cap f + \operatorname{cone}(r^i, r^j))$, then there must exist $x^{(1)}, \ldots, x^{(K)} \in \{Z^2 \cap f + \operatorname{cone}(r^i, r^j)\}$ such that $x^I = \sum_k \lambda_k x^{(k)}$ and $\sum_k \lambda_k = 1$. Let $s^{(1)} = s_i^{(1)} e_i + s_j^{(1)} e_j$ be such that

$$\begin{pmatrix} s_i^{(1)} \\ s_j^{(1)} \end{pmatrix} = \left[\begin{array}{c|c} r_1^i & r_1^j \\ r_2^i & r_2^j \end{array} \right]^{-1} \begin{pmatrix} x_1^{(1)} - f_1 \\ x_2^{(1)} - f_2 \end{pmatrix}.$$

Since $\operatorname{cone}(r^i) \neq \operatorname{cone}(r^j)$, the inverse exists, and as $x^{(1)} \in f + \operatorname{cone}(r^i, r^j)$, we have that $s_i^{(1)}, s_j^{(1)} \geq 0$ so $(x^{(1)}, s^{(1)}) \in P_I$. Defining $s^{(2)}, \ldots, s^{(K)}$ similarly, we have $(x^{(k)}, s^{(k)}) \in P_I$ for all k. By linearity, $s^I = \sum_k \lambda_k s^{(k)}$, thus $(x^I, s^I) = \sum_k \lambda_k (x^{(k)}, s^{(k)})$, hence (x^I, s^I) is not a vertex of $\operatorname{conv}(P_I)$. □

While all vertices of $\text{conv}(P_I)$ have the form given in Lemma 2, not all points of that form are necessarily vertices of $\text{conv}(P_I)$. However, points of that form are all, by construction, feasible for $\text{conv}(P_I)$. We can therefore build an expression of $\text{conv}(P_I)$ using the convex hull of these points and $\text{cone}\{r^i : i \in N\}$. In order to express this in a compact form, we first introduce some notation.

Given a pair of indices (i, j), C_{ij} denotes the conic polyhedron with apex f and two extreme rays r^i and r^j.

Definition 22. $C_{ij} := \{x \in \mathbb{R}^2 : x = f + r^i s_i + r^j s_j,\ s_i, s_j \geq 0\}$.

Definition 23. *$\mathcal{X}_{ij}$ is the set of vertices of* $\text{conv}(C_{ij} \cap \mathbb{Z}^2)$.

Note that the set $\mathcal{X}_{ij}$ is the set of vertices of its own convex hull $\text{conv}(\mathcal{X}_{ij})$, i.e. every element of $\mathcal{X}_{ij}$ is a vertex of $\text{conv}(\mathcal{X}_{ij})$. Also, $\text{conv}(\mathcal{X}_{ij}) = \text{conv}(C_{ij} \cap \mathbb{Z}^2)$ is called the integer hull of C_{ij}.

Once we fix i and j, every point $x \in \mathbb{R}^2$ has a unique representation as $x = f + r^i s_i + r^j s_j$. If r^i and r^j are not parallel, that representation is given by $(s^i\ s^j)^T = [r^i | r^j]^{-1}(x - f)$.

Definition 24. *Let $x, r^i, r^j \in \mathbb{R}^2$. We define $s^x_{i,j}$ and $s^x_{j,i}$ to be such that*

$$x = f + s^x_{i,j}\, r^i \ + \ s^x_{j,i}\, r^j.$$

These values exist and are unique unless $r^i = \nu r^j$, for some $\nu \in \mathbb{R}$.

Observe that $s^x_{i,j}, s^x_{j,i}$ exist and $s^x_{i,j}, s^x_{j,i} \geq 0$ if and only if $x \in C_{ij}$. The following definition is similar except that it deals with the vector r^j.

Definition 25. *Let $r^i, r^j, r^k \in \mathbb{R}^2$. We define $\lambda^j_{i,k}$ and $\lambda^j_{k,i}$ to be such that*

$$r^j = \lambda^j_{i,k}\, r^i \ + \ \lambda^j_{k,i}\, r^k.$$

These values exist and are unique unless $r^i = \nu r^k$, for some $\nu \in \mathbb{R}$.

Finally, we can express $\text{conv}(P_I)$ in terms of a convex hull and its extreme rays,

$$\text{conv}(P_I) = \text{conv}\{(x, s) : x \in \mathcal{X}_{ij},\ s = s^x_{i,j} e_i + s^x_{j,i} e_j,\ i, j \in N\} + \text{cone}\{(r^i, e_i) : i \in N\}.$$

2.3 Polarity

In the context of optimization, the term *polar* is most commonly used to denote a set describing all the valid inequalities of a polyhedron. Let P be a polyhedron. In all generality, the polar of $P \in \mathbb{R}^n$ could be defined as

$$Q := \{(\alpha, \alpha_0) \in \mathbb{R}^n \times \mathbb{R} : \alpha x \geq \alpha_0 \text{ for all } x \in P\}.$$

Since we are interested in valid inequalities for $P = \text{conv}(P_I)$, which features particular properties, we derive here a specific polar for a family of polyhedra that includes $\text{conv}(P_I)$. For the sake of conciseness, we call them *radial* polyhedra. This polar is tightly related to the 1-polar in Nemhauser and Wolsey [66], the latter applying to full-dimensional polytopes.

Definition 26. *We call a polyhedron P* radial *if*

(a) P is not empty,

(b) P does not contain a line,

(c) P does not contain the origin 0, and

(d) for every $x \in P$, $\mu x \in P$ for all $\mu \geq 1$.

Remark that the condition (d) could alternatively be written $P = P + \text{cone}(P)$, or $P \subseteq \text{recc}(P)$.

We showed earlier that the dimension of $\text{conv}(P_I)$ is n, as is the dimension of its projection on the space of the s variables. There is thus a one-to-one relationship between the valid inequalities for $\text{conv}(P_I)$ and the valid inequalities for that projection $\text{proj}_s(\text{conv}(P_I))$. More precisely any valid inequality $\beta^T x + \alpha^T s \geq \gamma$ for $\text{conv}(P_I)$ can reformulated as

$$(\alpha^T + \beta^T R)s \geq \gamma - \beta^T f$$

by eliminating the coefficients for x, yielding a valid inequality for $\text{proj}_s(\text{conv}(P_I))$. Conversely, any valid inequality $\alpha^T s \geq \alpha_0$ for $\text{proj}_s(\text{conv}(P_I))$ is a valid inequality for $\text{conv}(P_I)$. Note that $0 \notin \text{proj}_s(\text{conv}(P_I))$, $\text{proj}_s(\text{conv}(P_I)) \subseteq \mathbb{R}^n_+$, and the recession cone of $\text{proj}_s(\text{conv}(P_I))$ is $\mathbb{R}^n_+$, so $\text{proj}_s(\text{conv}(P_I))$ is radial.

Through normalization of the right-hand side, valid inequalities can be divided in three classes: $\alpha^T x \geq 1$, $\beta^T x \geq 0$ and $\gamma^T x \leq 1$. Proposition 6 lets us dismiss the latter class for radial polyhedra.

Proposition 6. *Let P be a radial polyhedron. Every facet-defining inequality of P is of the form $\alpha^T x \geq 1$, $\alpha \in \mathbb{R}^n$ or $\beta^T x \geq 0$, $\beta \in \mathbb{R}^n$.*

Proof. Consider a facet-defining inequality of type $\gamma^T x \leq 1$. Either $\gamma^T x \leq 0$ for all $x \in P$, in which case $\gamma^T x \leq 1$ does not describe a proper face of P, or there exists $\bar{x} \in P$ such that $0 < \gamma^T \bar{x} \leq 1$. Then, $\mu \bar{x} \in P$ for all $\mu \geq 1$. In particular, choosing $\mu = \frac{2}{\gamma^T \bar{x}}$, we obtain $\gamma^T(\mu \bar{x}) = 2$, hence $\gamma^T x \leq 1$ is not a valid inequality for P. □

Furthermore, we can write a variant of the separating hyperplane theorem for radial polyhedra that involves only valid inequalities of the first class.

Proposition 7. *Given a radial polyhedron P and a point $y \notin P$, there exists a valid inequality $\alpha^T x \geq 1$ for P such that $\alpha^T y < 1$.*

Proof. By the separating hyperplane theorem, there exists a valid inequality for P that separates y. (a). If the inequality is of the form $\alpha^T x \geq 1$, then the claim is proven. (b). Assume that the inequality is of the form $\beta^T x \geq 0$. As it separates y, we know that $\beta^T y < 0$. Since $0 \notin P$, by the separating hyperplane theorem, there also exist a valid inequality separating 0. That second inequality can not be of the form $\bar{\beta}^T x \geq 0$ or $\bar{\gamma}^T x \leq 1$ as it would then not separate 0. Let $\bar{\alpha}^T x \geq 1$ be that valid inequality for P. If $\bar{\alpha}^T y < 1$ then it separates y and the claim is proven. Otherwise, $\bar{\alpha}^T y \geq 1$. We now linearly combine the two valid inequalities with the positive coefficients $\frac{\bar{\alpha}^T y}{-\beta^T y}$ and 1, yielding a third valid inequality $\tilde{\alpha}^T x \geq 1$ with $\tilde{\alpha} = \frac{\bar{\alpha}^T y}{-\beta^T y}\beta + \bar{\alpha}$. That inequality separates y since $\frac{\bar{\alpha}^T y}{-\beta^T y}\beta^T y + \bar{\alpha}^T y = 0 < 1$. (c). Assume that the inequality is of the form $\gamma^T x \leq 1$. As we have shown earlier, there does not exist $\bar{x} \in P$ such that $\gamma^T \bar{x} > 0$. Indeed, we would then have $\mu\bar{x} \in P$ for all $\mu \geq 1$. In particular, choosing $\mu = \frac{2}{\gamma^T \bar{x}}$, we would obtain $\gamma^T(\mu\bar{x}) = 2$, showing that $\gamma^T x \leq 1$ is not a valid inequality for P. Hence we can strengthen the inequality by writing $\gamma^T x \leq 0$, or equivalently $\beta^T x \geq 0$ where $\beta = -\gamma$. Using (b), we obtain a valid inequality of the desired form. □

We are now ready to write the definition of the polar of a radial polyhedron. Although we arbitrarily restrict ourselves to valid inequalities of the form $\alpha^T x \geq 1$, we will show at the end of this section why this choice preserves the generality of the definition.

Definition 27. *Let P be a radial polyhedron. The* polar *Q of P is the set of all $\alpha \in \mathbb{R}^n$ such that $\alpha^T x \geq 1$ is a valid inequality for P:*

$$Q = \left\{\alpha \in \mathbb{R}^n : \ \alpha^T x \geq 1, \ \textit{for all } x \in P\right\}.$$

We next present a description of Q in terms of vertices and extreme rays of P. This is especially handy as this is the type of description that we have of $\text{conv}(P_I)$.

Proposition 8 ([66] Proposition 5.1)**.** *Q is described by*

$$\begin{aligned} Q = \{\ \alpha \in \mathbb{R}^n : \quad & \alpha^T x^k \geq 1 \quad \textit{for all } x^k \ \textit{extreme point of } P \\ & \alpha^T r^j \geq 0 \quad \textit{for all } r^j \ \textit{extreme} \quad \textit{ray of } P\ \}. \end{aligned} \tag{2.1}$$

Proof. Let Q' denote the right-hand side of (2.1). $Q \subseteq Q'$: Suppose $\bar{\alpha} \in Q$. For every x^k extreme point of P and r^j extreme ray of P, we have $\bar{\alpha}^T(x^k + \mu r^j) \geq 1$ for all $\mu \geq 0$. This implies $\bar{\alpha}^T x^k \geq 1$ and $\bar{\alpha}^T r^j \geq 0$. Hence $\bar{\alpha} \in Q'$. $Q' \subseteq Q$: Conversely if $\alpha \in Q'$ and $x \in P$, then $x = \sum_k \lambda_k x^k + \sum_j \mu_j r^j$ for some λ, μ satisfying $\sum_k \lambda_k = 1$, $\lambda_k \geq 0$, $\mu_j \geq 0$. Hence $\alpha^T x = \sum_k \lambda_k(\alpha^T x^k) + \sum_j \mu_j(\alpha^T r^j) \geq 1$. So $\alpha \in Q$. □

Proposition 8 gives a set of constraints describing Q and we know from linear programming theory that all facet-defining inequalities for Q are part of these constraints (modulo scalar multiplication). The description may also include non-facet-defining, hence redundant, constraints. However, Proposition 9 shows that all constraints of the form $\alpha^T x^k \geq 1$, where x^k is a vertex of P, are facet-defining for Q.

Proposition 9. *The facet-defining inequalities of Q are*

(a). $\alpha^T x^k \geq 1$ for all x^k extreme point of P

(b). $\alpha^T r^j \geq 0$ for all r^j extreme ray of P such that $r^j \notin \text{cone}\{x : x$ is an extreme point of $P\}$.

Proof. (a). If P only has one vertex u, then the constraint $\alpha^T u \geq 1$ of Q separates $\alpha = 0$. None of the other constraints of Q in (2.1) separate 0, so $\alpha^T u \geq 1$ is necessary to its description, and is hence facet-defining for Q. Otherwise, let y be an arbitrary vertex of P and let $P^y = \text{conv}\{x : x \text{ is a vertex of P }, x \neq y\} + \text{recc}(P)$. Because $\text{recc}(P^y) = \text{recc}(P)$, P^y is also radial, and we denote its polar by Q^y. Obviously, $P^y \subsetneq P$, indeed $y \in P \setminus P^y$, and by Proposition 7, there exists an inequality $\bar{\alpha}^T x \geq 1$ that is valid for P^y and separates y. Thus $\bar{\alpha} \in Q^y$ while $\bar{\alpha} \notin Q$, proving that $Q^y \neq Q$. Therefore, all the inequalities of the form $\alpha^T y \geq 1$ for y extreme point of P are necessary to the description of Q, and are hence facet-defining for Q.

(b). Let $P^X = \text{conv}\{x : x \text{ is a vertex of } P\} + \text{cone}\{x : x \text{ is a vertex of } P\}$. Since P is radial, $P^X \subseteq P + \text{cone}(P)$ and as noted earlier, $P = P + \text{cone}(P)$, so $P^X \subseteq P$. Let t be an extreme ray of P such that $t \notin \text{cone}\{x : x \text{ extreme point of } P\}$, i.e. $t \notin \text{recc}(P^X)$, and let $P^t = P^X + \text{cone}\{r : r \text{ is an extreme ray of } P, r \neq t\}$. Because t is an extreme ray of P, it can not be expressed as a conic combination of other rays of P, so $t \notin \text{recc}(P^t)$. By construction, P^t is radial and we denote its polar by Q^t. Furthermore, $\text{recc}(P^t) \subseteq \text{recc}(P)$ hence $P^t \subseteq P$. Let w be an arbitrary vertex of P. As $t \notin \text{recc}(P^t)$, there exist $M \in \mathbb{R}_+$ sufficiently large such that $z = w + Mt$ does not belong to P^t, while by construction it belongs to P. By Proposition 7, there exists an inequality $\tilde{\alpha}^T x \geq 1$ that is valid for P^t and separates z. Thus $\tilde{\alpha} \in Q^t$ while $\tilde{\alpha} \notin Q$, proving that $Q^t \neq Q$. Therefore, all inequalities of the form $\alpha^T t \geq 0$ for t extreme ray of P such that $t \notin \text{cone}\{x : x \text{ vertex of } P\}$ are necessary to the description of Q, and are hence facet-defining for Q. □

One elegant property of radial polyhedra, which they share with full-dimensional polytopes [66], is a simple duality relationship between them and their polar. Proposition 10 and Proposition 11 establish this duality.

Proposition 10. *The polar Q of a radial polyhedron P is a radial polyhedron.*

Proof. (a). Since $0 \notin P$, by Proposition 7, there exist a valid inequality $\bar{\alpha}^T x \geq 1$ for P that separates 0, hence $\bar{\alpha} \in Q$, showing that Q is not empty. (b). Since P is not empty, $0^T x \geq 1$ is not a valid inequality for P, thus $0 \notin Q$. (c). By Proposition 8, Q is a polyhedron. Let $\alpha \in Q$, we know that $\alpha^T x \geq 1$ for all $x \in P$. Then for all $x \in P$, $(\mu\alpha)^T x = \alpha^T(\mu x) \geq 1$, since $\mu x \in P$. Thus $\mu\alpha \in Q$. □

Proposition 11 ([66] Proposition 5.4)**.** *The polar of Q is P.*

Proof. By Proposition 10, Q is a radial polyhedron, thus its polar can be defined as in Definition 27. Let $\overline{P} = \{y \in \mathbb{R}^n : \ y^T\alpha \geq 1, \text{ for all } \alpha \in Q\}$ be the polar of Q. If $x \in P$, then $\alpha^T x \geq 1$, for all $\alpha \in Q$. Thus $x \in \overline{P}$, so $P \subseteq \overline{P}$. Now let $y \notin P$. By Proposition 7, there exists a valid inequality $\alpha^T x \geq 1$ of P such that $\alpha^T y < 1$. Since $\alpha \in Q$, $y \notin \overline{P}$, so $\overline{P} \setminus P = \emptyset$. □

Corollary 1. *The facet-defining inequalities of P are*

(a). $\alpha^T x \geq 1$ for all α extreme point of Q

(b). $\beta^T x \geq 0$ for all β extreme ray of Q such that $\beta \notin \text{cone}\{\alpha : \alpha \text{ extreme point of } P\}$.

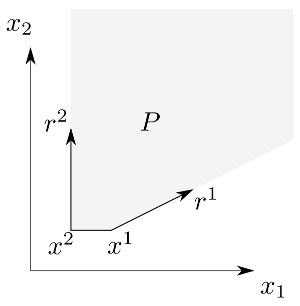

Figure 2.1: Example radial set P

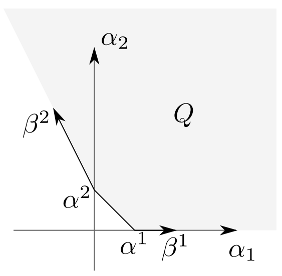

Figure 2.2: Example polar Q

The following example illustrates the properties that we established in this section.

Example 3. *Let $P \subseteq \mathbb{R}^2$ be given by (Figure 2.1)*

$$P = \operatorname{conv}(x^1, x^2) + \operatorname{cone}(r^1, r^2),$$

where

$$x^1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \; x^2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \; r^1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \; r^2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

It is easy to verify that P is radial. From the vertices and extreme rays of P immediately follows a description of its polar Q in terms of the constraints (Figure 2.2)

$$\begin{aligned} Q = \{(\alpha_1, \alpha_2) \in \mathbb{R}^2 : 2\alpha_1 + \alpha_2 &\geq 1 && (2.2)\\ \alpha_1 + \alpha_2 &\geq 1 && (2.3)\\ 2\alpha_1 + \alpha_2 &\geq 0 && (2.4)\\ \alpha_2 &\geq 0\}. && (2.5) \end{aligned}$$

By optimizing over Q, we can obtain vertices and extreme rays of Q. In our small example, we can observe that

$$Q = \operatorname{conv}(\alpha^1, \alpha^2) + \operatorname{cone}(\beta^1, \beta^2)$$

with

$$\alpha^1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \; \alpha^2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \; \beta^1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \; \beta^2 = \begin{pmatrix} -1 \\ 2 \end{pmatrix}.$$

and that (2.2), (2.3) *and* (2.5) *are facet-defining for Q while* (2.4) *is not (it is strictly dominated by* (2.2)*). Indeed, the corresponding extreme ray of P*

$$r^1 \in \operatorname{cone}(x^1, x^2).$$

Conversely, the vertices and extreme rays of Q yield a constraint description of P

$$\begin{aligned} P = \{(x_1, x_2) \in \mathbb{R}^2 : \quad x_1 \qquad &\geq 1 && (2.6)\\ x_2 &\geq 1 && (2.7)\\ x_1 \qquad &\geq 0 && (2.8)\\ -x_1 + 2x_2 &\geq 0\} && (2.9) \end{aligned}$$

where (2.6)*,* (2.7) *and* (2.9) *are facet-defining for Q while* (2.8) *is not (it is strictly dominated by* (2.6)*). Again, the corresponding ray of Q*

$$\beta^1 \in \operatorname{cone}(\alpha^1, \alpha^2).$$

We now have all the tools necessary to separate facet-defining inequalities for $\operatorname{conv}(P_I)$. Indeed, we have developed a description of $\operatorname{conv}(P_I)$ in terms of its extreme rays and (a superset of) its vertices. Proposition 8 provides us with a way to write down its polar from that description. We can thus construct valid inequalities for $\operatorname{conv}(P_I)$ from feasible points of its polar. Furthermore, Corollary 1 shows that all the vertices of the polar are facet-defining inequalities for $\operatorname{conv}(P_I)$, and all facet-defining inequalities for $\operatorname{conv}(P_I)$ are given by vertices and extreme rays of its polar.

2.4 Separation

Putting the results of the previous section to practice, the polar Q of $\operatorname{proj}_s(\operatorname{conv}(P_I))$ is given by

$$\begin{aligned} Q = \{\ \alpha \in \mathbb{R}^n :\quad & s^T\alpha \geq 1, \quad \forall (x,s) \text{ extreme point of } \operatorname{conv}(P_I) \\ & t^T\alpha \geq 0, \quad \forall (r,t) \text{ extreme ray of } \operatorname{conv}(P_I)\ \} \end{aligned} \tag{2.10}$$

or, with our notation,

$$\begin{aligned} Q = \{\ \alpha \in \mathbb{R}^n :\quad & s^x_{i,j}\alpha_i + s^x_{j,i}\alpha_j \geq 1, \quad \forall i,j \in N,\ \forall x \in \mathcal{X}_{ij}, \\ & \alpha_i \geq 0, \quad \forall i \in N \qquad\qquad \}. \end{aligned} \tag{2.11}$$

Recall however that as not all $x \in \mathcal{X}_{ij}$ are guaranteed to correspond to vertices of $\operatorname{conv}(P_I)$, the latter expression may introduce redundant (yet valid) inequalities $s^x_{i,j}\alpha_i + s^x_{j,i}\alpha_j \geq 1$.

Any $\alpha \in Q$ yields a valid inequality $\alpha^T x \geq 1$ for P. In particular, by solving the linear optimization problem

$$\begin{aligned} \min\quad & c^T\alpha \\ \text{s.t.}\quad & s^x_{i,j}\alpha_i + s^x_{j,i}\alpha_j \geq 1 \quad \forall i,j \in N,\ \forall x \in \mathcal{X}_{ij} \\ & \alpha \in \mathbb{R}^n_+ \end{aligned} \tag{2.12}$$

for any $c \geq 0$ with the simplex algorithm, we obtain a finite optimal solution α^* that is a vertex of Q and hence corresponds to a facet-defining inequality ${\alpha^*}^T x \geq 1$ of $\operatorname{conv}(P_I)$. Since that inequality is an intersection cut, it separates the fractional point $(f, 0)$.

Furthermore, we can perform separation on any arbitrary point $(\bar{x}, \bar{s}) \in \mathbb{R}^2 \times \mathbb{R}^n$ with $\bar{x} = f + R\bar{s}$, i.e. any point of the linear relaxation of P_I, by solving the linear optimization problem $\min\{\bar{s}^T\alpha : \alpha \in Q\}$. Let α^* be the optimal solution. If $\bar{s}^T\alpha^* < 1$, we found a most-violated, facet-defining valid inequality for $\operatorname{conv}(P_I)$ that separates $(\bar{x}, \bar{s})$. Otherwise, if $\bar{s}^T\alpha^* \geq 1$, it means that $\bar{s}^T\alpha \geq 1$ for all $\alpha \in Q$, meaning that no valid inequality for $\operatorname{conv}(P_I)$ separates $(\bar{x}, \bar{s})$. In other words, it means that $(\bar{x}, \bar{s}) \in \operatorname{conv}(P_I)$.

2.5 Linear programming derivation of the polar of $\operatorname{conv}(P_I)$

In this section, we develop a slightly different expression of the polar of $\operatorname{conv}(P_I)$ using a linear programming point a view. Recall that

$$P_I := \{(x,s) \in \mathbb{Z}^2 \times \mathbb{R}^n_+ : x = f + Rs\}.$$

By definition, an inequality of the form $\alpha^T s \geq 1$ is valid for P_I if $\alpha^T s \geq 1$ for all $(x, s) \in P_I$. Equivalently, $\alpha^T s \geq 1$ is valid for P_I if $\alpha^T s \geq 1$ for every vertex (x, s) of $\operatorname{conv}(P_I)$. From this, we can directly write down the polar Q of $\operatorname{conv}(P_I)$ as

$$Q = \{\alpha \in \mathbb{R}^n_+ : \alpha^T s \geq 1, \text{ for all } (x, s) \text{ vertex of } \operatorname{conv}(P_I)\} \tag{2.13}$$

and separation of $(\bar{x}, \bar{s})$ is operated by means of

$$\begin{array}{ll} \min & \bar{s}^T \alpha \\ \text{s.t.} & s^T \alpha \geq 1, \text{ for all } (x, s) \text{ vertex of } \operatorname{conv}(P_I). \end{array} \tag{2.14}$$

By expliciting the expression of P_I and using Lemma 2 to characterize the vertices of $\operatorname{conv}(P_I)$, we get

$$\begin{array}{ll} \min & \bar{s}^T \alpha \\ \text{s.t.} & s^T \alpha \geq 1, \text{ for all } x \in \mathcal{X}, \text{ for all } s \in \mathbb{R}^n_+ \text{ such that } x = f + Rs, \\ & \alpha \in \mathbb{R}^n_+ \end{array} \tag{2.15}$$

where $\mathcal{X} = \bigcup_{i,j \in N} \mathcal{X}_{ij}$. Note that $\alpha \geq 0$ comes from the expression (2.12) of Q where we can see that $Q \subseteq \mathbb{R}^n_+$. In general, in a set of constraints $a^T x \geq b$ for all $a \in G$, we can replace the left-hand side by an optimization problem and obtain a single constraint $\min\{a^T x : a \in G\} \geq b$. We do this with the "for all s" part of the constraint of (2.15) in order to obtain a linear programming problem in the left-hand side.

$$\begin{array}{ll} \min & \bar{s}^T \alpha \\ \text{s.t.} & \left(\begin{array}{ll} \min & \alpha^T s \\ \text{s.t.} & Rs = x - f \\ & s \in \mathbb{R}^n_+ \end{array} \right) \geq 1, \text{ for all } x \in \mathcal{X} \end{array} \tag{2.16}$$

This type of bilevel optimization problem is particularly hard to tackle in practice. However, since the inner problem is linear and we only use its optimal objective function value, we have by strong duality that

$$\left(\begin{array}{ll} \min & \alpha^T s \\ \text{s.t.} & Rs = x - f \\ & s \in \mathbb{R}^n_+ \end{array} \right) = \left(\begin{array}{ll} \max & (x - f)^T p \\ \text{s.t.} & R^T p \leq \alpha \\ & p \in \mathbb{R}^2 \end{array} \right)$$

(see e.g. [21] for more details on linear programming duality), yielding

$$\begin{array}{ll} \min & \bar{s}^T \alpha \\ \text{s.t.} & \left(\begin{array}{ll} \max & (x - f)^T p \\ \text{s.t.} & R^T p \leq \alpha \\ & p \in \mathbb{R}^2 \end{array} \right) \geq 1, \text{ for all } x \in \mathcal{X}. \end{array} \tag{2.17}$$

Again, in general, a constraint having an optimization problem in the left-hand side $\max\{a^T x : a \in G\} \geq b$ can be transformed into an existence condition $\exists a \in G : a^T x \geq b$. Applying this to (2.17), we obtain

$$\begin{array}{lll} \min & \bar{s}^T \alpha & \\ \text{s.t.} & (x - f)^T p_x \geq 1 & \text{for all } x \in \mathcal{X} \\ & -\alpha + R^T p_x \leq 0 & \text{for all } x \in \mathcal{X} \\ & p_x \in \mathbb{R}^2 & \text{for all } x \in \mathcal{X} \end{array} \tag{2.18}$$

We now compare the expressions (2.12) and (2.18) of the separation problem. The optimization problem (2.12) has n variables, and $\sum_{ij \in N} |\mathcal{X}_{ij}|$ constraints. The problem (2.18) on the other hand has $n + 2|\mathcal{X}|$ variables and $(n+1)|\mathcal{X}|$ constraints. As $\mathcal{X} = \bigcup_{i,j \in N} \mathcal{X}_{ij}$, it appears natural for (2.18) to be more complex in general.

But if n is large, since $\mathcal{X} \subset \mathbb{Z}^2$ is in a low-dimensional space, $\sum_{ij \in N} |\mathcal{X}_{ij}|$ may very well be much larger than $(n+1)|\mathcal{X}|$. Furthermore, in the fairly common case where the basic variables are $\{0,1\}$-constrained,

$$P_B = \{(x,s) \in \{0,1\}^2 \times \mathbb{R}^n_+ : x = f + Rs\},$$

then we only need to consider the four points $\{(0,0),(0,1),(1,0),(1,1)\}$ in $\mathcal{X}$. In that case, (2.18) will always have at most $n+8$ variables and $4n+4$ constraints.

However, the main advantage of (2.18) resides in the fact that its underlying reasoning can be extended to models stronger than P_I, like

$$P_{IU} = \{(x,s) \in \mathbb{Z}^2 \times \mathbb{R}^n_+ : x = f + Rs,\ s \le U\}$$

where we have upper bounds on the continuous variables. We easily obtain the separation problem

$$\begin{array}{lll} \min & \bar{s}^T \alpha & \\ \text{s.t.} & (x-f)^T p_x + U^T q_x \ge 1 & \text{for all } x \in \mathcal{X} \\ & -\alpha + R^T p_x + q_x \le 0 & \text{for all } x \in \mathcal{X} \\ & p_x \in \mathbb{R}^2 & \text{for all } x \in \mathcal{X} \\ & q_x \in \mathbb{R}^n, q_x \le 0 & \text{for all } x \in \mathcal{X} \end{array} \tag{2.19}$$

for P_{IU}. The problem (2.19) has $n + (2+n)|\mathcal{X}|$ variables and $(n+1)|\mathcal{X}|$ constraints. The idea can be taken further considering the k-row problem

$$P_C = \big\{(x,s) \in X \times \mathbb{R}^n_+ : A_x x + A_s s = b\big\}$$

where $X \subseteq \mathbb{Z}^m$. The resulting separation problem is

$$\begin{array}{lll} \min & \bar{s}^T \alpha & \\ \text{s.t.} & (b - A_x x)^T p_x \ge 1 & \text{for all } x \in X \\ & -\alpha + A_s^T p_x \le 0 & \text{for all } x \in X \\ & p_x \in \mathbb{R}^k & \text{for all } x \in X. \end{array} \tag{2.20}$$

In this case the number k of rows of $[A_x|A_s]$ can be greater than the number m of integer-constrained variables. We are thus no longer in the context of intersection cuts. Assuming that $|X|$ is finite, the linear program (2.20) has a block-angular structure, where each block correspond to one value of x. Furthermore, it corresponds to the cut-generating linear program (CGLP) of the lift-and-project method of Balas, Ceria and Cornuéjols [11] for a multiple-term disjunction. Adopting Balas' disjunctive notation [8],

$$P_C = \bigvee_{\bar{x} \in X} \big\{(\bar{x}, s) \in \mathbb{Z}^m \times \mathbb{R}^n_+ : A_x \bar{x} + A_s s = b\big\}$$

and (2.20) is the so-called reverse polar of $\operatorname{conv}(P_C)$ [8, 14]. Note that for $m = 1$ and $|X| = 2$, Balas and Perregaard [15] characterized the bases of (2.20) in terms of the bases of the linear relaxation of P_C, providing an extremely fast way of separating facet-defining inequalities of P_C. We refer to [14] for a survey of the disjunctive approach and the lift-and-project method.

2.6 Summary

In Section 2.2, we characterized the polyhedral structure of $\text{conv}(P_I)$ in terms of its vertices and extreme rays. This characterization let us write down an exact expression of its polar in Section 2.3. And we showed in Section 2.4 how we could use the polar to separate valid inequalities for $\text{conv}(P_I)$. This lays the foundations for the fast separation algorithm that we will develop in Chapter 3. Finally, in Section 2.5, we establish the connection between a slightly different expression of the polar and the concepts used in the lift-and-project method, offering some theoretical insight on the links with that separation technique.

Chapter 3

Separation of two-row cuts

In this chapter, we tackle the separation of facet-defining inequalities for the two-row model

$$P_I = \{(x, s) \in \mathbb{Z}^2 \times \mathbb{R}^n_+ : x = f + Rs\}$$

from a computational point of view.

It is not the first body of work addressing this issue. Espinoza [39] performed some of the first computational experiments with multi-row cutting planes. His approach is to separate intersection cuts from fixed lattice-free polyhedra, and it is not limited to the two-row case. More precisely, he uses three families of maximal lattice-free polyhedra in $\mathbb{R}^m$. This is very natural in the context of the continuous infinite relaxation

$$R_f := \{(x, s) \in \mathbb{Z}^m \times \mathbb{R}^\infty_+ : x = f + \sum_{r \in \mathbb{R}^m} r s_r,\ s \text{ has a finite support}\}$$

of P_I. Indeed, Borozan and Cornuéjols [25] showed that all undominated valid inequalities of R_f are intersection cuts from maximal lattice-free polyhedra in $\mathbb{R}^m$.

More recently, Dey, Lodi, Tramontani and Wolsey [34] and Basu, Bonami, Cornuéjols and Margot [18] also tackled the question, both focusing on parametric Type-2 lattice-free triangles in $\mathbb{R}^2$ (see Proposition 5). The precise shape of the triangle is determined by a heuristic procedure in [34], and [18] concentrates on the case where one of the two components of f is integral, a common situation when the LP relaxation of the problem features primal degeneracy.

One important advantage of the methods developed in [39, 34, 18] is that they enable the computation of a multi-row cut in closed form, hence extremely quickly. However, they do not guarantee that they generate the strongest or most violated cut for P_I. In particular, the information that is lost by taking the continuous infinite relaxation of P_I can be illustrated quite intuitively. Figure 3.1a shows a maximal lattice-free triangle B_ψ containing f in its interior. Because that triangle is maximal, the intersection cut derived from it is an undominated valid inequality $\sum_{r \in \mathbb{R}^2} \psi(r) s_r \geq 1$ for R_f ($\psi : \mathbb{R}^2 \to \mathbb{R}_+$ is a so-called *minimal* valid function). But when taking into account the geometry of P_I, i.e. the specific values of the columns r^j of R, we see that we could generate the exact same valid inequality with the lattice-free triangle L_α pictured on Figure 3.1b. The latter is not maximal, showing that the corresponding intersection cut

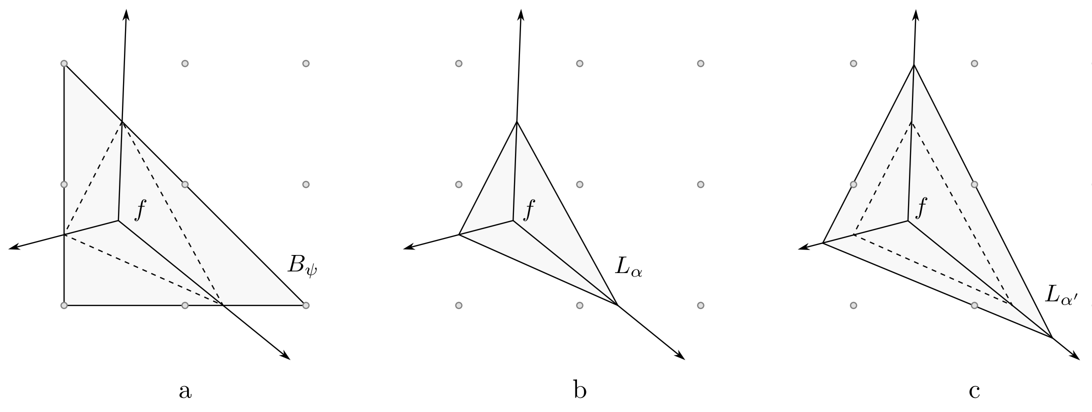

Figure 3.1: B_ψ and L_α

$\alpha^T s \geq 1$, with $\alpha_j = \psi(r^j)$ is not facet-defining of $\text{conv}(P_I)$. In this case, L_α is strictly included in the interior of a lattice-free triangle $L_{\alpha'}$ pictured in Figure 3.1c, which yields a strictly stronger intersection cut $\alpha'^T s \geq 1$.

Instead, we establish a method for separating maximally-violated intersection cuts that are facet-defining for $\text{conv}(P_I)$. Our method is based on the polar

$$Q = \left\{\alpha \in \mathbb{R}^n_+ : s^x_{i,j}\alpha_i + s^x_{j,i}\alpha_j \geq 1,\ \forall i,j \in N,\ \forall x \in \mathcal{X}_{ij}\right\}$$

of $\text{conv}(P_I)$ developed in Section 2.4 (recall that we define $s^x_{i,j}$ and $s^x_{j,i}$ to be such that $x = f + s^x_{i,j}r^i + s^x_{j,i}r^j$, and similarly $\lambda^j_{i,k}$ and $\lambda^j_{k,i}$ to be such that $r^j = \lambda^j_{i,k}r^i + \lambda^j_{k,i}r^k$). Andersen, Louveaux, Weismantel and Wolsey [3] showed that one can compute the vertices $\mathcal{X}_{ij}$ of the integer hull of a cone C_{ij} in $\mathbb{R}^2$ in polynomial time, yielding a polynomial-time separation problem for $\text{conv}(P_I)$: $\min\{c^T\alpha : \alpha \in Q\}$. However, we will see in Section 3.2 that there are significant hurdles to implement the method proposed in [3], which could make it slow in practice.

There are two main results in this chapter. The first result is to show that the complexity of the polar can be reduced from a quadratic to a linear (in n) number of integer hull computations in order to perform an exact separation. The second result is to provide an algorithm that avoids computing explicitly all integer hulls and hence to obtain a method that runs quickly in practice despite having no guaranteed polynomial running time.

The works presented in this chapter have been published in [59].

3.1 A compact representation of the polar of $\text{conv}(P_I)$

While it is possible to solve the separation problem by optimizing over the polar Q of $\text{conv}(P_I)$, we present an alternative, more compact formulation $\overline{Q}$. Indeed, to set up Q, we need to consider every pair (r^i, r^j) of rays and compute the vertices of the integer hull of the cone C_{ij} that they define (i.e. the set $\mathcal{X}_{ij}$). Every such vertex (every point in every $\mathcal{X}_{ij}$) generates one constraint of Q. On the other hand, we construct $\overline{Q}$ by considering only pairs of *consecutive* rays (r^i, r^{i+1}) and their respective $\mathcal{X}_{i,i+1}$, plus at

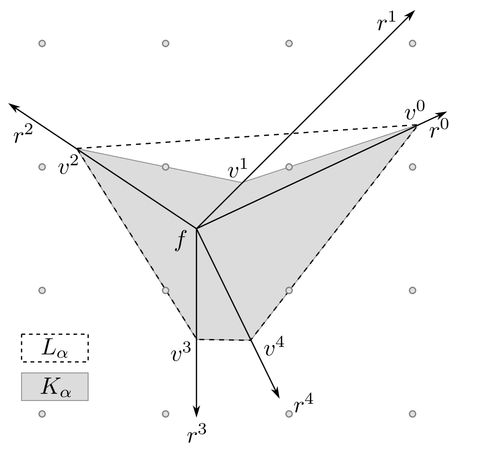

Figure 3.2: K_α is lattice-free while L_α is not

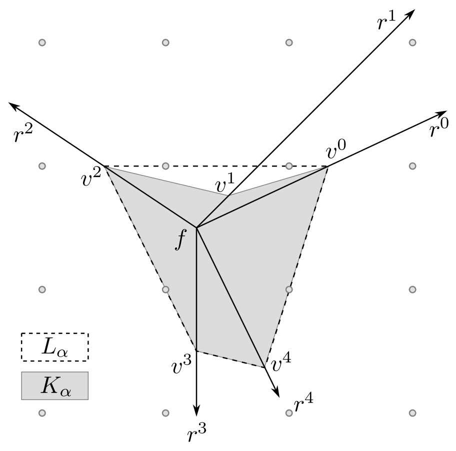

Figure 3.3: Both L_α and K_α are lattice-free, but K_α is not convex

most n constraints linking the α coefficients for triples of consecutive rays. More precisely, we consider the set

$$\overline{Q} = \{\, \alpha \in \mathbb{R}^n_+ \ : \ s^x_{i,i+1}\alpha_i + s^x_{i+1,i}\alpha_{i+1} \geq 1, \qquad \forall i,\ \forall x \in \mathcal{X}_{i,i+1}, \tag{3.1}$$

$$\alpha_i \leq \lambda^i_{i-1,i+1}\alpha_{i-1} + \lambda^i_{i+1,i-1}\alpha_{i+1},\ \forall i :\ r^i \in \text{cone}(r^{i-1}, r^{i+1})\,\}, \tag{3.2}$$

where the rays are indexed in counter-clockwise order, and modulo n (e.g. $r^{-1} \equiv r^{n-1}$). As mentioned in Section 2.2, we assume without loss of generality that no two vectors r^i, r^j are parallel with the same direction. Observe that the set Q is described by $\sum_{i=0}^{n-1}\sum_{j=i+1}^{n-1} |\mathcal{X}_{ij}|$ constraints while $\overline{Q}$ features at most $n + \sum_{i=0}^{n-1} |\mathcal{X}_{i,i+1}|$ constraints. Theorem 3 shows that optimizing over Q can be done through optimizing over $\overline{Q}$.

Theorem 3. *Let $c \in \mathbb{R}^n$, $c > 0$, the problem* $\min\{c^T\alpha \,:\, \alpha \in Q\}$ *and the problem* $\min\{c^T\alpha \,:\, \alpha \in \overline{Q}\}$ *share the same set of optimal solutions.*

Before proving Theorem 3 let us develop some geometric interpretation of its result. We already defined L_α as

$$L_\alpha := \{x \in \mathbb{R}^2 : \text{there exists } s \in \mathbb{R}^n_+ \text{ such that } x = f + Rs \text{ and } \alpha^T s \leq 1\}.$$

To simplify the discussion, we assume for the time being that $\alpha > 0$, which implies that L_α is bounded, and that cone$\{r^i : i \in N\} = \mathbb{R}^2$, so f is in the interior of L_α. Let v^i be the intersection of the boundary of L_α with the half line $f + \text{cone}(r^i)$. By Theorem 1, $v^i = f + \frac{1}{\alpha_i}r^i$, and $L_\alpha = \text{conv}(\{v^0, \ldots, v^{n-1}\})$. The geometric intuition behind the constraints of Q is that stating $\alpha \in Q$ is equivalent to stating that L_α is lattice-free. Instead, if we want $\alpha \in \overline{Q}$, we first consider only the constraints of Q that correspond to cones formed with consecutive rays. In other words, we state that $K_\alpha := \bigcup_i \text{conv}(\{f, v^i, v^{i+1}\})$ is lattice-free (Figure 3.2). Note that K_α is not necessarily convex; but we can observe that if it is, then $L_\alpha = K_\alpha$. This motivates the inclusion of the n additional constraints (3.2) in $\overline{Q}$, which enforce the convexity of K_α. Indeed, for K_α to be nonconvex, there must exist two consecutive triangles whose union is nonconvex, like conv$(\{f, v^0, v^1\})$ and conv$(\{f, v^1, v^2\})$ in Figure 3.2. The n additional constraints enforce that any v^i must be farther from f than the point in the line segment joining v^{i-1} and v^{i+1} along the half line

$f + \text{cone}(r^i)$. In the presence of such constraints, we miss some valid solutions $\alpha \in Q$, i.e. those which correspond to a nonconvex K_α (Figure 3.3). However, in every such solution, there is one $v^i = f + r^i/\alpha_i$ in the interior of L_α, and the cut can be trivially strengthened by decreasing α_i until v^i is on the border of L_α.

We prove Theorem 3 by showing that $\overline{Q}$ is a subset of Q (Lemma 4) and that every optimal solution to $\min\{c^T\alpha : \alpha \in Q\}$, is feasible for $\overline{Q}$ (Lemma 5). First, we need the following result which shows that when (3.2) holds, a similar constraint also holds for non-consecutive rays contained in the same cone. In other words, it is sufficient to impose convexity constraints on *consecutive* triangles of K_α in order to obtain convexity of K_α.

Lemma 3. *If, for all j such that $r^j \in \text{cone}(r^{j-1}, r^{j+1})$,*

$$\alpha_j \le \lambda^j_{j-1,j+1}\alpha_{j-1} + \lambda^j_{j+1,j-1}\alpha_{j+1},$$

then for all i, j, k such that $r^j \in \text{cone}(r^i, r^k)$,

$$\alpha_j \le \lambda^j_{i,k}\alpha_i + \lambda^j_{k,i}\alpha_k.$$

Proof. We prove it by induction on $p := k - i \pmod n$. If $p = 0$, $p = 1$ or $p = 2$, the result is true by hypothesis. We now prove that

$$\begin{array}{lllll} \text{if} & \forall i,j,l : & 2 \le l - i < p \pmod n & \text{and } r^j \in \text{cone}(r^i, r^l), & \alpha_j \le \lambda^j_{i,l}\alpha_i + \lambda^j_{l,i}\alpha_l \\ \text{then} & \forall i,j,l : & l - i = p \pmod n & \text{and } r^j \in \text{cone}(r^i, r^l), & \alpha_j \le \lambda^j_{i,l}\alpha_i + \lambda^j_{l,i}\alpha_l. \end{array}$$

Let j, k be such that $r^j, r^k \notin \{r^i, r^l\}$, $r^j \ne r^k$ and $r^j, r^k \in \text{cone}(r^i, r^l)$. Without loss of generality, we can assume that $r^j \in \text{cone}(r^i, r^k)$ and $r^k \in \text{cone}(r^j, r^l)$, i.e.

$$r^j = \lambda^j_{i,k} r^i + \lambda^j_{k,i} r^k \tag{3.3}$$

$$r^k = \lambda^k_{j,l} r^j + \lambda^k_{l,j} r^l \tag{3.4}$$

$$\lambda^j_{i,k}, \lambda^j_{k,i}, \lambda^k_{j,l}, \lambda^k_{l,j} \ge 0 \tag{3.5}$$

hence, using (3.4) in (3.3),

$$r^j = \lambda^j_{i,k} r^i + \lambda^j_{k,i}(\lambda^k_{j,l} r^j + \lambda^k_{l,j} r^l)$$
$$(1 - \lambda^j_{k,i}\lambda^k_{j,l}) r^j = \lambda^j_{i,k} r^i + \lambda^j_{k,i}\lambda^k_{l,j} r^l.$$

This describes r^j in terms of r^i and r^l, giving, by definition of the λ symbols (Definition 25),

$$\lambda^j_{i,l} = \frac{\lambda^j_{i,k}}{1 - \lambda^j_{k,i}\lambda^k_{j,l}}, \quad \lambda^j_{l,i} = \frac{\lambda^j_{k,i}\lambda^k_{l,j}}{1 - \lambda^j_{k,i}\lambda^k_{j,l}} \tag{3.6}$$

and this is well defined because $r^j \in \text{cone}(r^i, r^l)$.

Since the rays are ordered, $r^k \ne r^l$ and $r^k \in \text{cone}(r^i, r^l)$, we know that $k - i < p \pmod n$. Similarly, $l - j < p \pmod n$. Therefore, we can write, using the induction hypothesis,

$$\alpha_j \le \lambda^j_{i,k}\alpha_i + \lambda^j_{k,i}\alpha_k \tag{3.7}$$

$$\alpha_k \leq \lambda^k_{j,l}\alpha_j + \lambda^k_{l,j}\alpha_l \tag{3.8}$$

hence, replacing α_k in (3.7) by the right-hand side of (3.8) and given that $\lambda^j_{k,i} \geq 0$ in (3.5), we obtain the new inequality

$$\alpha_j \leq \lambda^j_{i,k}\alpha_i + \lambda^j_{k,i}(\lambda^k_{j,l}\alpha_j + \lambda^k_{l,j}\alpha_l)$$

which can be rewritten as

$$(1 - \lambda^j_{k,i}\lambda^k_{j,l})\alpha_j \leq \lambda^j_{i,k}\alpha_i + \lambda^j_{k,i}\lambda^k_{l,j}\alpha_l.$$

Since $\lambda^j_{i,l}, \lambda^j_{l,i} \geq 0$, given their expression in (3.6), we know that $(1 - \lambda^j_{k,i}\lambda^k_{j,l}) \geq 0$ and

$$\alpha_j \leq \frac{\lambda^j_{i,k}}{1 - \lambda^j_{k,i}\lambda^k_{j,l}}\alpha_i + \frac{\lambda^j_{k,i}\lambda^k_{l,j}}{1 - \lambda^j_{k,i}\lambda^k_{j,l}}\alpha_l$$

or equivalently, using again the expressions in (3.6),

$$\alpha_j \leq \lambda^j_{i,l}\alpha_i + \lambda^j_{l,i}\alpha_l.$$

We can prove similarly that $\alpha_k \leq \lambda^k_{i,l}\alpha_i + \lambda^k_{l,i}\alpha_l$ which concludes the induction for $l - i = p$. □

Lemma 4. *$\overline{Q}$ is a subset of Q.*

Proof. Consider $\alpha \in \mathbb{R}^n_+$ such that $\alpha \in \overline{Q}$. Some constraints of Q do not belong to the description of $\overline{Q}$. We must prove that they are satisfied. Let $x \in C_{ij}$ with $j = i + 1 \pmod n$. We consider all h, k such that $x \in C_{hk}$. Obviously, $C_{ij} \subseteq C_{hk}$, thus $r^i, r^j \in \text{cone}(r^h, r^k)$. In particular, using Lemma 3, we have

$$\alpha_i \leq \lambda^i_{h,k}\alpha_h + \lambda^i_{k,h}\alpha_k \tag{3.9}$$

$$\alpha_j \leq \lambda^j_{h,k}\alpha_h + \lambda^j_{k,h}\alpha_k. \tag{3.10}$$

Using the description of $\overline{Q}$, we also have

$$s^x_{i,j}\alpha_i + s^x_{j,i}\alpha_j \geq 1. \tag{3.11}$$

We now need to prove that $s^x_{h,k}\alpha_h + s^x_{k,h}\alpha_k \geq 1$.

Using Definition 24, we can express x in terms of f, r^i, r^j. And since $r^i, r^j \in \text{cone}(r^h, r^k)$, we can use Definition 25 to express them in terms of r^h, r^k:

$$x = f + s^x_{i,j}r^i + s^x_{j,i}r^j \tag{3.12}$$

$$r^i = \lambda^i_{h,k}r^h + \lambda^i_{k,h}r^k \tag{3.13}$$

$$r^j = \lambda^j_{h,k}r^h + \lambda^j_{k,h}r^k \tag{3.14}$$

hence, using (3.13)-(3.14) in (3.12),

$$x = f + (s^x_{i,j}\lambda^i_{h,k} + s^x_{j,i}\lambda^j_{h,k})\, r^h + (s^x_{i,j}\lambda^i_{k,h} + s^x_{j,i}\lambda^j_{k,h})\, r^k$$

which gives an expression of x in terms of r^h and r^k. Therefore, by Definition 24,

$$\left\{\begin{array}{l} s^x_{h,k} = s^x_{i,j}\lambda^i_{h,k} + s^x_{j,i}\lambda^j_{h,k} \\ \\ s^x_{k,h} = s^x_{i,j}\lambda^i_{k,h} + s^x_{j,i}\lambda^j_{k,h} \end{array}\right. . \tag{3.15}$$

Using (3.9)-(3.10) in (3.11), since $s^x_{i,j}, s^x_{j,i} \geq 0$, we obtain

$$s^x_{i,j}(\lambda^i_{h,k}\alpha_h + \lambda^i_{k,h}\alpha_k) + s^x_{j,i}(\lambda^j_{h,k}\alpha_h + \lambda^j_{k,h}\alpha_k) \geq 1$$

$$(s^x_{i,j}\lambda^i_{h,k} + s^x_{j,i}\lambda^j_{h,k})\,\alpha_h + (s^x_{i,j}\lambda^i_{k,h} + s^x_{j,i}\lambda^j_{k,h})\,\alpha_k \geq 1$$

which, given (3.15), is equivalent to $s^x_{h,k}\alpha_h + s^x_{k,h}\alpha_k \geq 1$ □

Lemma 5. *If $c > 0$, all optimal solutions to* $\min\{c^T\alpha : \alpha \in Q\}$ *are feasible for $\overline{Q}$.*

Proof. Let $\alpha^* \in Q \setminus \overline{Q}$. We want to prove that α^* is not an optimal solution to $\min\{c^T\alpha : \alpha \in Q\}$. Since $\alpha^* \notin \overline{Q}$, at least one constraint of $\overline{Q}$ that is not in Q must be violated by α^*, i.e. there exists i such that

$$r^i \in \text{cone}(r^{i-1}, r^{i+1}) \text{ and } \alpha^*_i > \lambda^i_{i-1,i+1}\alpha^*_{i-1} + \lambda^i_{i+1,i-1}\alpha^*_{i+1}$$

Consider α' such that

$$\alpha'_j = \begin{cases} \alpha^*_j, & j \neq i \\ \lambda^i_{i-1,i+1}\alpha^*_{i-1} + \lambda^i_{i+1,i-1}\alpha^*_{i+1}, & j = i. \end{cases} \tag{3.16}$$

We claim that $\alpha' \in Q$. First, trivially, for all $j, k \neq i$ and $x \in \mathcal{X}_{jk}$,

$$s^x_{j,k}\alpha'_j + s^x_{k,j}\alpha'_k \geq 1. \tag{3.17}$$

Then, for all k and $x \in \mathcal{X}_{ik}$, from Definition 24 and Definition 25, we have

$$\begin{aligned} x &= f + s^x_{i,k}\ r^i + s^x_{k,i}\ r^k \\ r^i &= \lambda^i_{i-1,i+1}r^{i-1} + \lambda^i_{i+1,i-1}r^{i+1} \end{aligned}$$

with $s^x_{i,k}, s^x_{k,i}, \lambda^i_{i-1,i+1}, \lambda^i_{i+1,i-1} \geq 0$, hence

$$x = f + s^x_{i,k}\lambda^i_{i-1,i+1}r^{i-1} + s^x_{i,k}\lambda^i_{i+1,i-1}r^{i+1} + s^x_{k,i}r^k$$

is a valid representation of x. Thus $(x,\ s^x_{i,k}\lambda^i_{i-1,i+1}e_{i-1} + s^x_{i,k}\lambda^i_{i+1,i-1}e_{i+1} + s^x_{k,i}e_k) \in P_I$ and since $\alpha^* \in Q$, it must satisfy

$$\begin{aligned} s^x_{i,k}\lambda^i_{i-1,i+1}\alpha^*_{i-1} + s^x_{i,k}\lambda^i_{i+1,i-1}\alpha^*_{i+1} + s^x_{k,i}\alpha^*_k &\geq 1 \\ s^x_{i,k}(\lambda^i_{i-1,i+1}\alpha^*_{i-1} + \lambda^i_{i+1,i-1}\alpha^*_{i+1}) + s^x_{k,i}\alpha^*_k &\geq 1 \\ s^x_{i,k}\alpha'_i + s^x_{k,i}\alpha'_k &\geq 1, \end{aligned} \tag{3.18}$$

the third inequality being obtained because of the construction of α' in (3.16). Together, (3.17) and (3.18) prove that $\alpha' \in Q$. By construction, $c^T\alpha' < c^T\alpha^*$, if $c > 0$. Therefore α^* is not optimal. □

As a byproduct, the proof of Lemma 5 shows that independently of the objective function c, given $\alpha^* \in Q \setminus \overline{Q}$, there exists $\alpha' \in \overline{Q}$ which provides coefficients for a cut that strictly dominates the one based on α^*. This is the reason for requiring $c > 0$.

Proof of Theorem 3. Lemma 5 shows that all optimal solutions to $\min\{c^T\alpha : \alpha \in Q\}$ are feasible for $\overline{Q}$. Since $\overline{Q} \subseteq Q$ (Lemma 4), they correspond to the set of optimal solutions to $\min\{c^T\alpha : \alpha \in \overline{Q}\}$. □

Corollary 2. *All vertices of Q are vertices of $\overline{Q}$.*

Proof. For any vertex α^* of Q, there must exist an objective function $\bar{c}$ such that α^* is the unique optimal solution to $\min\{\bar{c}^T\alpha \,:\, \alpha \in Q\}$. For any c such that $c_i < 0$, the problem is unbounded. For any c such that $c_i = 0$, the optimal solution is not unique. Therefore, all vertices of Q can be obtained by optimizing over Q with a positive objective function, and Theorem 3 applies. □

Theorem 3 does not provide a way to tackle the case where there are zero coefficients in the objective function c, which may be important since we typically want to separate points that contain zero components. However, the following corollary holds for any $c \geq 0$.

Corollary 3. *Given $c \geq 0$, any valid inequality $\alpha^T x \geq 1$ for $\operatorname{conv}(P_I)$ with $\alpha \in Q\backslash\overline{Q}$ is strictly dominated by a valid inequality $\hat{\alpha}^T x \geq 1$ with $\hat{\alpha} \in \overline{Q}$.*

Proof. Observe that α can be expressed as $\alpha = \hat{\alpha} + \hat{\beta}$ where $\hat{\alpha}$ is in the convex hull of the vertices of Q (thus $\hat{\alpha} \in \overline{Q}$ by Corollary 2) and $\hat{\beta}$ is in the recession cone of Q. As the recession cone of Q is in $\mathbb{R}^n_+$, it follows that $\hat{\beta} \geq 0$, hence $c^T\hat{\alpha} \leq c^T\alpha$. Since $\alpha \notin \overline{Q}$, $\hat{\beta} \neq 0$. In other words, $\hat{\alpha} \leq \alpha$ and $\hat{\alpha}_j < \alpha_j$ for some j. □

3.2 Separation algorithm

Optimizing over the set $\overline{Q}$ developed above requires explicit knowledge of the sets $\mathcal{X}_{i,i+1}$. More precisely, we should compute, for every $\operatorname{cone}(r^i, r^{i+1})$, the vertices of the convex hull of $\mathbb{Z}^2 \cap (f + \operatorname{cone}(r^i, r^{i+1}))$. To each of them corresponds one linear constraint of $\overline{Q}$. The number of such vertices is polynomial in the encoding length of (r^i, r^{i+1}) [53], and a polynomial-time algorithm for computing them has been presented in [3]. Note however that an important hurdle in implementing that algorithm is that one of its step is the computation of a Hilbert basis of the cone C_{ij}. Such computation is polynomial only in the special case of a cone in $\mathbb{R}^2$, and to the best of our knowledge, the algorithms specific to that case have never been implemented.

We adopt a fundamentally different approach that lets us avoid fully computing $\mathcal{X}_{i,i+1}$ and considering one linear constraint per point in $\mathcal{X}_{i,i+1}$. In addition to being easier to implement, we will see in Section 3.3 that this approach lets us keep the linear program that we optimize over extremely small. In order to do that, we relax the expression of $\overline{Q}$, by considering constraints (3.1) only for x in small sets $S_{i,i+1} \subseteq C_{i,i+1} \cap \mathbb{Z}^2$ instead of in $\mathcal{X}_{i,i+1}$. Note that this is indeed a relaxation since the constraints that correspond to points in $(C_{i,i+1} \cap \mathbb{Z}^2) \setminus \mathcal{X}_{i,i+1}$ are redundant yet valid for $\overline{Q}$. We denote this relaxation by $\overline{Q}(S) \supseteq \overline{Q}$, where $S = \cup_i S_{i,i+1}$.

$$\begin{aligned}\overline{Q}(S) = \{\, \alpha \in \mathbb{R}^n_+ \,:\, & s^x_{i,i+1}\alpha_i + s^x_{i+1,i}\alpha_{i+1} \geq 1, && \forall i,\ \forall x \in S_{i,i+1} \\ & \alpha_i \leq \lambda^i_{i-1,i+1}\alpha_{i-1} + \lambda^i_{i+1,i-1}\alpha_{i+1}, && \forall i : r^i \in \operatorname{cone}(r^{i-1}, r^{i+1}) \,\}\end{aligned}$$

We then follow a classic row-generation approach summarized in Algorithm 1. First, we initialize S to a reasonable subset of $\cup_i \mathcal{X}_{i,i+1}$. We then optimize over $\overline{Q}(S)$ and find a solution α. As $\overline{Q}(S)$ is a relaxation

```
Initialization:  $S := S_0$
Step A:          $\bar{\alpha} := \text{argmin}\{c^T\alpha \,:\, \alpha \in \overline{Q}(S)\}$

Step B:          Look for $x \in \mathbb{Z}^2$ such that $\bar{\alpha} \notin \overline{Q}(S \cup \{x\})$
                 If no such $x$ exist
                     $\bar{\alpha}$ is optimal for $\min\{c^T\alpha \,:\, \alpha \in \overline{Q}\}$, terminate.
                 Otherwise
                     $S := S \cup \{x\}$, go back to Step A.
```

Algorithm 1: Using $\overline{Q}(S)$ to optimize over $\overline{Q}$

```
Input:   $S \subset \mathbb{Z}^2$, $\bar{\alpha} \in \overline{Q}(S)$
Step 1:  Let $T$ be the set of vertices of conv$(S \cap L_{\bar{\alpha}})$.
         Check whether conv$(T)$ is lattice-free.
Step 2:  Check whether there are integer points
         in the relative interior of the edges of conv$(T)$
         that are in the interior of $L_{\bar{\alpha}}$.
Step 3:  Assume $\bar{\alpha} > 0$.
         Use Theorem 4 to check whether $L_{\bar{\alpha}}$ is lattice-free.
```

Algorithm 2: Oracle for finding integer points in the interior of $L_{\bar{\alpha}}$

of $\overline{Q}$, α may violate some constraints (3.1). If we find such a constraint, we add the corresponding point x to S, and iterate. Otherwise, if no such constraint is violated, α is valid for $\overline{Q}$ and is thus the desired optimal solution.

This scheme mainly relies on the fact that we possess an oracle that is able to find violated constraints of $\overline{Q}$, or prove that no such constraints exist. We describe such an oracle in this section. Note that the complexity of the algorithm as it is stated here is undefined, as it depends on the output of the oracle. Adopting a geometric perspective, we can restate the task of the oracle as follows: Given $\overline{Q}(S)$, $\bar{\alpha} := \text{argmin}\{c^T\alpha \,:\, \alpha \in \overline{Q}(S)\}$ and the polyhedron $L_{\bar{\alpha}}$, find $x \in \mathbb{Z}^2 \cap \text{interior}(L_{\bar{\alpha}})$ or prove that no such points exist. Barvinok [16] presented a polynomial-time algorithm that can solve this problem in any fixed dimension d, with the vertices of $L_{\bar{\alpha}}$ as its only input. However, we proceed otherwise, taking advantage of our specific two-dimensional setup and our knowledge of the set S. Our proposed oracle can be summarized as described in Algorithm 2. We detail the procedure in the rest of this section.

Step 1. We define $T \subseteq S$ to be the set of vertices of conv$(S \cap L_{\bar{\alpha}})$. An example is shown on Figure 3.4. After Step A, $s^x_{i,j}\bar{\alpha}_i + s^x_{j,i}\bar{\alpha}_j \geq 1$ for all $x \in S$. Therefore, no such x lies in the interior of $L_{\bar{\alpha}}$, and points in T must be on the boundary of $L_{\bar{\alpha}}$. We want to check whether conv(T) is lattice-free. By triangularizing conv(T), the problem reduces to finding integer points on line segments and in the interior of triangles with integer vertices (see e.g. [63] for related methods). Both can be solved by elementary modulo calculus. In particular, the number of integer points in the interior of a triangle conv$(\{0, u, v\})$, with $u, v \in \mathbb{Z}^2$, follows directly from Pick's formula (see for example [17])

$$N_{\text{interior}} = 1 + \frac{\det([u|v]) - \gcd(u_1, u_2) - \gcd(v_1, v_2) - \gcd(v_1 - u_1, v_2 - u_2)}{2} \tag{3.19}$$

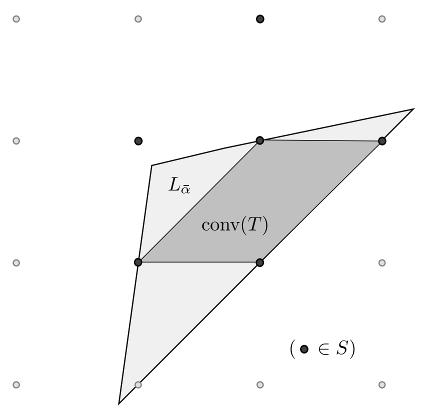

Figure 3.4: $T :=$ vertices of $\operatorname{conv}(S \cap L_{\bar{\alpha}})$

Then, knowing that the triangle contains integral points, we find them using the following procedure: If there are lattice points in the relative interior of two or three edges, we construct an integral point using integer combinations of these. This point is in the interior of conv(T) except when there is exactly one lattice point in the relative interior of each edge, in which case we divide conv(T) in the 4 sub-triangles they define, and proceed with one of these sub-triangles, as they all contain the same number of integer points in their interior. Otherwise, at least two edges contain no lattice points in their relative interior, and we make use of Lemma 6.

Lemma 6. *Let Δ be a triangle with integer vertices $\{0, u, v\}$ that has interior lattice points and such that* $\gcd(u_1, u_2) = \gcd(v_1, v_2) = 1$. *$\Delta$ has an interior lattice point w such that* $w = \frac{1}{\det([u|v])}u + \frac{k_v}{\det([u|v])}v$ *with* $k_v \in \mathbb{Z}_+$.

Proof. Any point w in $\mathbb{Z}^2$ can be expressed as $w = \beta_u u + \beta_v v$, with

$$\begin{pmatrix} \beta_u \\ \beta_v \end{pmatrix} = \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix}^{-1} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}.$$

By explicitly developing the matrix inverse, we get $\beta_u = \frac{k_u(w)}{\det([u|v])}$ and $\beta_v = \frac{k_v(w)}{\det([u|v])}$, with $k_u(w) := w_1 v_2 - w_2 v_1$ and $k_v(w) := u_1 w_2 - u_2 w_1$. Note that $k_u(w)$ and $k_v(w)$ are the components of w in the coordinate system defined by u and v, multiplied by $\det([u|v])$, and are thus integral as long as w is integer. We are looking for a point w in the interior of Δ, i.e. such that $k_u(w), k_v(w) \geq 1$ and $k_u(w)+k_v(w) \leq \det([u|v])-1$. The claim in this Lemma is that such a point exists even when $k_u(w)$ is fixed to 1. We now consider $k_u(w) = 1$ as a Diophantine equation with variables w_1, w_2, i.e. $w_1 v_2 - w_2 v_1 = 1$. Since $\gcd(v_2, -v_1) = 1$, there exist $\bar{w}_1, \bar{w}_2 \in \mathbb{Z}$ such that $k_u(\bar{w}) = 1$, and we could find the value of $\bar{w}$ using the Euclidian algorithm. If $1 \leq k_v(\bar{w}) \leq \det([u|v]) - 2$, then $\bar{w}$ is an interior lattice point. Otherwise, we build the integer point $w' = \bar{w} + \lambda v$, with $\lambda = -\lfloor k_v(\bar{w})/\det([u|v])\rfloor$. Observe that $k_u(w') = 1$ and $k_v(w') \in \{0, 1, \ldots, \det([u|v]) - 1\}$. The point w' has the desired form and is in the interior of the triangle Δ unless $k_v(w') = 0$ or $k_v(w') = \det([u|v]) - 1$. The first case, $k_v(w') = 0$, is impossible since the segment $(0, u)$ does not have lattice points in its relative interior. In the second case, we note that

$w' = \frac{1}{\det([u|v])}u + \frac{\det([u|v])-1}{\det([u|v])}v \in \mathbb{Z}^2$. By hypothesis, there exists an integer point in the interior of Δ. Let $\widehat{w} = \frac{k_u(\widehat{w})}{\det([u|v])}u + \frac{k_v(\widehat{w})}{\det([u|v])}v$ be such a point. We know that $k_u(\widehat{w}), k_v(\widehat{w}) \geq 1$ and $k_u(\widehat{w}) + k_v(\widehat{w}) < \det([u|v]))$. Finally, we build a second point $w'' = \widehat{w} + (k_u(\widehat{w}) - 1)(v - w') = \frac{1}{\det([u|v])}u + \frac{k_u(\widehat{w})+k_v(\widehat{w})-1}{\det([u|v])}v$, which proves the claim. □

Lemma 6 allows us to find an integer point $w = \frac{1}{\det([u|v])}u + \frac{k_v}{\det([u|v])}v$ in the interior of Δ by solving the Diophantine system

$$\begin{cases} u_1 + k_v v_1 = k_1 \det([u|v]) \\ u_2 + k_v v_2 = k_2 \det([u|v]) \end{cases}, \; k_v, k_1, k_2 \in \mathbb{Z}$$

for k_v, choosing the smallest positive solution. This can be done either by three applications of the Euclidean algorithm or by using the Hermite normal form of the system. In both cases, finding the smallest positive solution is easy as the set of solutions is a one-dimensional translated lattice.

Step 2. We now assume that $\operatorname{conv}(T)$ is lattice-free, and we check that the relative interior of its edges does not contain integer points that are in the interior of $L_{\bar{\alpha}}$. Note that since $L_{\bar{\alpha}}$ is convex, it is enough to check one integer point in the relative interior of each edge of $\operatorname{conv}(T)$.

Step 3. We now assume that no integer point was found in the interior of $L_{\bar{\alpha}}$ through Steps 1 and 2. Moreover, we assume that $\bar{\alpha}_i > 0$ for all i, hence $L_{\bar{\alpha}}$ is a polytope; we show in Section 3.3 how we proceed if this assumption is not true. We start by showing that $L_{\bar{\alpha}}$ is tight at three affinely independent points in S, or in other words that $\operatorname{conv}(T) = \operatorname{conv}(S \cap L_{\bar{\alpha}})$ is full-dimensional.

Lemma 7. *Let $\bar{\alpha}$ be a vertex of $\overline{Q}(S)$. If $\operatorname{cone}(r^1, \ldots, r^n) = \mathbb{R}^2$ and $\bar{\alpha} > 0$, then $S \cap L_{\bar{\alpha}}$ contains three affinely independent points.*

Proof. Note that we can assume wlog that for every $x \in S$ there exists a unique i such that $x \in S_{i,i+1}$. Consider now the constraints defining the set $\overline{Q}$(S).

$$s^x_{i,i+1}\alpha_i + s^x_{i+1,i}\alpha_{i+1} \geq 1, \qquad \forall i,\ \forall x \in S_{i,i+1} \tag{3.20}$$

$$\alpha_i \leq \lambda^i_{i-1,i+1}\alpha_{i-1} + \lambda^i_{i+1,i-1}\alpha_{i+1}, \quad \forall i :\ r^i \in \operatorname{cone}(r^{i-1}, r^{i+1}) \tag{3.21}$$

$$\alpha_i \geq 0 \qquad \forall i \tag{3.22}$$

There are $|S|$ constraints of type (3.20), n of type (3.21), and n of type (3.22), of which a subset of n linearly independent overall must be tight at $\bar{\alpha}$. Because $\bar{\alpha} > 0$, none of the nonnegativity constraints are tight for $\bar{\alpha}$. Now observe that v^i is a vertex of $L_{\bar{\alpha}}$ only if it is not on $\operatorname{aff}(v^{i-1}, v^{i+1})$. In other words, only if $\bar{\alpha}_i \neq \lambda^i_{i-1,i+1}\bar{\alpha}_{i-1} + \lambda^i_{i+1,i-1}\bar{\alpha}_{i+1}$, i.e. the associated constraint (3.21) is not tight.

Moreover, if $\operatorname{cone}(r^1, \ldots, r^n) = \mathbb{R}^2$, then $f \in \operatorname{interior}(L_{\bar{\alpha}})$. Therefore, $L_{\bar{\alpha}}$ is full-dimensional, and has at least three vertices. This implies that at most $n-3$ of the constraints (3.21) are tight for $\bar{\alpha}$. Equivalently, we have at least three tight constraints of type (3.20) for $\bar{\alpha}$. If the corresponding three integer points are affinely independent, the result follows.

Suppose now that they are on a line. More specifically let W be a collinear set of such tight points, i.e. $W \subseteq S \cap \operatorname{boundary}(L_{\bar{\alpha}})$ such that $|W| \geq 3$ and $\dim(\operatorname{aff}(W)) = 1$. Since $L_{\bar{\alpha}}$ is convex, all points in W

must belong to a single facet $\mathrm{conv}(v^i, v^j)$ of $L_{\bar\alpha}$. Now let us define K as the index set of the rays inside the corresponding cone, i.e. $K := \{k : r^k \in \mathrm{cone}(r^i, r^j)\}$. Observe that there are at most $|K|$ linearly independent tight constraints including the variables with indices in K only. Thus, there must be at least $n - |K|$ linearly independent tight constraints including at least one of the $n - |K|$ remaining variables. For at least one of these remaining variables, the associated ray supports a vertex of $L_{\bar\alpha}$, i.e. there exists $h \notin K$ such that v^h is a vertex of $L_{\bar\alpha}$. Therefore, at least one of the $n - |K|$ remaining constraints (3.21) is not tight for $\bar\alpha$. It follows that there is at least one additional tight constraint (3.20), which does not correspond to a point in W. □

As suggested by Lemma 7, we ensure that we have three affinely independent points on the boundary of $L_{\bar\alpha}$ by adding artificial rays to P_I, as needed, in order to have $\mathbb{R}^2$-spanning rays. By using a zero objective function cost for variables associated to artificial rays, we do not modify the separation problem. Observe that since $\mathrm{conv}(T)$ is a lattice-free polyhedron in $\mathbb{R}^2$ with integer vertices, we know that it has at most four vertices [60], thus $|T|$ may only be three or four, i.e. $\mathrm{conv}(T)$ is a triangle or a qualidrateral.

Definition 28 summarizes the assumptions we can make in Step 3: we call $(T, L_{\bar\alpha})$ *checkable* if we found integer points in the interior of $L_{\bar\alpha}$ neither in Step 1 nor in Step 2.

Definition 28. *We call a couple* $(T, L_{\bar\alpha})$ checkable *if*

(a) $\mathrm{conv}(T)$ *and* $L_{\bar\alpha}$ *are full-dimensional convex polytopes in* $\mathbb{R}^2$*,*

(b) the vertices of $\mathrm{conv}(T)$ *are integral and belong to the boundary of* $L_{\bar\alpha}$*,*

(c) $\mathrm{conv}(T)$ *is lattice-free,*

(d) the integer points in the relative interior of the edges of $\mathrm{conv}(T)$ *do not belong to the interior of* $L_{\bar\alpha}$*.*

We showed previously that it is easy to verify whether $(T, L_{\bar\alpha})$ is checkable. In the remainder of this section, we show that it is computationally cheap to check whether $L_{\bar\alpha}$ is lattice-free when $(T, L_{\bar\alpha})$ is checkable. Lemma 8, 9, 10 and 11 cover the four possible cases.

Lemma 8. *Let* $(T, L_{\bar\alpha})$ *be checkable and* $\mathrm{conv}(T)$ *be a lattice-free triangle with exactly one integer point in the relative interior of each edge. Then* $L_{\bar\alpha}$ *is lattice-free.*

Proof. $\mathrm{conv}(T)$ is a maximal lattice-free body. Therefore, $L_{\bar\alpha} \supseteq \mathrm{conv}(T)$ is lattice-free if and only if $L_{\bar\alpha} = \mathrm{conv}(T)$. Since the integer points on the edges of $\mathrm{conv}(T)$ are not in the interior of $L_{\bar\alpha}$, they are on its boundary, so $L_{\bar\alpha} = \mathrm{conv}(T)$. □

Lemma 9. *Let* $(T, L_{\bar\alpha})$ *be checkable and* $\mathrm{conv}(T)$ *be a triangle with vertices* D*,* $D + u$*,* $D + v$ *such that* $\det([u|v]) = 1$*. Then* $L_{\bar\alpha}$ *is lattice-free if and only if* $L_{\bar\alpha}$ *contains neither* $D + u + v$ *nor* $D + u - v$ *nor* $D + v - u$ *in its interior.*

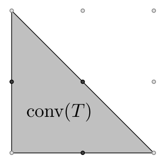

Figure 3.5: One lattice point in the relative interior of each edge of conv(T) (Lemma 8)

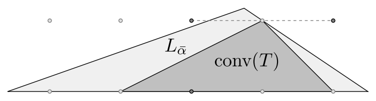

Figure 3.7: One or more lattice points in the relative interior of one edge of conv(T) (Lemma 10)

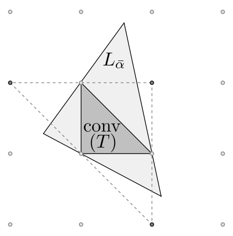

Figure 3.6: conv(T) is a unimodular triangle (Lemma 9)

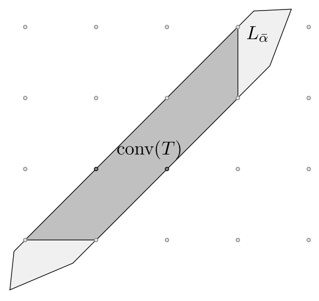

Figure 3.8: conv(T) is a quadrilateral (Lemma 11)

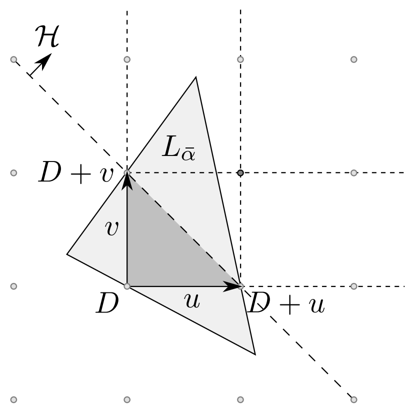

Figure 3.9: The half-plane $\mathcal{H}$ in the proof of Lemma 9

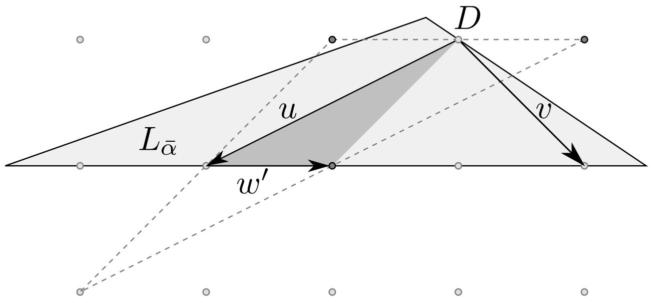

Figure 3.10: w' in Lemma 10

Proof. Let $\mathcal{H}$ be the half-plane delimited by the line $(D+u, D+v)$ not containing D (see Figure 3.9). We first consider the vertices of $L_{\bar\alpha}$ that lie in $\mathcal{H}$. Observe that by convexity, they must belong to $D+\text{cone}(u,v)$, otherwise $D+u$ or $D+v$ would belong to the interior of $L_{\bar\alpha}$. Since (u,v) is an integral basis of $\mathbb{Z}^2$, there is no integer point in the interior of $\text{conv}(\{D, D+v\})+\text{cone}(\{u\})$ or $\text{conv}(\{D, D+u\})+\text{cone}(\{v\})$. Hence, if the vertices of $L_{\bar\alpha}$ all lie in $\text{conv}(\{D, D+v\})+\text{cone}(\{u\})$ or all lie in $\text{conv}(\{D, D+u\})+\text{cone}(\{v\})$, then $L_{\bar\alpha}\cap\mathcal{H}$ is lattice-free. Otherwise, $D+u+v$ is in the interior of $L_{\bar\alpha}$. By symmetry for the two other half-planes, the result follows. □

Lemma 10. *Let* $(T, L_{\bar\alpha})$ *be checkable and* $\text{conv}(T)$ *be a lattice-free triangle with vertices* D, $D+u$, $D+v$ *such that* $\gcd(u_1,u_2)=\gcd(v_1,v_2)=1$ *and* $\gcd(w_1,w_2)\neq 1$, *with* $w=v-u$. *Then* $L_{\bar\alpha}$ *is lattice-free if and only if* $L_{\bar\alpha}$ *contains neither* $D+w'$ *nor* $D-w'$ *in its interior, with* $w'=\frac{w}{\gcd(w_1,w_2)}$.

Proof. By convexity, the point $D+u+w'$ belongs to $L_{\bar\alpha}$, but since it is in the relative interior of an edge of $\text{conv}(T)$, it is not in the interior of $L_{\bar\alpha}$. Thus, $D+u+w'$ is on the boundary of $L_{\bar\alpha}$ and we consider the triangle $\text{conv}(\{D, D+u, D+u+w'\})$. By (3.19), $\det([-u, w'])=1$ and Lemma 9 applies. Therefore, in that case, $L_{\bar\alpha}$ is lattice-free if and only if $D+w'$, $D-w'$ and $D+2u+w'$ do not lie in its interior. Note that since the line $(D+u, D+v)$ is a facet of $L_{\bar\alpha}$, $D+2u+w'$ can not be in $L_{\bar\alpha}$. □

Lemma 11. *Let* $(T, L_{\bar\alpha})$ *be checkable and* $\text{conv}(T)$ *be a lattice-free quadrilateral. Then* $L_{\bar\alpha}$ *is lattice-free.*

Proof. As for Lemma 10, we can decompose $\text{conv}(T)$ in unimodular triangles (Figure 3.11) and apply Lemma 9 on one of them. The vertices of this unimodular triangle can be vertices of $\text{conv}(T)$ or integer points in the relative interior of the edges of $\text{conv}(T)$, belonging in both cases to the boundary of $L_{\bar\alpha}$. One can easily see that the integer points to be verified in Lemma 9 are either on the edges of $\text{conv}(T)$ or trivially not in the interior of $L_{\bar\alpha}$, by convexity. □

Theorem 4. *Let* $(T, L_{\bar\alpha})$ *be checkable and* $\text{conv}(T)$ *be a 2-dimensional convex polytope of which* D, $D+u$ *and* $D+v$ *are three vertices. Then* $L_{\bar\alpha}$ *is lattice-free if and only if it contains neither* $D+u'+v'$ *nor* $D+u'-v'$ *nor* $D+v'-u'$ *in its interior, with* $u'=\frac{u}{\gcd(u_1,u_2)}$ *and* $v'=\frac{v}{\gcd(v_1,v_2)}$.

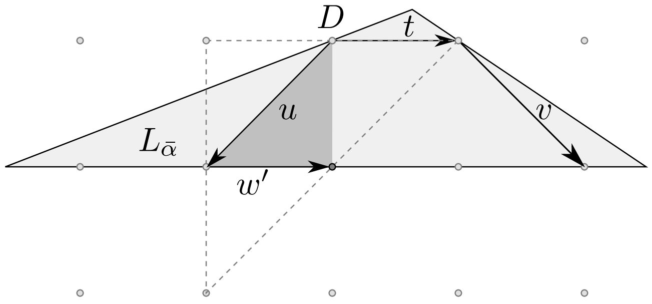

Figure 3.11: Unimodular triangle decomposition in Lemma 11

Proof. This follows from Lemma 8, 9, 10 and 11. □

The results in this section show that, in Step 3, it is enough to check at most three integer points against the interior of $L_{\bar{\alpha}}$ to verify that $L_{\bar{\alpha}}$ is lattice-free.

3.3 Computations

In this section, we present an implementation of the two-row cut separator described previously and computational results obtained with it. First, we need to address an issue that was not covered before.

Handling of the case $\alpha_i = 0$. So far, we have only considered the case where $L_{\bar{\alpha}}$ is a polytope, i.e. $\bar{\alpha}_i > 0$ for all i, while the constraints of $\overline{Q}(S)$ only ensure $\bar{\alpha}_i \geq 0$. Note that if $L_{\bar{\alpha}}$ is lattice-free and unbounded, then it is necessarily a split set. If only one coefficient $\bar{\alpha}_h$ is zero, then we can check that $L_{\bar{\alpha}}$ is inscribed in a split set parallel to r^h. If two such coefficients $\bar{\alpha}_j$ and $\bar{\alpha}_k$ are zero, then $L_{\bar{\alpha}}$ is not lattice-free and we can easily construct an integer point $f + \mu_j r^j + \mu_k r^k$ in its interior. However, both these operations involve computing a rational representation of the rays r^i and the point f, which are usually only available in floating-point form. Besides numerical difficulties, this could yield points with large coefficients, and that are very "far" from f. Recall that when we find points in $\mathbb{Z}^2 \cap \text{interior}(L_{\bar{\alpha}})$, we add them to the set S, whose elements correspond to the constraints of $\overline{Q}$ that are considered in its relaxation $\overline{Q}(S)$. In that context, large coefficients cause numerical instability, and points that are "far" from f typically correspond to weak (i.e. dominated) constraints of $\overline{Q}$.

We tackle this issue in a different way, by imposing positive lower bounds on α, thus ensuring that we obtain a bounded polyhedron $L_{\bar{\alpha}}$. This can yield problems in case some of these bounds end up tight in $\bar{\alpha}$, as then the result of Lemma 7 is not guaranteed to hold, i.e. $\text{conv}(T)$ may not be full-dimensional. If it is full-dimensional nevertheless, then Theorem 4 holds and we can still use the oracle described in Algorithm 2. Otherwise, as mentioned earlier, we could fall back on verifying that $L_{\bar{\alpha}}$ is lattice-free with the polynomial-time algorithm of Barvinok [16]. But for that case, we instead implement the naive enumeration that runs in $w(L_{\bar{\alpha}})$ iterations, where

$$w(L_{\bar{\alpha}}) = \max\{x_1 \ : \ x \in L_{\bar{\alpha}}\} - \min\{x_1 \ : \ x \in L_{\bar{\alpha}}\}.$$

This is not polynomial in the encoding length of $L_{\bar{\alpha}}$, and potentially much more costly than Algorithm 2, which is basically $O(1)$ for fixed values of $|S|$ and n. In practice however, it is possible to bound the number of iterations by suitably choosing the value of the lower bounds on α, for instance $\alpha_i \geq \frac{|r^i|}{K}$ for some $K > 0$. Since the vertices of $L_{\bar{\alpha}}$ are of the form $v^i = f + \frac{1}{\bar{\alpha}_i} r^i$, this ensures that $L_{\bar{\alpha}}$ is contained in a disc of radius K centered at f, implying $w(L_{\bar{\alpha}}) \leq 2K$. In our implementation, $K = 500$. Note also that if we fix the value of K, the complexity of Algorithm 1 becomes polynomial, as it performs at most $O(K^2)$ iterations.

Remark that $L_{\bar{\alpha}}$ being lattice-free is enough for $\bar{\alpha}^T s \geq 1$ to be a valid inequality for P_I, but if some components of $\bar{\alpha}$ are at their positive lower bound, then $\bar{\alpha}^T s \geq 1$ is not guaranteed to define a *facet* of $\operatorname{conv}(P_I)$, as the lower bounds on α are not actual constraints of the polar of $\operatorname{conv}(P_I)$. For this reason, whenever we generate a valid inequality having a coefficient α_j at lower bound, we consider instead the intersection cut corresponding to a split set of direction r^j, i.e. the lattice-free body $L_\alpha = \{x \in \mathbb{R}^2 : \lfloor p^T f \rfloor \leq p^T x \leq \lceil p^T f \rceil\}$ where $p \in \mathbb{Z}^2$ is an integral vector orthogonal to r^j. This is the only facet-defining inequality for P_I having $\alpha_j = 0$, and is hence a solution to $\min\{{s^*}^T \alpha \ : \ \alpha \in Q\}$ in this case, if K is sufficiently large. The computation can fail since $p^T f$ may be integral, which is made more likely by the fact that we must convert (approximately) a floating-point representation of r^j into a rational in order to compute p. If it fails for every such α_j at lower bound, then we discard the current cut, so that we return only facet-defining inequalities.

Computational experiment. Algorithm 3 summarizes the computational experiment performed in order to measure the practical speed of our method. Given a mixed-integer problem, the algorithm starts by optimizing over its linear relaxation. In the outer loop, we extract, from the simplex tableau associated to the current optimal solution x^*, the two-row models to be used for cut generation. In the inner loop, we separate one inequality with each model and add the cuts that separate x^* to the linear relaxation, over which we then reoptimize, yielding a new solution x^*. The inner loop terminates when no more separating inequality is found. At that point, the next iteration of the outer loop will build different models based on a new simplex tableau. Observe that at a given iteration of the outer loop, all the generated inequalities are at most of rank r.

In order to compare two-row inequalities with their single-row counterpart, we compute the one-row intersection cut associated to each row of the simplex tableau, at every outer loop iteration, before we start separating two-row inequalities. These cuts are intersection cuts on a one-row relaxation of the original problem, keeping the integrality constraint only for the corresponding basic variable (i.e. *non-lifted* intersection cuts). Note that while much easier to compute, they are a subset of those we can obtain with our two-row separator.

Although more sophisticated options exist (see e.g. [39, 34, 19, 18]), our method for building the two-row models is essentially heuristic. We arbitrarily restrict ourselves to reading rows from optimal simplex tableaux. Our intent is to build models whose constraints have similar supports, while covering all

```
input: a mixed-integer problem $P$, its linear relaxation $P_{LP}$

for $r = 1$ to RANK_MAX                                    (outer loop)
        Optimize over $P_{LP}$. Let $x^*$ be the optimal solution.
        Compute the optimal simplex tableau.
        Build up to MODELS_MAX two-row models.

        for each row of the simplex tableau,
                generate the corresponding one-row intersection cut.
        end for
        Add the one-row cuts separating $x^*$ to $P_{LP}$.

        do                                                 (inner loop)
                Optimize over $P_{LP}$. Let $x^*$ be the optimal solution.

                for each two-row model,
                        generate a cut, trying to separate $x^*$.
                end for
                Add the cuts separating $x^*$ to $P_{LP}$.

        while at least one cut was added.
end for
```

Algorithm 3: Computational experiment

relevant rows. Intuitively, this can be motivated by observing that any intersection cut generated from a model whose two rows have disjoint support is equivalent to a linear combination of two intersection cuts from the corresponding one-row models. In practice, we select up to `MODELS_MAX` models meeting the following requirements:

1. Each of the two rows is a *suitable* simplex tableau row, i.e.
   a. its basic variable is integer-constrained,
   b. its density, i.e. the ratio of the number of nonzero coefficients in the row over the number of columns, does not exceed `ROW_DENSITY_MAX`.
2. Each of the two rows is used in at most (`ROW_USE_MAX` $- 1$) other selected models.
3. At least one of the two rows has a fractional right-hand side.
4. Among the models that are not selected, none has a higher score. The score of a two-row model is computed as $(c - d)$ where c is the number of colums having nonzero coefficients in both rows, and d is the number of columns a having nonzero coefficient in exactly one row.

Results analysis. Tables 3.4 and 3.5 present the results of our experiment. The testbed is composed of problems from the MIPLIB 3 [23] and MIPLIB 2003 [1] libraries. We report results on all the instances except for three having no integrality gap (`dsbmip`, `enigma`, `disctom`), four whose optimal solution is unknown (`dano3mip`, `liu`, `momentum3`, `t1717`), and five for which the experiment runs out of memory (`ds`, `momentum2`, `stp3d`, `mzzv42z`, `rd-rplusc-21`). Tables 3.1 and 3.2 gather the characteristics of the

	rows	columns	integer	binary	continuous
10teams	230	2025	1800	all	225
air03	124	10757	10757	all	0
air04	823	8904	8904	all	0
air05	426	7195	7195	all	0
arki001	1048	1388	538	415	850
bell3a	123	133	71	39	62
bell5	91	104	58	30	46
blend2	274	353	264	231	89
cap6000	2176	6000	6000	all	0
danoint	664	521	56	all	465
dcmulti	290	548	75	all	473
egout	98	141	55	all	86
fast0507	507	63009	63009	all	0
fiber	363	1298	1254	all	44
fixnet6	478	878	378	all	500
flugpl	18	18	11	0	7
gen	780	870	150	144	726
gesa2	1392	1224	408	240	672
gesa2_o	1248	1224	720	336	504
gesa3	1368	1152	384	216	768
gesa3_o	1224	1152	672	336	480
gt2	29	188	188	24	0
harp2	112	2993	2993	all	0
khb05250	101	1350	24	all	1326
l152lav	97	1989	1989	all	0
lseu	28	89	89	all	0
misc03	96	160	159	all	1
misc06	820	1808	112	all	1696
misc07	212	260	259	all	1
mitre	2054	10724	10724	all	0
mod008	6	319	319	all	0
mod010	146	2655	2655	all	0
mod011	4480	10958	96	all	0
modglob	291	422	98	all	0
noswot	182	128	100	75	25
nw04	36	87482	87482	all	0
p0033	16	33	33	all	0
p0201	133	201	201	all	0
p0282	241	282	282	all	0
p0548	176	548	548	all	0
p2756	755	2756	2756	all	0
pk1	45	86	55	all	31
pp08a	136	240	64	all	176
pp08aCUTS	246	240	64	all	176
qiu	1192	840	48	all	792
qnet1	503	1541	1417	1288	124
qnet1_o	456	1541	1417	1288	124
rentacar	6803	9557	55	all	9502
rgn	24	180	100	all	80
rout	291	556	315	300	241
set1ch	492	712	240	all	472
seymour	4944	1372	1372	all	0
stein27	118	27	27	all	0
stein45	331	45	45	all	0
vpm1	234	378	168	all	210
vpm2	234	378	168	all	210

Table 3.1: Instance statistics (MIPLIB 3)

relevant instances. The general conditions of our experiments are detailed in Table 3.3. The columns of Tables 3.4 and 3.5 are composed of two parts. The first one (*one-row only*) serves as a comparison point using only one-row intersection cuts, i.e. split cuts from a simple disjunction on a basic variable (`MODELS_MAX = 0`, `ROW_USE_MAX = 0`). The column *cuts* indicates the number of separating one-row intersection cuts and *%gc* is the percentage of integrality gap closed as a result. We compute gap closures as

$$\%gc = 100\,\frac{z_{\text{LP+cuts}} - z_{\text{LP}}}{z_{\text{MIP}} - z_{\text{LP}}}$$

where z_{MIP} is the optimal objective function value of the original problem, z_{LP} the one of its LP relaxation, and $z_{\text{LP+cuts}}$ the one of its LP relaxation with cuts added. The second part (*one-row + two-row*)

	rows	columns	integer	binary	continuous
a1c1s1	3312	3648	192	all	3456
aflow30a	479	842	421	all	421
aflow40b	1442	2728	1364	all	1364
atlanta-ip	21732	48738	46773	46667	1965
glass4	396	322	302	all	20
manna81	6480	3321	3321	18	0
momentum1	42680	5174	2349	all	2825
msc98-ip	15850	21143	20290	20237	853
mzzv11	9499	10240	10240	9989	0
net12	14021	14115	1603	all	12512
nsrand-ipx	735	6621	6620	all	1
opt1217	64	769	768	all	1
protfold	2112	1835	1835	all	0
roll3000	2295	1166	738	246	428
sp97ar	1761	14101	14101	all	0
timtab1	171	397	171	64	226
timtab2	294	675	294	113	381
tr12-30	750	1080	360	360	720

Table 3.2: Instance statistics (MIPLIB 2003, instances not included in MIPLIB 3)

CPU:	Intel Core i7-990X at 3.47GHz, 6 cores, 12 threads
RAM:	DDR3-1333 SDRAM (24Gb)
Compiler:	GCC 4.6.3 20120306 (Red Hat 4.6.3-2)
Environment:	GNU/Linux (Fedora 15), kernel 2.6.43.8-1.fc15.x86_64
Cut generation:	Implemented in C++, single threaded
LP solver:	IBM CPLEX 12.4 (C library API), 64 bits, single threaded

Table 3.3: Conditions of the experiments

corresponds to Algorithm 3 with `MODELS_MAX = 5000` and `ROW_USE_MAX = 4`, i.e. each row of the simplex tableau is used to build at most 4 different two-row models. Note that we do not consider rows with more than 40% nonzero components (`ROW_DENSITY_MAX = 0.4`). In both cases, we limit ourselves to rank-5 inequalities (`RANK_MAX = 5`), and we discard cuts whose dynamism (i.e. the quotient of the largest and the smallest nonzero coefficient, in absolute value) exceeds 10^6, as they are likely to cause numerical difficulties. Moreover, we consider that a cut $\alpha^T x \geq 1$ "separates" a point x^* only if its violation at x^* is at least 10^{-6} i.e. $1 - \alpha^T x^* \geq 10^{-6}$. The column *one-row cuts* indicates the number of separating one-row intersection cuts generated as part of Algorithm 3. In the subcategory *two-row*, *models* indicates the overall number of times the two-row separation procedure is called, *time* shows the total time spent within the algorithm, in seconds, and *cuts* indicates the number of two-row cuts that succeed at separating the corresponding x^*. The set S in Algorithm 1 is initialized with the four points $(\lfloor f_1 \rfloor, \lfloor f_2 \rfloor)$, $(\lfloor f_1 \rfloor, \lceil f_2 \rceil)$, $(\lceil f_1 \rceil, \lfloor f_2 \rfloor)$ and $(\lceil f_1 \rceil, \lceil f_2 \rceil)$, and *+total* denotes the total number times a point was added to a set S, across all separations. Finally, *%gc* shows the percentage of gap closed by adding all the separating cuts.

The primary objective of our experiment is to assess whether our separator is fast in practice. In particular, since Algorithm 1 does not have a proven complexity bound, we need to evaluate how many iterations it performs in a practical setting. Over the course of the experiment conducted to generate Table 3.4 and Table 3.5, only 10.4 points are added to S on average per call to the separator, in addition to the four initial points, i.e. Algorithm 1 performs 11.4 iterations on average. On average, the computation takes 20.5ms per call to the separator (1.8ms when considering MIPLIB 3 only). A cut is successfully computed in 99.87% of the cases, failure happening when no facet-defining inequality could be generated as explained above, and 1.39% of the generated cuts do separate the corresponding point x^*. Note that

instance	one-row only		one-row + two-row					
			one-row	two-row				
	cuts	%gc	cuts	models	+total	time	cuts	%gc
10teams	699	0.00	699	286	267	0.179	0	0.00
air03	36	100.00	36	232	111	0.316	0	100.00
air04	1299	9.49	1299	0	0	0.000	0	9.49
air05	1051	6.32	1051	0	0	0.000	0	6.32
arki001	163	27.28	161	14592	72606	32.034	238	32.23
bell3a	71	69.56	71	3008	16994	3.614	430	68.25
bell5	115	26.23	90	1656	8678	1.403	25	23.08
blend2	46	21.61	66	12453	5988	4.357	203	26.73
cap6000	67	54.19	67	0	0	0.000	0	54.19
danoint	100	0.43	95	4657	68866	55.138	121	0.61
dcmulti	278	58.32	206	7052	16350	6.574	370	65.64
egout	79	69.81	47	1867	3781	0.908	244	93.37
fast0507	1662	3.10	1662	453	220	0.759	0	3.10
fiber	253	17.04	195	30092	84777	37.087	401	18.81
fixnet6	93	18.66	78	27487	18291	14.743	535	53.54
flugpl	43	14.22	38	540	1428	0.353	133	20.37
gen	211	61.19	224	7957	36457	10.054	366	63.66
gesa2	290	47.25	270	22181	34055	11.352	687	70.54
gesa2_o	398	47.13	380	52715	71444	29.022	960	67.23
gesa3	324	49.72	208	12012	65236	18.045	668	74.46
gesa3_o	421	67.36	358	39177	80491	27.407	1049	74.62
gt2	79	97.54	78	817	2807	0.485	20	99.00
harp2	130	11.85	133	6612	26458	6.808	264	18.53
khb05250	53	95.57	38	588	258	0.192	43	90.67
l152lav	326	15.20	326	0	0	0.000	0	15.20
lseu	80	38.00	93	722	3144	0.540	96	36.89
markshare1	29	0.00	29	0	0	0.000	0	0.00
markshare2	34	0.00	34	0	0	0.000	0	0.00
mas74	74	4.38	74	0	0	0.000	0	4.38
mas76	77	3.06	77	0	0	0.000	0	3.06
misc03	275	4.56	293	3330	1792	1.129	100	17.36
misc06	37	63.18	52	5383	2796	1.949	140	86.35
misc07	392	0.72	352	6980	13931	5.004	24	0.72
mitre	5631	83.93	5496	125000	826666	210.567	3141	84.45
mkc	725	39.40	788	130000	36106	51.893	676	26.31
mod008	33	11.22	33	0	0	0.000	0	11.22
mod010	258	57.73	256	140	24	0.085	2	58.84
mod011	22	6.87	21	6178	5629	2.869	52	12.41
modglob	50	28.72	42	2392	13963	2.754	122	48.41
noswot	163	0.00	119	3536	5176	1.112	167	0.00
nw04	76	17.95	76	0	0	0.000	0	17.95
p0033	34	12.77	57	1046	10709	1.681	76	57.01
p0201	325	25.93	383	1502	4121	1.131	116	45.17
p0282	182	16.03	247	21296	135084	34.065	100	13.54
p0548	300	50.83	367	17816	57142	15.693	261	66.53
p2756	264	0.89	374	39743	48256	18.420	201	42.12
pk1	68	0.00	68	0	0	0.000	0	0.00
pp08a	204	77.53	158	4048	7886	3.168	217	90.16
pp08acuts	99	47.14	148	5336	23651	8.664	238	60.23
qiu	116	3.05	92	1804	38900	30.450	88	4.64
qnet1	298	22.99	288	13592	18518	8.649	45	28.01
qnet1_o	152	47.84	125	12859	24792	7.785	121	51.54
rentacar	9	0.00	9	368	1254	5.036	0	0.00
rgn	72	0.00	72	385	110	0.123	47	0.00
rout	195	7.81	194	456	414	0.205	7	9.30
set1ch	464	80.71	304	16179	20079	6.433	543	94.14
seymour	22038	14.75	22167	45000	338910	652.530	12	15.01
stein27	452	0.00	437	3248	8342	2.595	18	0.00
stein45	1069	0.00	1089	30119	119287	33.615	268	0.00
swath	278	0.60	283	4320	9450	6.401	342	2.53
vpm1	87	27.29	115	10169	2079	2.192	130	51.69
vpm2	143	38.79	159	15165	18774	7.750	322	53.56
average	695.032	29.415	691.081	12492.677	38912.065	22.344	232.726	36.180

Table 3.4: Time and gap closed on MIPLIB 3

instance	one-row only		one-row + two-row					
			one-row	two-row				
	cuts	%gc	cuts	models	+total	time	cuts	%gc
`a1c1s1`	278	36.35	219	11214	51555	12.391	529	43.86
`aflow30a`	154	19.99	170	9388	2211	3.455	92	21.73
`aflow40b`	169	11.52	179	52312	6659	22.780	112	14.30
`atlanta-ip`	9050	8.75	12993	120000	2936415	5566.283	446	8.75
`glass4`	108	0.00	178	3052	299	0.751	885	0.00
`manna81`	1980	100.00	1980	10000	503	3.316	0	100.00
`momentum1`	10043	61.03	15421	105000	5586289	3987.552	2471	57.56
`msc98-ip`	21799	53.94	25529	90000	823555	2052.142	308	53.95
`mzzv11`	18354	20.06	17285	90000	2976550	10139.824	709	20.16
`net12`	4180	9.25	4399	90928	845309	554.549	204	11.54
`nsrand-ipx`	1071	25.51	1292	44768	35315	39.955	363	29.59
`opt1217`	130	0.53	126	88	20	0.040	2	0.53
`protfold`	5651	18.24	5759	127651	145047	169.947	2006	13.65
`roll3000`	3158	67.18	2897	105000	957758	479.864	630	60.07
`sp97ar`	2585	10.79	2673	34004	11648	27.467	49	11.61
`timtab1`	555	27.20	353	12162	37555	18.771	676	47.84
`timtab2`	669	24.96	577	23686	155427	76.760	968	34.21
`tr12-30`	442	68.53	393	25800	302668	99.013	362	92.22
average	4465.3	31.32	5134.6	53058.5	826376.8	1291.9	600.7	34.53

Table 3.5: Time and gap closed on MIPLIB 2003 (instances not included in MIPLIB 3)

for being considered a separating cut, a valid inequality must also satisfy the condition on coefficient dynamism described previously. On the other hand, 86.61% of the generated one-row intersection cuts are separating (in the same *one-row + two-row* experiment), with separation taking approximately 0.14ms on average in our implementation. Recall however that two-row cuts are separated *after* one-row cuts, hence every separating two-row cut is, in the separation sense, "stronger" than all the one-row cuts generated before. Note finally that overall in our experiment, only 16.98% of the time is taken by the two-row separator, the rest being spent optimizing over the LP relaxation, computing the optimal LP tableau, and selecting pairs of rows.

The secondary objective of this experiment is to evaluate the usefulness of two-row cuts in a separation scheme. In terms of average gap closed (35.81%, compared to 29.85% with one-row intersection cuts only), the addition of two-row inequalities does seem to slightly strengthen the original formulation, without however providing a compelling argument to justify their computational cost. But it should be noted that on some instances (e.g. `misc03`, `p0201`, `p0548`, `p2756`, `pp08a`, `qnet1`, `tr12-30`), two-row cuts provide a significant improvement in the LP bound, without the addition of a disproportionate number of cuts. On a lot of other instances, the significant improvement brought by two-row cuts could be attributed to their sheer number. Remark also that in some cases, the amount of gap closed by *one-row + two-row* is smaller than with *one-row only*. This can happen since, as we do not limit ourselves to rank-1 cuts, the bases (and hence the tableaux) used in the various experiments can differ, starting from the second outer loop iteration.

Figure 3.12 illustrates the evolution of the average gap closed from one (outer loop) iteration to the next, over both instance sets. The bars labeled *one-row only* and *one-row + two-row* correspond to the gap closed after each iteration in their respective experiment. The *one-row (+ two-row)* bar corresponds to the gap closed at each iteration of the *one-row + two-row* experiment before the addition of two-row inequalities, i.e. at line 12 of Algorithm 3. Observe that while the difference between the two experiments is noticeable, one-row cuts still seem to close most of the gap in the *one-row + two-row* experiment. This might indicate that the main advantage provided by the two-row inequalities arises from obtaining useful

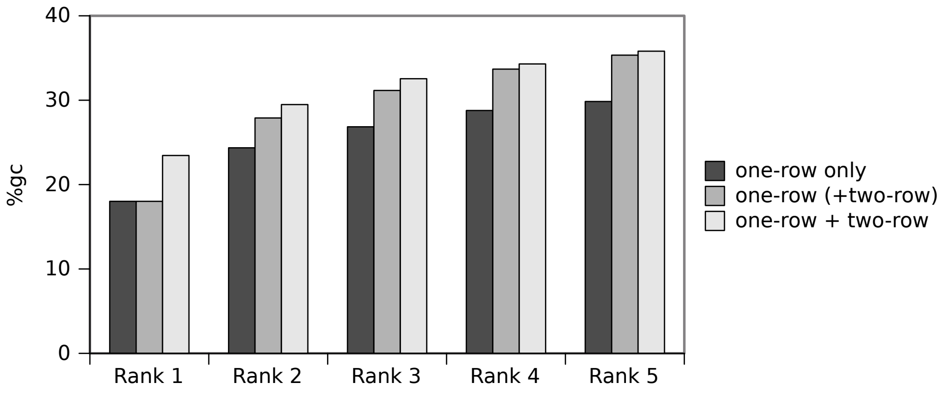

Figure 3.12: Gap closed, for different values of r

relaxations from simplex bases that are not reached adding one-row cuts only.

In Table 3.6 and Table 3.7, we survey the lattice-free sets that we obtained throughout the experiment, and classify them according to the taxonomy described in [37, 35]. We count only lattice-free sets corresponding to separating two-row cuts. One can observe that no Type-1 triangle is generated, which is worth noting as only cuts corresponding to Type-1 triangles have infinite split rank [4, 35]. This could be caused by the fact that the set P_I must take a very specific form in order for us to be able to build a Type-1 triangle with each of its vertices on $f + \text{cone}(r^i)$ for some ray r^i.

Split sets play a special role in the context of intersection cuts. In particular, any two-row intersection cut from a split set can be obtained, at a fraction of the computational cost, as a one-row cut, by combining the two initial rows. Such a cut is thus a rank-1 split cut (when constructing models from the initial LP formulation). In order to evaluate the importance of these cuts, we re-ran the experiment, discarding all two-row cuts except the ones arising from split sets. Therefore, this new experiment uses only split cuts of rank at most 5. The results are shown in Table 3.8 and Table 3.9, under the column "*+ two-row splits*". It appears that intersection cuts on split sets alone can close most of the gap closed by all two-row cuts together, going from 29.85% (*one-row only*) to 34.40% (*+ two-row splits*), instead of 35.81% (*+ two-row all*). More importantly, they do so despite the addition of fewer cuts, 135 in average (*+ two-row splits*), instead of 316 (*+ two-row all*).

3.4 Summary

In this chapter, we developed a compact formulation of the polar of the two-row model studied by Andersen et al. [4], built an algorithm to separate two-row cuts using this formulation, and showed computationally that it is fast in practice. As a result, we are able to generate separating two-row intersection cuts without fixing the underlying lattice-free set within a few milliseconds of computing time. We do not answer however the question of the practical usefulness of these cuts, as our experiments show mixed results in this regard. More precisely, for the instances tested, two-row intersection cuts close significantly more gap than one-row intersection cuts, but we can achieve almost as much when

instance	cuts	T^1	T^2	T^3	Q^1	Q^2	split
`10teams`	0	0	0	0	0	0	0
`air03`	0	0	0	0	0	0	0
`air04`	0	0	0	0	0	0	0
`air05`	0	0	0	0	0	0	0
`arki001`	238	0	55	27	116	1	39
`bell3a`	430	0	100	16	211	0	103
`bell5`	25	0	5	0	9	0	11
`blend2`	203	0	61	24	101	3	14
`cap6000`	0	0	0	0	0	0	0
`danoint`	121	0	5	8	80	6	22
`dcmulti`	370	0	19	22	255	17	57
`egout`	244	0	41	12	71	6	114
`fast0507`	0	0	0	0	0	0	0
`fiber`	401	0	107	22	194	4	74
`fixnet6`	535	0	33	14	145	9	334
`flugpl`	133	0	33	5	84	3	8
`gen`	366	0	48	20	219	11	68
`gesa2`	687	0	125	24	483	14	41
`gesa2_o`	960	0	267	46	510	11	126
`gesa3`	668	0	104	22	445	17	80
`gesa3_o`	1049	0	89	36	840	7	77
`gt2`	20	0	1	0	10	0	9
`harp2`	264	0	65	18	156	2	23
`khb05250`	43	0	6	13	17	0	7
`l152lav`	0	0	0	0	0	0	0
`lseu`	96	0	47	5	35	3	6
`markshare1`	0	0	0	0	0	0	0
`markshare2`	0	0	0	0	0	0	0
`mas74`	0	0	0	0	0	0	0
`mas76`	0	0	0	0	0	0	0
`misc03`	100	0	9	0	47	0	44
`misc06`	140	0	24	9	86	0	21
`misc07`	24	0	3	0	5	0	16
`mitre`	3141	0	1135	77	1822	21	86
`mkc`	676	0	89	23	474	19	71
`mod008`	0	0	0	0	0	0	0
`mod010`	2	0	0	0	0	0	2
`mod011`	52	0	18	3	31	0	0
`modglob`	122	0	14	8	50	4	46
`noswot`	167	0	25	3	129	0	10
`nw04`	0	0	0	0	0	0	0
`p0033`	76	0	25	5	36	0	10
`p0201`	116	0	6	2	11	5	92
`p0282`	100	0	5	5	68	3	19
`p0548`	261	0	79	5	114	0	63
`p2756`	201	0	31	12	18	1	139
`pk1`	0	0	0	0	0	0	0
`pp08a`	217	0	29	37	84	10	57
`pp08acuts`	238	0	43	13	142	16	24
`qiu`	88	0	0	3	85	0	0
`qnet1`	45	0	6	0	29	1	9
`qnet1_o`	121	0	21	4	81	4	11
`rentacar`	0	0	0	0	0	0	0
`rgn`	47	0	0	7	3	1	36
`rout`	7	0	1	0	6	0	0
`set1ch`	543	0	137	125	205	10	66
`seymour`	12	0	3	0	3	0	6
`stein27`	18	0	11	0	4	1	2
`stein45`	268	0	27	5	182	23	31
`swath`	342	0	35	25	159	6	117
`vpm1`	130	0	38	9	37	7	39
`vpm2`	322	0	45	14	179	17	67
average	232.726	0.000	49.516	11.742	130.177	4.242	37.048

Table 3.6: Lattice-free body types (MIPLIB 3)

instance	cuts	T^1	T^2	T^3	Q^1	Q^2	split
a1c1s1	529	0	20	60	92	3	354
aflow30a	92	0	17	4	55	4	12
aflow40b	112	0	27	8	69	1	7
atlanta-ip	446	0	47	5	252	46	96
glass4	885	0	4	1	1	0	879
manna81	0	0	0	0	0	0	0
momentum1	2471	0	15	1	1	0	2454
msc98-ip	308	0	11	1	250	23	23
mzzv42z	345	0	16	2	225	27	75
net12	204	0	12	0	48	14	130
nsrand-ipx	363	0	53	17	163	2	128
opt1217	2	0	0	0	0	0	2
protfold	2819	0	97	112	2186	241	183
rd-rplusc-21	327	0	49	25	82	3	168
roll3000	630	0	61	14	386	0	169
sp97ar	49	0	15	1	10	0	23
timtab1	676	0	38	73	113	9	443
timtab2	968	0	52	92	226	18	580
tr12-30	362	0	28	248	52	1	33
average	609.895	0.000	29.579	34.947	221.632	20.632	303.105

Table 3.7: Lattice-free body types (MIPLIB 2003, instances not included in MIPLIB 3)

restricting to two-row cuts from split sets, which are split cuts of the same rank. In that context, the main direction for further research is to evaluate computationally the impact of various strengthenings of the two-row relaxation, obtained by reintroducing some of the original constraints that were dropped in it.

instance	one-row only		+ two-row splits		+ two-row all	
			two-row		two-row	
	cuts	%gc	cuts	%gc	cuts	%gc
10teams	699	0.00	0	0.00	0	0.00
air03	36	100.00	0	100.00	0	100.00
air04	1299	9.49	0	9.49	0	9.49
air05	1051	6.32	0	6.32	0	6.32
arki001	163	27.28	65	27.97	238	32.23
bell3a	71	69.56	48	67.39	430	68.25
bell5	115	26.23	3	22.38	25	23.08
blend2	46	21.61	6	22.61	203	26.73
cap6000	67	54.19	0	54.19	0	54.19
danoint	100	0.43	28	0.57	121	0.61
dcmulti	278	58.32	24	66.62	370	65.64
egout	79	69.81	70	80.19	244	93.37
fast0507	1662	3.10	0	3.10	0	3.10
fiber	253	17.04	86	23.15	401	18.81
fixnet6	93	18.66	361	53.30	535	53.54
flugpl	43	14.22	1	14.19	133	20.37
gen	211	61.19	54	63.95	366	63.66
gesa2	290	47.25	50	57.52	687	70.54
gesa2_o	398	47.13	69	59.11	960	67.23
gesa3	324	49.72	92	73.18	668	74.46
gesa3_o	421	67.36	126	68.59	1049	74.62
gt2	79	97.54	5	97.54	20	99.00
harp2	130	11.85	10	12.43	264	18.53
khb05250	53	95.57	13	94.58	43	90.67
l152lav	326	15.20	0	15.20	0	15.20
lseu	80	38.00	3	35.33	96	36.89
markshare1	29	0.00	0	0.00	0	0.00
markshare2	34	0.00	0	0.00	0	0.00
mas74	74	4.38	0	4.38	0	4.38
mas76	77	3.06	0	3.06	0	3.06
misc03	275	4.56	43	17.38	100	17.36
misc06	37	63.18	9	72.54	140	86.35
misc07	392	0.72	16	0.72	24	0.72
mitre	5631	83.93	100	84.74	3141	84.45
mkc	725	39.40	133	35.29	676	26.31
mod008	33	11.22	0	11.22	0	11.22
mod010	258	57.73	2	58.84	2	58.84
mod011	22	6.87	12	6.99	52	12.41
modglob	50	28.72	70	44.22	122	48.41
noswot	163	0.00	15	0.00	167	0.00
nw04	76	17.95	0	17.95	0	17.95
p0033	34	12.77	7	56.76	76	57.01
p0201	325	25.93	92	44.47	116	45.17
p0282	182	16.03	15	11.40	100	13.54
p0548	300	50.83	61	61.98	261	66.53
p2756	264	0.89	136	47.64	201	42.12
pk1	68	0.00	0	0.00	0	0.00
pp08a	204	77.53	70	89.29	217	90.16
pp08acuts	99	47.14	32	59.93	238	60.23
qiu	116	3.05	41	8.35	88	4.64
qnet1	298	22.99	11	31.73	45	28.01
qnet1_o	152	47.84	20	47.05	121	51.54
rentacar	9	0.00	0	0.00	0	0.00
rgn	72	0.00	40	2.93	47	0.00
rout	195	7.81	0	7.81	7	9.30
set1ch	464	80.71	144	90.92	543	94.14
seymour	22038	14.75	15	13.27	12	15.01
stein27	452	0.00	19	0.00	18	0.00
stein45	1069	0.00	31	0.00	268	0.00
swath	278	0.60	120	1.48	342	2.53
vpm1	87	27.29	34	43.34	130	51.69
vpm2	143	38.79	58	52.48	322	53.56
average	695.032	29.415	39.677	34.791	232.726	36.180

Table 3.8: Two-row cuts on split sets (MIPLIB 3)

instance	one-row only		+ two-row splits		+ two-row all	
			two-row		two-row	
	cuts	%gc	cuts	%gc	cuts	%gc
a1c1s1	278	36.35	337	40.97	529	43.86
aflow30a	154	19.99	13	17.15	92	21.73
aflow40b	169	11.52	3	11.67	112	14.30
atlanta-ip	9050	8.75	53	8.74	446	8.75
glass4	108	0.00	879	0.00	885	0.00
manna81	1980	100.00	0	100.00	0	100.00
momentum1	10043	61.03	4716	61.15	2471	57.56
msc98-ip	21799	53.94	25	53.94	308	53.95
mzzv11	18354	20.06	430	21.65	709	20.16
net12	4180	9.25	179	10.08	204	11.54
nsrand-ipx	1071	25.51	166	28.54	363	29.59
opt1217	130	0.53	2	0.53	2	0.53
protfold	5651	18.24	310	16.03	2006	13.65
roll3000	3158	67.18	159	57.23	630	60.07
sp97ar	2585	10.79	45	12.93	49	11.61
timtab1	555	27.20	363	47.26	676	47.84
timtab2	669	24.96	638	35.61	968	34.21
tr12-30	442	68.53	61	71.80	362	92.22
average	4465.333	31.324	465.500	33.071	600.667	34.532

Table 3.9: Two-row cuts on split sets (MIPLIB 2003, instances not included in MIPLIB 3)

Chapter 4

Some relaxations and their best-case gap closure

In Chapter 3, we presented a method for separating facet-defining inequalities for the two-row model P_I, and evaluated the impact of adding such two-row cuts to the formulation of mixed-integer programming problems. The results suggest that their practical usefulness is limited, at least in our implementation. However, since the separator we used is exact (in that it is able to find a violated inequality whenever one exists), this tends to indicate that it is the model P_I itself that is weak. As a consequence, it is natural to look for different, stronger relaxations of the original MIP. In this chapter, we present four stronger (i.e. less relaxed) variants of P_I that have been proposed in the literature, and for which some steps have been made towards practical separation. We then perform a simple numerical experiment that aims at getting a first estimation of their potential, setting aside, for the time being, the issue of efficient cut generation. This approach can be seen as a preliminary exploration of various research directions, attempting to identify those which may yield interesting computational results.

Parts of the works in this chapter have been presented in [58].

4.1 Method

One traditional approach to cutting plane generation is to study specific relaxations of general MIPs featuring particular characteristics, to then devise ways to find valid inequalities for these relaxations. This is the approach that we adopted so far, and P_I is one such relaxation.

The best way to evaluate the computational usefulness of a relaxation is to add facet-defining inequalities for that relaxation to the initial problem formulation. We can then measure the reduction in computing time that these cuts bring to the resolution of the problem, when compensated by the time needed to compute them. However, computing the cuts is sometimes difficult, especially for multi-row models with more than two rows, for which we do not know yet a fast separation algorithm.

The alternative that we adopt here is to evaluate the relaxations relative to the initial objective function.

Given the problem

$$z_{MIP} = \min\{c^T x : x \in P\} \tag{4.1}$$

whose feasible region is the mixed-integer set P, let

$$z_{LP} = \min\{c^T x : x \in P_{LP}\} \tag{4.2}$$

be a linear relaxation, with $P = P_{LP} \cap \{x \in \mathbb{R}^n : x_j \in \mathbb{Z}, \text{ for } j \in J\}$. The integrality gap of (4.1) is defined as $z_{MIP} - z_{LP}$. Now let P_{relax} be an arbitrary relaxation of P, i.e. $P \subseteq P_{\text{relax}}$. Adding all the facet-defining inequalities of $\text{conv}(P_{\text{relax}})$ to the formulation (4.1) yields a new optimization problem whose linear relaxation is

$$z_{LP2} = \min\{c^T x : x \in P_{LP} \cap \text{conv}(P_{\text{relax}})\}. \tag{4.3}$$

Because $P \subseteq P_{LP} \cap \text{conv}(P_{\text{relax}}) \subseteq P_{LP}$, we know that $z_{LP} \leq z_{LP2} \leq z_{MIP}$. We say that the percentage of gap closed by the facet-defining inequalities of $\text{conv}(P_{\text{relax}})$ is given by

$$\%gc = 100\, \frac{z_{LP2} - z_{LP}}{z_{MIP} - z_{LP}}.$$

The gap closure can be seen as a measure of the "tightness" of the relaxation $P_{LP} \cap \text{conv}(P_{\text{relax}})$ as projected on the objective function direction c. A value $\%gc = 100$ indicates that the relaxation is "as good as" the original problem if one is interested only in the optimal objective function value z_{MIP} (and not in the optimal solution).

Gap closure as a measure has very strong limitations. A 100% gap closure does not mean that the optimal solution of (4.3) satisfies the integrality constraints, only that it has the same objective function value as the optimal solution of (4.1). Some integer problems may not even have an integrality gap (i.e. $z_{LP} = z_{MIP}$), in which case $\%gc$ is not defined, but it may be hard to find an integer solution on the optimal face of (4.2). Furthermore, for feasibility problems, the notion of objective function does not exist.

In practice however, gap closure can be taken as a broad indication, especially if averaged over a range of problem instances. The objective function value of the LP relaxation does have a meaning, because stronger LP bounds in individual nodes of a branch and bound tree enable heavier pruning, and are thus highly beneficial to the overall performance.

Optimizing over $P_{LP} \cap \text{conv}(P_{\text{relax}})$ is difficult as it implies having a description of the convex hull of the mixed-integer set P_{relax}. However, if $\text{conv}(P_{\text{relax}}) \subseteq P_{LP}$, then $P_{LP} \cap \text{conv}(P_{\text{relax}}) = \text{conv}(P_{\text{relax}})$ and (4.3) becomes

$$z_{\text{relax}} = \min\{c^T x : x \in P_{\text{relax}}\} \tag{4.4}$$

which we can solve, like the original problem, by standard MIP techniques. Unfortunately, anticipating on the specifics of the relaxations we consider (see Section 4.2), we can already state that the relationship $P_{\text{relax}} \subseteq P_{LP}$ holds for none of them, because in each we drop the bounds of some variables. But we can see $\text{conv}(P_{\text{relax}})$ as a further relaxation of $P_{LP} \cap \text{conv}(P_{\text{relax}})$. In this context, $z_{\text{relax}} = \min\{c^T x : x \in P_{\text{relax}}\}$ can be seen as a lower bound on $z_{\text{LP2}} = \min\{c^T x : x \in P_{LP} \cap \text{conv}(P_{\text{relax}})\}$. We know that $z_{\text{relax}} \leq z_{MIP}$ because $P_{\text{relax}} \supseteq P$, and Proposition 12 shows that $z_{\text{relax}} \geq z_{LP}$, even though $P_{\text{relax}} \not\subseteq P_{LP}$.

Proposition 12. *Let P_{relax} be a relaxation of P in which all the constraints dropped from the linear relaxation P_{LP} are nonbinding at an optimal point of $\min\{c^T x : x \in P_{LP}\}$. Then*

$$\min\{c^T x : x \in P_{relax}\} \geq \min\{c^T x : x \in P_{LP}\},$$

i.e. the optimization problem over P_{relax} has an optimal objective function value no smaller than over P_{LP}.

Proof. Let us consider the relaxation $P_{LP'}$ of P_{LP} which consists in dropping from P_{LP} all the constraints that are not binding at its optimal solution (see Table 4.3). Since these constraints are not binding, the optimal solution over P_{LP} is still optimal if optimizing over $P_{LP'}$. As $P_{LP'}$ does not have integrality constraints, it is easy to see that it is a further relaxation of P_{relax}, i.e. $P_{LP'} \supseteq P_{\text{relax}}$. $\square$

We will see in Section 4.2 that, although Proposition 12 seems to restrict to very specific types of relaxations, it applies to all those we consider.

In this chapter, for all five relaxations, we report the gap closure corresponding to z_{relax}, defined as $\%gc = 100\,(z_{\text{relax}} - z_{LP})/(z_{MIP} - z_{LP})$, which is comprised between 0 and 100 and is a lower bound on the percentage of gap closed by adding all cuts from the associated model. A limitation of the technique is that these cuts would be m-row cuts derived from one single m-row model, where m is the number of rows of the original problem. An interesting information would be provided if we could obtain the gap closed by cuts from several k-row models with $k < m$. However, this would involve computing $z = \min\{c^T x : x \in P_{LP} \cap \text{conv}(P_{\text{relax }1}) \cap \cdots \cap \text{conv}(P_{\text{relax }T})\}$ which is even harder than $\min\{c^T x : x \in P_{LP} \cap \text{conv}(P_{\text{relax}})\}$.

In [41], Fischetti and Monaci adopted a similar approach and performed computations for the corner and the strict corner relaxations (and others). We extend these results with three more relaxations. We present in the next section a quick survey of the relaxations that we are interested in.

4.2 Models

We study five types of relaxations of the mixed-integer feasible region P, all of them function of a basis of the linear programming relaxation.

The first relaxation is the m-row version of the intersection cut model P_I

$$P_I = \{(x, s) \in \mathbb{Z}^m \times \mathbb{R}^n_+ : x = f + Rs\}$$

introduced by Balas in [7], and further characterized in the two-row case (i.e. for $m = 2$) by Andersen, Louveaux, Weismantel and Wolsey [4] and Cornuéjols and Margot [29]. This is the model, with $m = 2$, for which we developed an exact separator in Chapter 3. As we saw there, it consists in dropping from P integrality constraints on nonbasic variables and bounds on basic variables. Note that only one bound on the nonbasic variables is included in the description of P_I ($s \geq 0$), while MIP instances typically come in a so-called *computational* form with possibly both lower and upper bounds on the variables. However,

Computational form $0 \le x \le U$	Standard form $\bar{x}, \bar{s} \ge 0,\ \bar{x} + \bar{s} = U$	
variable x	variable $\bar{x}$	slack $\bar{s}$
basic	basic	basic
nonbasic at lower bound	nonbasic	basic
nonbasic at upper bound	basic	nonbasic

Table 4.1: Computational form and standard form

we can build any desired relaxation directly in computational form by implicitly considering a problem in standard form where a slack is added for each upper bound constraint. Proposition 13 shows how this is handled. In practice, for P_I, this amounts to also dropping the non-binding bound of each nonbasic variable.

Proposition 13. *In a problem in computational form, any variable x_i that has a lower bound 0 and an upper bound U_i corresponds, in standard form, to a nonnegative variable $\bar{x}_i \ge 0$ and a nonnegative slack $\bar{s}_i \ge 0$. When x_i is basic, both $\bar{x}_i$ and $\bar{s}_i$ are basic. When x_i is nonbasic at lower bound, $\bar{x}_i$ is nonbasic and $\bar{s}_i$ is basic. When x_i is nonbasic at upper bound, $\bar{x}_i$ is basic and $\bar{s}_i$ is nonbasic. See Table 4.1.*

Proof. This follows from incorporating the upper bound in the constraints of the problem, and explicitly writing the slack $\bar{s}_i \ge 0$, yielding $\bar{x}_i + \bar{s}_i = U_i$. When x_i is basic, both $\bar{x}_i$ and $\bar{s}_i$ may take nonzero values, so they must be basic. In order to set $x_i = \bar{x}_i$, we set $\bar{x}_i$ to be basic in the same row as x_i, and $\bar{s}_i$ to be basic in the explicit upper bound constraint. Similarly, when x_i is at lower bound, $\bar{s}_i$ must be basic in the upper bound constraint, and $\bar{x}_i$ must be nonbasic in order to not affect the basis status of the other variables. The same reasoning yields the result for x_i at upper bound. □

The other four relaxations consist in various strengthenings of the model P_I built by reintroducing some of the constraints dropped from P. In particular, Dey and Wolsey [38], Basu, Conforti, Cornuéjols and Zambelli [20] and Fukasawa and Günlük [45] considered ways to exploit finite bounds on the basic variables x. Specifically, Dey and Wolsey [38] extend the geometric intuition that is characteristic of intersection cuts by introducing the concept of S-free set. An S-free set is a convex set that does not contain any element of S in its interior. Using the same formula as for the intersection cuts, one can build a cut from an S-free set, instead of a lattice-free (i.e. $\mathbb{Z}^m$-free) set, where S is the feasible region of the x variables. They show that the resulting inequality is valid for the strengthened model

$$P_{S\text{-free}} = \{(x, s) \in S \times \mathbb{R}^n_+ : x = f + Rs\}$$

if S is the set of integral points in some rational polyhedron. This condition is not restrictive because, in practice, we define S to be the intersection of $\mathbb{Z}^m$ with the bound constraints on x. Furthermore, all valid inequalities for $P_{S\text{-free}}$ can be obtained through such intersection cuts [38], and there is a one-to-one correspondence between minimal inequalities for the infinite relaxation of $P_{S\text{-free}}$

$$\{(x, s) \in S \times \mathbb{R}^\infty_+ : x = f + \sum_{r \in \mathbb{R}^m} r s_r,\ s \text{ has a finite support}\}$$

and maximal S-free sets [20].

On the other hand, Dey and Wolsey [37, 33] and Conforti, Cornuéjols and Zambelli [27] considered the problem of exploiting any integrality constraint on non-basic variables. They propose a way to, given a valid intersection cut for P_I, strengthen the coefficients of the nonbasic integer variables by solving a so-called *lifting* problem. Assuming, for simplicity, that P is a pure integer set, the lifting yields valid inequalities for

$$\{(x, s) \in \mathbb{Z}^m \times \mathbb{Z}^n_+ : x = f + Rs\}. \tag{4.5}$$

Dey and Wolsey [37, 33] show, in the two-row case, that these inequalities are *extreme* for the infinite model

$$\{(x, s) \in \mathbb{Z}^2 \times \mathbb{Z}^\infty_+ : x = f + \sum_{r \in \mathbb{R}^2} r s_r,\ s \text{ has a finite support}\} \tag{4.6}$$

if the initial inequality was extreme for the infinite relaxation of P_I. The lifting is unique if the initial intersection cut arises from specific types of lattice-free sets (namely Type-1 and Type-2 triangles [37, 33]) and sequence-dependent otherwise. This has been extended with analogous results in the multi-row and S-free cases by Conforti, Cornuéjols and Zambelli [27].

Note that (4.5) is known as a *corner relaxation* of P. The corner relaxation was studied in details by Gomory [49, 50] and Gomory and Johnson [51], and is obtained from P by dropping bounds on all basic variables. One interesting variant of the corner relaxation, referred to as the *strict* corner relaxation, is obtained from P by only dropping those bounds on basic variables that are not tight. In standard form, it corresponds to dropping the nonnegativity constraint of variables that are not zero in the optimal LP solution. The strict corner relaxation differs from the corner relaxation only when primal degeneracy exists in the optimal basis. It is the intersection of the corner relaxations for all bases corresponding to one optimal vertex. As most MIP formulations are highly degenerate, the strict corner provides insights on the role played by degeneracy in the LP formulation.

Lastly, one can take advantage of the presence of upper bounds on nonbasic variables. Andersen, Louveaux and Weismantel [2] tackled that case, taking into account an upper bound on one nonbasic variable. This can be seen as considering at a pair of two-row models arising from adjacent simplex bases. They describe geometrically the facet structure of that relaxation, showing that an additional class of facet-defining valid inequalities then arises, whose coefficients can be read from pentagons in $\mathbb{R}^2$, with formula that they develop. Although the problem has not been thoroughly studied for more than one upper bound, we will consider here the fully-strengthened model

$$P_{IU} = \{(x, s) \in \mathbb{Z}^m \times \mathbb{R}^n_+ : x = f + Rs, s \leq U\}.$$

As a consequence, we are probably further away from a practical implementation of a separator for that particular model than for the other ones.

Table 4.2 presents a summary of the constraints that are dropped in each of the relaxations. The constraints represented are the integrality constraints (denoted "$\in \mathbb{Z}$"), and bound constraints (denoted "bnd."). These constraints apply on the variables, which are themselves divided in categories depending on whether they are basic or nonbasic, and on whether they are integral-constrained ("int.") or not ("cont." for continuous). Constraints that are kept from the original formulation are denoted by "$\surd$", while those that are dropped are marked "$\times$". Some bound constraints are kept only if they are tight at the vertex corresponding to the current basis, in which case they are marked "T".

	basic			nonbasic	
	int.		cont.		
	$\in \mathbb{Z}$	bnd.	bnd.	$\in \mathbb{Z}$	bnd.
P_I	$\surd$	$\times$	$\times$	$\times$	T
S-free	$\surd$	$\surd$	$\times$	$\times$	T
lifting	$\surd$	$\times$	$\times$	$\surd$	T
strict	$\surd$	T	$\times$	$\surd$	T
P_{IU}	$\surd$	$\times$	$\times$	$\times$	$\surd$

Table 4.2: Constraints from the original problem; $\surd$: keep, `T`: keep if tight, $\times$: drop

	basic			nonbasic	
	int.		cont.		
	$\in \mathbb{Z}$	bnd.	bnd.	$\in \mathbb{Z}$	bnd.
P_{LP}	$\times$	$\surd$	$\surd$	$\times$	$\surd$
$P_{LP'}$	$\times$	$\times$	$\times$	$\times$	T

Table 4.3: P_{LP} and $P_{LP'}$; $\surd$: keep, `T`: keep if tight, $\times$: drop

4.3 Results

The gap closure corresponding to each relaxation is presented in Table 4.4 for the instances from the MIPLIB 3 [23] and in Table 4.5 for the instances from the MIPLIB 2003 [1]. The instances for which all the computations succeeded are grouped in Table 4.6. The problems `dsbmip` and `enigma` are not shown because they have no integrality gap. The problems `dano3mip`, `liu`, `momentum3` and `t1717` are not considered either because no optimal solution is known for them. The computations were performed with CPLEX 12.3 on a machine with an Intel Core i7-990 CPU. The time limit was fixed at four hours per problem. The absence of a result when the time limit is hit is denoted by a "?" in the tables.

It appears on Table 4.6 that the relaxation "S-free" making use of bounds on basic variables is the strengthening of P_I that has the most impact on gap closure. However, as $P_{S\text{-free}}$ exploits the bounds on all m basic variables, it is potentially much stronger than a strengthened k-row model, where k is typically smaller than m. Furthermore, "S-free", as well as "P_{IU}", strengthen P_I by adding bound constraints that are also present in the LP relaxation. As we noted before, each relaxation should be intersected with the LP relaxation in an exact computation, but we do not do that for practical reasons. Although $\text{conv}(P_{LP} \cap P_{\text{relax}})$ is different from $\text{conv}(P_{LP}) \cap \text{conv}(P_{\text{relax}})$, the contribution of bound constraints in $P_{S\text{-free}}$ and P_{IU} may be overemphasized as some of that contribution might be achieved by intersecting with P_{LP}. As a result, the strengthenings "lifting" and "strict" may be at a disadvantage over "S-free" and "P_{IU}".

More significantly, we can observe that on average, exploiting all the bounds on nonbasic variables yields little improvement on gap closure, despite the fact that, as mentioned earlier, this improvement is likely overestimated. Therefore, although we developed in Section 2.5 an expression of the polar of P_{IU}, this would tend to indicate that cuts from valid inequalities for P_{IU} are less promising computationally than lifted cuts from valid inequalities for P_I.

While variations can occur due to different LP solvers yielding different optimal bases, the overlapping

instance	P_I	S-free	lifting	strict	P_{IU}
10teams	0.00	0.00	?	?	0.00
air03	100.00	100.00	100.00	100.00	100.00
air04	1.76	100.00	1.26	4.58	4.91
air05	2.29	100.00	2.29	4.51	3.10
arki001	4.54	12.84	?	?	?
bell3a	60.13	83.79	83.79	83.79	60.13
bell5	4.10	21.87	4.40	4.40	4.10
blend2	37.98	74.70	37.98	37.98	59.46
cap6000	18.44	60.38	33.06	33.06	43.45
danoint	1.91	2.90	1.74	1.74	1.74
dcmulti	?	60.63	?	23.98	?
egout	100.00	100.00	100.00	100.00	100.00
fast0507	?	?	46.08	46.08	?
fiber	?	11.24	72.97	?	6.55
fixnet6	78.87	79.40	78.87	78.87	78.87
flugpl	97.38	100.00	97.38	97.38	97.38
gen	38.00	48.37	?	?	38.00
gesa2_o	32.41	33.83	32.55	32.46	32.35
gesa2	31.76	33.09	32.38	32.16	31.76
gesa3_o	49.20	52.95	49.20	51.93	49.42
gesa3	45.71	46.34	49.20	49.83	45.71
gt2	46.79	100.00	46.79	46.79	57.12
harp2	10.97	31.35	?	?	20.82
khb05250	58.60	58.60	62.07	62.07	58.60
l152lav	2.50	70.29	2.50	14.69	4.02
lseu	?	25.79	?	?	8.16
markshare1	0.00	0.00	?	?	0.00
markshare2	0.00	0.00	?	?	0.00
mas74	9.19	9.19	?	?	9.19
mas76	7.73	9.70	?	?	7.85
misc03	?	100.00	7.24	17.59	7.24
misc06	13.74	0.00	6.04	6.04	6.04
misc07	0.00	54.88	?	0.72	0.00
mitre	79.25	80.70	?	?	?
mkc	?	13.27	?	?	0.00
mod008	2.85	3.42	50.22	56.44	3.76
mod010	18.34	100.00	18.34	100.00	31.95
mod011	3.49	5.03	3.49	3.49	3.49
modglob	28.21	28.66	31.29	31.29	28.21
noswot	0.00	0.00	0.00	0.00	0.00
nw04	3.87	4.90	39.78	51.75	4.11
p0033	0.45	5.27	?	?	0.50
p0201	?	60.81	?	?	?
p0282	6.70	79.11	8.90	9.28	7.71
p0548	0.01	18.82	0.02	0.02	0.01
p2756	0.15	2.50	0.29	0.29	0.15
pk1	0.00	0.00	0.00	0.00	0.00
pp08acuts	?	45.92	?	?	32.75
pp08a	?	66.10	?	?	?
qiu	0.00	0.00	0.00	0.00	0.00
qnet1_o	28.69	77.10	28.71	64.80	43.42
qnet1	13.72	79.45	13.72	64.31	15.06
rentacar	0.00	0.00	0.00	0.00	0.00
rgn	0.00	0.00	0.00	0.00	0.00
rout	?	5.19	?	?	?
set1ch	?	99.98	?	?	?
seymour	?	?	6.02	?	?
stein27	0.00	100.00	0.00	0.00	0.00
stein45	0.00	100.00	0.00	0.00	0.00
swath	?	3.70	31.95	34.42	3.41
vpm1	?	56.36	?	?	78.18
vpm2	?	54.67	?	?	?

Table 4.4: Results on MIPLIB 3

instance	P_I	S-free	lifting	strict	P_{IU}
a1c1s1	24.42	24.74	24.42	24.42	24.42
aflow30a	14.20	26.22	14.20	14.20	17.64
aflow40b	15.61	20.53	15.61	15.61	16.84
atlanta-ip	?	?	?	?	?
ds	?	8.76	?	?	?
glass4	0.00	0.00	0.00	0.00	0.00
manna81	?	100.00	100.00	100.00	100.00
momentum1	?	61.59	57.09	?	57.08
momentum2	?	68.30	?	?	?
msc98-ip	31.58	50.16	?	32.54	31.58
mzzv11	3.85	100.00	3.85	10.37	4.66
mzzv42z	?	100.00	?	10.43	?
net12	?	52.73	?	?	?
nsrand-ipx	0.00	?	?	?	0.12
opt1217	?	0.53	?	?	0.53
protfold	8.74	?	8.74	8.74	?
rd-rplusc-21	0.00	0.00	0.00	0.00	0.00
roll3000	?	29.16	?	?	3.90
sp97ar	?	?	?	?	?
stp3d	?	?	?	?	?
timtab1	?	70.10	?	?	82.06
timtab2	?	?	?	?	?
tr12-30	?	?	?	?	?

Table 4.5: Results on MIPLIB 2003

instance	P_I	S-free	lifting	strict	P_{IU}
air03	100.00	100.00	100.00	100.00	100.00
air04	1.76	100.00	1.26	4.58	4.91
air05	2.29	100.00	2.29	4.51	3.10
bell3a	60.13	83.79	83.79	83.79	60.13
bell5	4.10	21.87	4.40	4.40	4.10
blend2	37.98	74.70	37.98	37.98	59.46
cap6000	18.44	60.38	33.06	33.06	43.45
danoint	1.91	2.90	1.74	1.74	1.74
egout	100.00	100.00	100.00	100.00	100.00
fixnet6	78.87	79.40	78.87	78.87	78.87
flugpl	97.38	100.00	97.38	97.38	97.38
gesa2_o	32.41	33.83	32.55	32.46	32.35
gesa2	31.76	33.09	32.38	32.16	31.76
gesa3_o	49.20	52.95	49.20	51.93	49.42
gesa3	45.71	46.34	49.20	49.83	45.71
gt2	46.79	100.00	46.79	46.79	57.12
khb05250	58.60	58.60	62.07	62.07	58.60
l152lav	2.50	70.29	2.50	14.69	4.02
misc06	13.74	0.00	6.04	6.04	6.04
mod008	2.85	3.42	50.22	56.44	3.76
mod010	18.34	100.00	18.34	100.00	31.95
mod011	3.49	5.03	3.49	3.49	3.49
modglob	28.21	28.66	31.29	31.29	28.21
noswot	0.00	0.00	0.00	0.00	0.00
nw04	3.87	4.90	39.78	51.75	4.11
p0282	6.70	79.11	8.90	9.28	7.71
p0548	0.01	18.82	0.02	0.02	0.01
p2756	0.15	2.50	0.29	0.29	0.15
pk1	0.00	0.00	0.00	0.00	0.00
qiu	0.00	0.00	0.00	0.00	0.00
qnet1_o	28.69	77.10	28.71	64.80	43.42
qnet1	13.72	79.45	13.72	64.31	15.06
rentacar	0.00	0.00	0.00	0.00	0.00
rgn	0.00	0.00	0.00	0.00	0.00
stein27	0.00	100.00	0.00	0.00	0.00
stein45	0.00	100.00	0.00	0.00	0.00
a1c1s1	24.42	24.74	24.42	24.42	24.42
aflow30a	14.20	26.22	14.20	14.20	17.64
aflow40b	15.61	20.53	15.61	15.61	16.84
glass4	0.00	0.00	0.00	0.00	0.00
mzzv11	3.85	100.00	3.85	10.37	4.66
rd-rplusc-21	0.00	0.00	0.00	0.00	0.00
Average	22.56	47.35	25.58	30.68	24.75

Table 4.6: Results on MIPLIB 3 and MIPLIB 2003 – instances with no missing data

results (i.e. “lifting” and “strict”) are mostly consistent with what Fischetti and Monaci reported in [41]. Note that, in [41], the relaxation “lifting” is referred to as “Group” and “strict” is labeled “Corner”.

Chapter 5

Separation over arbitrary mixed-integer sets

In this chapter, we aim at evaluating precisely the strength of the multi-row relaxations that are used to generate cutting planes. It is the same broad objective as in Chapter 4, except that instead of approximate indications about families of relaxations, we now want an accurate assessment of the usefulness of the derived cutting planes. Indeed, the strong limitations to the generality of the approach adopted in Chapter 4 were a trade-off for speed and simplicity, because we wanted to avoid the costly process of computing the cuts.

Here instead, we develop techniques to compute the cuts derived from any given relaxation. In other words, we consider the question of separating valid inequalities from arbitrary mixed-integer sets.

The study is computational in that we take our models from libraries of mixed-integer problem instances that are widely used for benchmarking purposes, then numerically generate cuts for these models. It is also theoretical in that we do not compute cuts to solve problems faster overall, but rather to evaluate whether they could help solve problems faster *if* we had an efficient separator for the associated relaxation. In this perspective, we build a generic separator to identify those relaxations for which trying to develop a specific separator would be worthwhile.

The content of this chapter is joint work with Domenico Salvagnin.

5.1 Previous works

In this section, we present some previous works that are related to the subject of this chapter. They tackle the computational side of the study of multi-row cuts.

Espinoza [39] performed the first extensive computational tests with multi-row intersection cuts. He uses three types of fixed lattice-free sets in $\mathbb{R}^m$. The first is a maximal lattice-free simplex with integer vertices, the m-dimensional extension of Type-1 triangles (see Section 1.8). The second is a cross polytope, scaled and translated to contain the 0-1 hypercube, so as to also be a maximal lattice-free set. The third is the m-dimensional extension of a specific Type-2 triangle. He exploits the integrality of the nonbasic variable through a heuristic method similar to Dey and Wolsey's lifting [33], although without the guarantee of generating facet-defining inequalities for the corresponding strengthened model. He tested the three types of lattice-free sets separately for $m \in \{1, \dots, 10\}$, on MIP instances from MIPLIB 3 [23] and MIPLIB 2003 [1]. The results of these experiments are promising in that they show that the addition of multi-row cuts can speed-up the overall branch-and-cut by up to 5% in geometric mean over the testset, for specific types of cuts and values of m. However, they also show mixed results in general (i.e. without a priori knowledge of which types of cuts will be successful), with in most cases a slight increase in computing time over the solver's defaults, in geometric mean over the testset.

Dey, Lodi, Tramontani and Wolsey [34] focused on two-row intersection cuts from Type-2 triangles. They devise a heuristic method for building a Type-2 triangle that is parametric in the model data. The aim is to exploit the problem structure, namely the values of f and R in $P_I = \{(x, s) \in \mathbb{Z}^2 \times \mathbb{R}^n_+ : x = f + Rs\}$, to generate strong cuts that are, in some way, "different" from GMI cuts. The cuts are then strengthened with an approximation of the lifting method of Dey and Wolsey [33], to take advantage of the integrality of nonbasic variables. The method is tested on modified versions of the randomly generated multidimensional knapsack instances of Atamtürk [6]. From these experiments, they conclude that some two-row cuts do provide an advantage over GMI cuts, although few of the many two-row cuts generated are eventually helpful. This result emphasizes the importance of the selection of the two-row cuts. Furthermore, their data show that in the presence of many nonbasic integer variables, the performance of both two-row and GMI cuts deteriorates, suggesting a need for different relaxations in that case. To the contrary, the presence in the original problem of the bounds that are dropped in P_I did not seem to significantly impair the effectiveness of the two-row cuts (compared to when no such bounds are present).

Basu, Bonami, Cornuéjols and Margot [18] study the case of two-row models P_I for which one of the components of f is integral. This situation typically arises when the optimal basis of the LP relaxation is degenerate, which is frequent in MIP formulations. They present a heuristic algorithm for generating maximal lattice-free Type-1 and Type-2 triangles for such model. They show how to compute an intersection cut from these triangles, and develop a closed-form formula implementing Dey and Wolsey's lifting [33]. Note that the latter formula applies to all Type-1 and Type-2 triangles, even when both components of f are fractional. An alternative closed-form formula is also presented, that exploits nonnegativity of one of the basic variables. Experiments are performed on instances from a slight modification of MIPLIB 3 [61], measuring the percentage of gap closed by the two-row cuts. They conclude that the family of cuts that they test is effective, but not competitive with GMI cuts.

In [59], we present an exact separator for the two-row intersection cut model P_I. Chapter 3 covers the results of our computational experiments. We conclude that two-row intersection cuts close significantly more gap than one-row intersection cuts, but we can achieve almost as much when restricting to two-row cuts from split sets, which can be obtained by reading the GMI from a linear combination of the two rows.

The work of Fukasawa and Goycoolea [44] does not take place in the context of multi-row cuts. However, it is very related to the content of this chapter in that they develop an exact separator for the mixed-integer knapsack (i.e. one-row) model

$$P = \{x \in \mathbb{R}^n : a^t x \leq b,\, l \leq x \leq u,\, x_j \in \mathbb{Z}, \forall j \in J\}.$$

Part of the techniques they develop to speed up separation translate directly to our problem. In particular, they also deploy row-generation to optimize over the polar set of P (see Section 5.4), and they lift valid inequalities from tight bounds (see Section 5.5). The objective of their computations is to compare a specific type of cuts alone to all so-called knapsack cuts, i.e. to all valid inequalities for P together. The specific cuts they study are obtained through the mixed-integer rounding procedure (MIR [67]), which when applied to tableau rows simply yields GMIs. Their results show that in terms of strengthening of the LP bound, MIR cuts alone achieve almost as much as knapsack cuts.

A constant among these results is that they underline the impressive power of the classic GMI cuts and their variants. This raises the following question: How strong does a relaxation have to be for the cuts it yields to significantly outperform GMIs? This is one of the questions we will tackle in this chapter.

5.2 Polarity for general polyhedra

We now set aside the question of the relaxation, and focus on the problem of separating cuts for any given relaxation. Let P_J be a mixed-integer set

$$P_J = \{x \in \mathbb{R}^n : x \in P_{LP},\ x_j \in \mathbb{Z} \text{ for all } j \in J\} \tag{5.1}$$

where P_{LP} is a polyhedron in $\mathbb{R}^n$ and J is a subset of $N = \{1, \ldots, n\}$. As in Chapter 2, we start by studying the polar of $\operatorname{conv}(P_J)$, establishing the well-known properties related to polarity in the context of general polyhedra. We refer the interested reader to [71] for more details. For conciseness, in the remainder of this section, we denote $P := \operatorname{conv}(P_J)$.

Since we want the polar to describe the valid inequalities for P, a natural way to define it would be with a set such as

$$\{(\alpha, \alpha_0) \in \mathbb{R}^{n+1} : \alpha^T x \geq \alpha_0, \text{ for all } x \in P\}. \tag{5.2}$$

An issue with the set in (5.2) is that it is defined in $\mathbb{R}^{n+1}$ for a polyhedron P in $\mathbb{R}^n$, preventing us from having the simplifying symmetry that we had in Chapter 2, i.e. that a polyhedron is itself the polar of its polar.

A simple way to overcome this issue is, noting that (5.2) always describes a polyhedral cone, to build a "conified" variant P^+ of P such that the descriptions of P and P^+ are interchangeable in a trivial way.

Definition 29. *Given a polyhedron P, by the Minkowski-Weyl theorem, there exist finite sets $\mathcal{V}$, $\mathcal{R}$, and $\mathcal{L}$ such that*

$$P = \operatorname{conv}(\mathcal{V}) + \operatorname{cone}(\mathcal{R}) + \operatorname{lin}(\mathcal{L}).$$

Let $\mathcal{V}^+ := \{(v, -1) \in \mathbb{R}^{n+1} : v \in \mathcal{V}\}$, $\mathcal{R}^+ := \{(r, 0) \in \mathbb{R}^{n+1} : r \in \mathcal{R}\}$, and $\mathcal{L}^+ := \{(l, 0) \in \mathbb{R}^{n+1} : l \in \mathcal{L}\}$. We define P^+ as

$$P^+ := \operatorname{cone}\{\mathcal{V}^+, \mathcal{R}^+\} + \operatorname{lin}\{\mathcal{L}^+\}. \tag{5.3}$$

By construction, P^+ is a polyhedral cone generated by the rays $\mathcal{V}^+$, $\mathcal{R}^+$, $\mathcal{L}^+$ and $-\mathcal{L}^+$ (Figures 5.1 and 5.2).

Proposition 14 (Link between the inner descriptions of P and P^+)**.**

(a). v is a vertex of P if and only if $(v, -1)$ is an extreme ray of P^+.

(b). r is an extreme ray of P if and only if $(r, 0)$ is an extreme ray of P^+.

(c). l is in the lineality space of P if and only if $(l, 0)$ is in the lineality space of P^+.

Proof. (a). The point v is a vertex of P iff $v \in \mathcal{V}$ and there do not exist coefficients λ, μ and ν such that

$$v = \sum_{x \in \mathcal{V}\setminus\{v\}} \lambda_x x + \sum_{r \in \mathcal{R}} \mu_r r + \sum_{l \in \mathcal{L}} \nu_l l, \qquad \text{with } \lambda_i, \mu_i \geq 0 \text{ for all } i \text{ and } \sum_i \lambda_i = 1.$$

That is, iff $(v, -1) \in \mathcal{V}^+$ and there do not exist coefficients λ, μ and ν such that

$$\begin{array}{rcl} v & = & \displaystyle\sum_{x \in \mathcal{V}\setminus\{v\}} \lambda_x x \;+\; \sum_{r \in \mathcal{R}} \mu_r r \;+\; \sum_{l \in \mathcal{L}} \nu_l l, \\ -1 & = & \displaystyle\sum_{x \in \mathcal{V}\setminus\{v\}} \lambda_x (-1) \;+\; \sum_{r \in \mathcal{R}} \mu_r 0 \;+\; \sum_{l \in \mathcal{L}} \nu_l 0, \end{array} \qquad \text{with } \lambda_i, \mu_i \geq 0 \text{ for all } i.$$

The latter condition expresses that $(v, -1)$ is a vertex of P^+. (b) and (c) are shown in the same way. □

Lemma 12. *$(\alpha, \alpha_0) \in \mathbb{R}^{n+1}$ describes a valid inequality $\alpha^T x \geq \alpha_0$ for P if and only if $\alpha^T x + \alpha_0 x_0 \geq 0$ is valid for P^+.*

Proof. The inequality $\alpha^T x \geq \alpha_0$ is valid for P iff (a) $\alpha^T v \geq \alpha_0$ for all $v \in \mathcal{V}$, (b) $\alpha^T r \geq 0$ for all $r \in \mathcal{R}$, and (c) $\alpha^T l = 0$ for all $l \in \mathcal{L}$. These three conditions are respectively equivalent to (a') $\alpha^T v + v_0 \alpha_0 \geq 0$ for all $(v, v_0) \in \mathcal{V}^+$, (b') $\alpha^T r + r_0 \alpha_0 \geq 0$ for all $(r, r_0) \in \mathcal{R}^+$, and (c') $\alpha^T l + l_0 \alpha_0 = 0$ for all $(l, l_0) \in \mathcal{L}^+$. The latter conditions are met iff $\alpha^T x + \alpha_0 x_0 \geq 0$ is valid for P^+. □

Proposition 15 (Link between the outer descriptions of P and P^+)**.** *$(\alpha, \alpha_0) \in \mathbb{R}^{n+1}$ describes a facet-defining inequality $\alpha^T x \geq \alpha_0$ for P if and only if $\alpha^T x + \alpha_0 x_0 \geq 0$ is facet-defining for P^+.*

Proof. A valid inequality (α, α_0) is facet-defining for P iff it can not be expressed as a conic combination of valid inequalities (β^i, β^i_0) for P, all distinct from (α, α_0), i.e. $(\beta^i, \beta^i_0) \neq (\lambda\alpha, \lambda\alpha_0)$ for all i and $\lambda > 0$. Given Lemma 12, the same statement holds for (α, α_0) to be facet-defining for P^+. □

Corollary 4. *$P = \{x \in \mathbb{R}^n : (x, -1) \in P^+\}$, i.e. $P = \mathrm{proj}_x(P^+ \cap \{x_0 = -1\})$.*

Proof. Consider the set of all valid inequalities $\alpha^T x + \alpha_0 x_0 \geq 0$ for P^+. Fixing $x_0 = 1$ and projecting it out, one obtains, by Lemma 12, the set of all valid inequalities for P. □

We next define the polar Q of a polyhedral cone $C \subseteq \mathbb{R}^m$. Note that we are mainly interested in the case $C = P^+$ with $m = n + 1$.

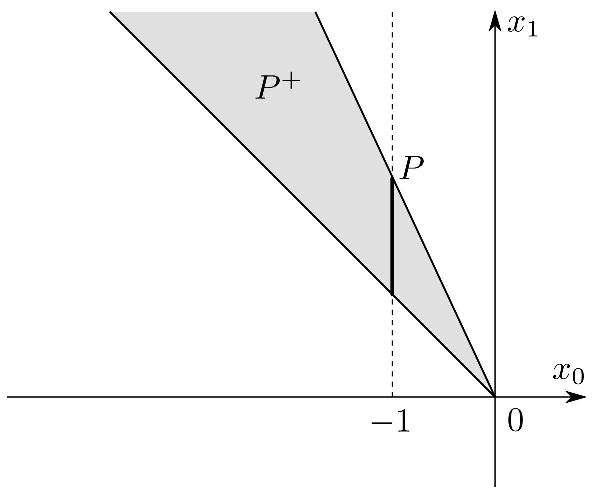

Figure 5.1: $P = \operatorname{conv}\{1, 2\}$

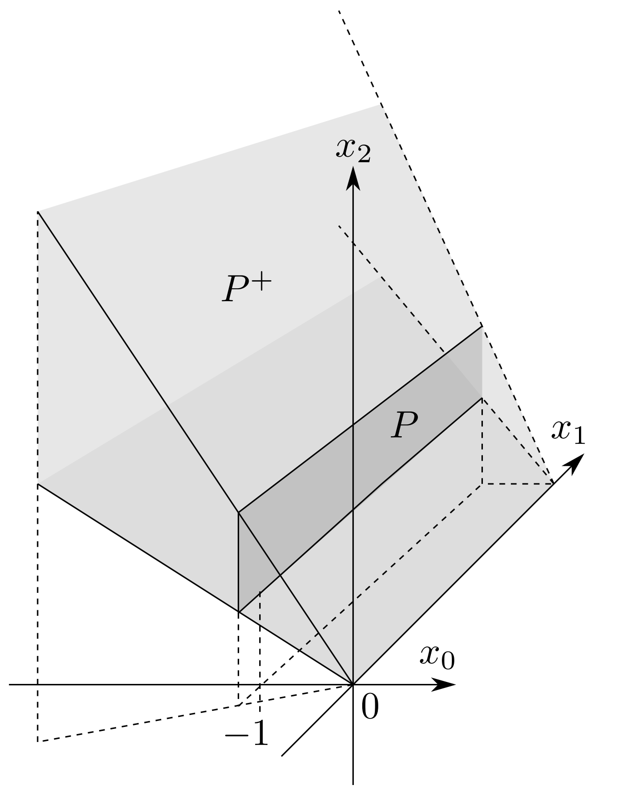

Figure 5.2: $P = \operatorname{conv}\left\{\begin{pmatrix} -\frac{1}{4} \\ 1 \end{pmatrix}, \begin{pmatrix} -\frac{1}{4} \\ 2 \end{pmatrix}\right\} + \operatorname{cone}\left\{\begin{pmatrix} 1 \\ 0 \end{pmatrix}\right\}$

Lemma 13. *If $\alpha^T x \geq -1$ is a valid inequality for C, then $\alpha^T x \geq 0$ is also valid for C.*

Proof. Let $y \in C$ be such that $-1 \leq \alpha^T y < 0$. Because C is a cone, the point $z := \frac{-2}{\alpha^T y} y$ belongs to C. But $\alpha^T z = \frac{-2}{\alpha^T y}\alpha^T y = -2 \not\geq -1$. Therefore, since $\alpha^T x \geq -1$ is valid, there does not exist $y \in C$ such that $\alpha^T y < 0$. □

Proposition 16. *Any valid inequality for a polyhedral cone C that defines a proper face of C is of the form $\alpha^T x \geq 0$*

Proof. An inequality of the form $\alpha^T x \geq 1$ can not be valid for C as it cuts the origin 0. By Lemma 13, an inequality of the form $\alpha^T x \geq -1$ can not define a proper face of C. □

Definition 30. *The polar of a polyhedral cone $C \subseteq \mathbb{R}^m$ is given by*

$$Q := \left\{\alpha \in \mathbb{R}^m : \alpha^T x \geq 0,\ \textit{for all } x \in C\right\}.$$

Any polyhedral cone C can be represented as

$$C = \text{cone}(\mathcal{S}) + \text{lin}(\mathcal{T})$$

where $\mathcal{S}$ is a finite set of rays of C and $\mathcal{T}$ is a set of vectors forming a basis of its lineality space. We assume without loss of generality that the set $\mathcal{S}$ is a minimal generating set, i.e. that $\text{cone}(\mathcal{S} \setminus \{y\}) + \text{lin}(\mathcal{T}) \neq C$ for any $y \in \mathcal{S}$. Given such a description of C, an alternative definition of Q arises:

Proposition 17. *The set*

$$\left\{ \alpha \in \mathbb{R}^m \quad : \quad \begin{array}{ll} \alpha^T s \geq 0 & \textit{for all } s \in \mathcal{S} \\ \alpha^T t = 0 & \textit{for all } t \in \mathcal{T} \end{array} \right\}. \tag{5.4}$$

is the polar of C.

Proof. Let Q be the polar of C and Q' be the set defined by (5.4). (a). $Q \subseteq Q'$: All the constraints $\alpha^T s \geq 0$ of Q' are constraints of Q. For every $t \in \mathcal{T}$, t and $-t$ belong to C, so all the constraints $\alpha^T t = 0$ of Q' are implied by constraints of Q. (b). $Q' \subseteq Q$: Since $\mathcal{S}$ and $\mathcal{T}$ generate C, any point $x \in C$ can be expressed as $x = \sum_{s\in\mathcal{S}} \mu_s s + \sum_{t\in\mathcal{T}} \nu_t t$ with $\mu \geq 0$. Taking the same coefficients and combining the constraints of Q', we obtain $\sum_{s\in\mathcal{S}} \mu_s \alpha^T s + \sum_{t\in\mathcal{T}} \nu_t \alpha^T t \geq 0$, hence $\alpha^T x \geq 0$. Therefore, every constraint of Q is implied by a combination of the constraints of Q'. □

Note that it follows immediately from the expression (5.4) that Q is a polyhedral cone.

Proposition 18. *A valid inequality $x^T \alpha \geq 0$ for Q is facet-defining if and only if there exists $s \in \mathcal{S}$ such that $x \in \text{cone}(s) + \text{lin}(\mathcal{T})$.*

Proof. ($\Rightarrow$, "only if"): We prove the contrapositive, i.e. that if there does not exist $s \in \mathcal{S}$ such that $x \in \text{cone}(s) + \text{lin}(\mathcal{T})$, then $x^T\alpha \geq 0$ is not facet-defining for Q. Let $y \in \text{cone}(\mathcal{S})$ and $l \in \text{lin}(\mathcal{T})$ be such that $x = y + l$. Since $l^T\alpha = 0$ for all $\alpha \in Q$, the inequalities $x^T\alpha \geq 0$ and $y^T\alpha \geq 0$ define the same face of P. If $y \notin \text{cone}(s)$ for any $s \in \mathcal{S}$, then y can be expressed as a conic combination of rays in $\mathcal{S}$. In other words, $y = \sum_{s_i \in \mathcal{S}} \lambda_i s_i$ with $\lambda_i \geq 0$ for all i, and $y \neq \nu s_i$ for all $s_i \in \mathcal{S}$ and $\nu \in \mathbb{R}_+$. The inequality $y^T\alpha \geq 0$ is thus a combination of valid inequalities for Q that are distinct from it, so it is not facet-defining. ($\Leftarrow$, "if"): We use the description (5.4) of Q and show that all the inequality constraints are necessary and thus facet-defining. Let $z \in \mathcal{S}$ and define C^z as $C^z := \text{cone}(\mathcal{S} \setminus \{z\}) + \text{lin}(\mathcal{T})$. Because we assume that $\mathcal{S}$ is a minimal generating set, $C^z \subsetneq C$. C^z is a polyhedral cone so we can define its polar Q^z and by Proposition 17,

$$Q^z \left\{ \alpha \in \mathbb{R}^m \quad : \quad \begin{array}{ll} \alpha^T s \geq 0 & \text{for all } s \in \mathcal{S} \setminus \{z\} \\ \alpha^T t = 0 & \text{for all } t \in \mathcal{T} \end{array} \right\}.$$

By the separating hyperplane theorem and Lemma 13, there must exist a valid inequality $\beta^T x \geq 0$ for C^z that separates z, i.e. such that $\beta^T z < 0$. As $\beta^T x \geq 0$ is not valid for C, it follows that $\beta \in Q^z \setminus Q$, showing that $Q^z \neq Q$. Therefore the inequality $\alpha^T z \geq 0$ is necessary to the description of Q in (5.4). □

Proposition 19. *If Q is the polar of a polyhedral cone C, then C is the polar of Q.*

Proof. Recall that $Q = \{\alpha \in \mathbb{R}^m : \alpha^T x \geq 0, \text{ for all } x \in C\}$. As Q is a polyhedral cone, we can define its polar using Definition 30. Let $G = \{y \in \mathbb{R}^m : y^T\alpha \geq 0, \text{ for all } \alpha \in Q\}$ be the polar of Q. If $y \in G$ then $\alpha^T y \geq 0$ for all $\alpha \in Q$, so $y \in C$. If $x \in C$ then $\alpha^T x \geq 0$ for all $\alpha \in Q$, so $x \in G$. Thus $G = C$. □

Corollary 5. *Let $Q = \text{cone}(B) + \text{lin}(\Gamma)$, where $\text{cone}(B \setminus \{b\}) + \text{lin}(\Gamma) \neq Q$ for any $b \in B$. A valid inequality $\alpha^T x \geq 0$ for C is facet-defining if and only if $\alpha \in \text{cone}(\beta) + \text{lin}(\Gamma)$ for some $\beta \in B$.*

Proposition 20 and Corollary 6 highlight the duality relationship between the dimension of C (resp. Q) and the dimension of the lineality space of Q (resp. the lineality space of C).

Proposition 20. *Let C be a polyhedral cone and Q its polar, the dimension of the lineality space of C is $m - \dim(Q)$.*

Proof. Let k be the dimension of the lineality space of C. There exist k and at most k linearly independent vectors v such that $v \in C$ and $-v \in C$. For each such vector v, for all $\alpha \in Q$ and $x \in C$, $\alpha^T x + \alpha^T v \geq 0$ and $\alpha^T x - \alpha^T v \geq 0$. Hence we have k and at most k linearly independent equalities $\alpha^T v = 0$ that are valid for Q, so $\dim(Q) = m - k$. □

Corollary 6. *Let C be a polyhedral cone and Q its polar, the dimension of the lineality space of Q is $m - \dim(C)$.*

A direct consequence of Proposition 20 and Corollary 6 is that Q (resp. C) is pointed if and only if C (resp. Q) is full-dimensional.

Coming back to the case of the general polyhedron P, Proposition 14 shows how to construct a description of the polyhedral cone P^+ from P and Proposition 17 expresses the polar Q of P^+ from that description.

$$Q = \left\{ (\alpha, \alpha_0) \in \mathbb{R}^{n+1} \quad : \quad \begin{array}{ll} \alpha^T x \geq \alpha_0 & \text{for all } x \in \mathcal{V} \\ \alpha^T r \geq 0 & \text{for all } r \in \mathcal{R} \\ \alpha^T l = 0 & \text{for all } l \in \mathcal{L} \end{array} \right\}, \tag{5.5}$$

where the finite sets $\mathcal{V}$, $\mathcal{R}$, and $\mathcal{L}$ generate $P = \text{conv}(\mathcal{V}) + \text{cone}(\mathcal{R}) + \text{lin}(\mathcal{L})$. Note that if P is pointed, one can construct minimum sets $\mathcal{V}$, $\mathcal{R}$, and $\mathcal{L}$ as follows: $\mathcal{V}$ is the set of the vertices of P, $\mathcal{R}$ is the set of the extreme rays of P, and $\mathcal{L}$ is empty.

Using Corollary 5 and Proposition 15, we know that if P is full-dimensional, the extreme rays of Q correspond to the facet-defining inequalities for P.

5.3 Separation and normalization

Let x^* be a point that we want to separate. We are looking for a facet-defining inequality (α, α_0) of P such that $\alpha^T x^* < \alpha_0$. Note that, by the separating hyperplane theorem, such a valid inequality exists if and only if $x^* \notin P$. One can find (α, α_0) by maximizing, over the set of the valid inequalities, the violation at x^*. This corresponds to solving

$$\begin{array}{rl} \min & \boldsymbol{\alpha}^T x^* - \boldsymbol{\alpha}_0 \\ \text{s.t.} & (\boldsymbol{\alpha}, \boldsymbol{\alpha}_0) \in Q. \end{array} \tag{5.6}$$

Throughout this chapter, we take the convention that variables are emphasized in bold within optimization problems, so as to distinguish them from constant data. Since Q is a polyhedron, (5.6) is a linear programming problem. But since it is a cone, $(0, 0)$ will be an optimal solution when $x^* \in P$, and the problem will always be unbounded when a violating inequality exists. As formulated here, we thus have no means of discriminating between solutions of (5.6), e.g. looking for a most violated inequality with finitely bounded coefficients. To do so, we first need to impose some normalization on the cut coefficients.

A common form of normalization is to fix the right-hand side α_0 of the cuts. We did exactly that in Chapter 2, fixing $\alpha_0 = 1$, and showing that the resulting polar set still contained all facet-defining inequalities for radial polyhedra. Since we are now in the general case, we would have to consider a disjunctive constraint of the type $\alpha_0 = 1 \vee \alpha_0 = 0 \vee \alpha_0 = -1$. Indeed, any valid inequality (α, α_0) can be made to fall into one of these three cases by dividing it by $|\alpha_0|$ if $\alpha_0 \neq 0$, but all three cases are necessary in order to cover all the possibilities. Unfortunately, incorporating such disjunctive constraint would make optimizing over Q much more complex, because it could no longer be modeled as a linear programming problem.

An alternative is to bound the magnitude of the cut coefficients with a constraint of the type

$$\sum_{i=1}^{n} |\alpha_i| \leq 1. \tag{5.7}$$

The issue with the normalization (5.7) is that it amounts to intersecting Q with several half-spaces, as is made obvious by the equivalent linear formulation

$$\begin{cases} -\gamma_i \leq \alpha_i \leq \gamma_i, & \text{for all } i \\ \sum_{i=1}^{n} \gamma_i \leq 1. \end{cases}$$

When Q is pointed, intersecting it with several half-spaces may yield vertices that do not correspond to an extreme ray of Q. Therefore, if we optimize over that truncated cone, we could obtain optimal solutions that do not give facet-defining inequalities.

To overcome these limitations, we adopt a normalization proposed by Balas and Perregaard in [14]

$$\alpha^T(\tilde{x} - x^*) = 1 \tag{5.8}$$

where $\tilde{x}$ is an arbitrary point that belongs to the convex hull of P. That normalization is linear, and consists in intersecting Q with a single hyperplane. As we now demonstrate, it makes the optimization problem bounded, while not cutting away any relevant solutions.

Proposition 21. *The optimization problem*

$$\begin{aligned} s = \quad \min \quad & \boldsymbol{\alpha}^T x^* - \boldsymbol{\alpha}_0 \\ \text{s.t.} \quad & (\boldsymbol{\alpha}, \boldsymbol{\alpha}_0) \in Q \\ & \boldsymbol{\alpha}^T(\tilde{x} - x^*) = 1 \end{aligned} \tag{5.9}$$

is always bounded. In particular, if $(\bar{\alpha}, \bar{\alpha}_0)$ is an optimal solution and $\bar{s}$ its objective function value, then $\bar{s} \geq -1$, which means that the violation of $(\bar{\alpha}, \bar{\alpha}_0)$ in x^* is at most one.

Proof. As $\tilde{x} \in P$, $\alpha^T\tilde{x} \geq \alpha_0$ is a valid constraint for Q. The normalization constraint (5.8) gives $\alpha^T\tilde{x} = 1 - \alpha^T x^*$, which lets us replace $\alpha^T\tilde{x}$ in the previous inequality, and obtain $\alpha^T x^* - \alpha_0 \geq -1$. □

Remark that $\bar{s} = -1$ when the optimal solution corresponds to an inequality that is tight at $\tilde{x}$, because $\alpha^T x^* = \alpha_0 - 1$ if and only if $\tilde{\alpha}^T x^* = \alpha_0$.

We now show that the addition of the normalization constraint does not remove any interesting valid inequality from the feasible region of the optimization problem (5.9).

Proposition 22. *Given any valid inequality $(\alpha, \alpha_0) \in Q$ separating $x^* \notin P$, there exists $\lambda > 0$ such that $(\lambda\alpha, \lambda\alpha_0) \in Q$ separates x^* and satisfies $(\lambda\alpha)^T(\tilde{x} - x^*) = 1$, for any $\tilde{x} \in P$.*

Proof. By hypothesis, $\alpha^T\tilde{x} \geq \alpha_0$ and $\alpha^T x^* < \alpha_0$. Thus $\alpha^T\tilde{x} > \alpha^T x^*$, and $\alpha^T(\tilde{x} - x^*) > 0$. The claim follows, by letting $\lambda = 1/(\alpha^T(\tilde{x} - x^*))$. □

Corollary 7. *The optimization problem (5.9) may be infeasible only if $x^* \in P$.*

One motivation for our choice of normalization is that, when P is full-dimensional (i.e. when Q is pointed), we can obtain facet-defining inequalities for P (i.e. extreme rays of Q) by using the simplex method. We now show that this result holds even if P is not full-dimensional.

Lemma 14. *Let C be a polyhedral cone in $\mathbb{R}^m$, $e_k \notin \text{aff}(C)$, and $C_{\{k\}} := C + \text{lin}(e_k)$. If the inequality $\alpha^T x \geq 0$ is facet-defining for $C_{\{k\}}$, then it is facet-defining for C.*

Proof. First note that $\alpha_k = 0$. Indeed, since $\alpha^T x \geq 0$ defines a proper face of $C_{\{k\}}$, there exist $w \in C_{\{k\}}$ such that $\alpha^T w = 0$. Therefore, $\alpha^T(w + \mu e_k) \geq 0$, so $\alpha^T \mu e_k \geq 0$, for all $\mu \in \mathbb{R}$.

Now let $d = \dim(C_{\{k\}})$. Because $e_k \notin \text{aff}(C)$, $d = \dim(C) + 1$. Since $C \subseteq C_{\{k\}}$, $\alpha^T x \geq 0$ is valid for C. Since $\alpha^T x \geq 0$ is facet-defining for $C_{\{k\}}$, there exist d affinely independent points $y^0, \ldots y^{d-1}$ in $C_{\{k\}} \cap \{x \in \mathbb{R}^m : \alpha^T x = 0\}$. We assume without loss of generality that $y^0 = 0$. The $d-1$ points $y^1, \ldots y^{d-1}$ are linearly independent, so the matrix

$$Y := \left[y^1 | \cdots | y^{d-1}\right]$$

is full column rank (i.e. $\text{rank}(Y) = d-1$). For each point y^i, one can construct $z^i = y^i + \mu_i e_k$ with $\mu_i \in \mathbb{R}$ such that $z^i \in C$. The matrix

$$Z := \left[z^1 | \cdots | z^{d-1}\right]$$

differs from Y only in the kth row, so its rank is at least $d-2$. As $\alpha_k = 0$, $\alpha^T z^i = \alpha^T y^i = 0$ for all i. Hence we know $d-1$ affinely independent z^i points in $C \cap \{x \in \mathbb{R}^m : \alpha^T x = 0\}$, which is thus of dimension at least $d - 2 = \dim(C) - 1$.

We finally show that it is of dimension exactly $\dim(C) - 1$. Indeed, otherwise $\alpha^T x = 0$ for all $x \in C$, implying that $\alpha^T x = 0$ for all $x \in C_{\{k\}}$, which contradicts the hypothesis that $\alpha^T x \geq 0$ is facet-defining for $C_{\{k\}}$. $\square$

Lemma 15. *Let $C_K := C + \text{lin}(\{e_k : k \in K\})$ be a full-dimensional polyhedral cone in $\mathbb{R}^m$ such that $\dim(C + \text{lin}(\{e_k : k \in K,\ k \neq \bar{k}\})) < n$ for any $\bar{k} \in K$. If $\alpha^T x \geq 0$ is facet-defining for C_K, then it is facet-defining for C.*

Proof. Because $\dim(C_{K \setminus \{\bar{k}\}}) < \dim(C_K)$, we know that $e_{\bar{k}} \notin \text{aff}(C_{K \setminus \{\bar{k}\}})$, for all $\bar{k} \in K$. Furthermore, $e_{\bar{k}} \notin \text{aff}(C_{K \setminus \bar{K}})$, for all $\bar{k} \in \bar{K} \subseteq K$. Let $\kappa := |K|$, $K = \{k^1, \ldots, k^\kappa\}$, and $K^p := \{k^1, \ldots, k^p\}$ where $0 \leq p \leq \kappa$. The proof works by induction on p. By hypothesis, $\alpha^T x \geq 0$ is facet-defining for C_{K^κ}. By Lemma 14, if it is facet-defining for C_{K^p}, then it is also facet-defining for $C_{K^{p-1}}$. Finally we obtain that $\alpha^T x \geq 0$ is facet-defining for $C_{K^0} = C$. $\square$

Lemma 16. *Let Q be a polyhedral cone in $\mathbb{R}^m$ and H the hyperplane $\{\alpha \in \mathbb{R}^m : \pi^T \alpha = 1\}$. If $Q \cap H$ is pointed and α^* is a vertex of $Q \cap H$, then*

(a). $\dim(\text{lin.space}(Q)) \in \{0, 1\}$,

(b). if $\dim(\text{lin.space}(Q)) = 1$, *then α^* is on the lineality direction of Q,*

(c). if $\dim(\text{lin.space}(Q)) = 0$, *then α^* is an extreme ray of Q.*

Proof. (a). We contradict $\dim(\text{lin.space}(Q)) \geq 2$. Let l^1 and l^2 be two distinct vectors of a basis of the linear space of Q. Both $\pi^T l^1 \neq 0$ and $\pi^T l^2 \neq 0$, otherwise $Q \cap H$ would not be pointed. Let $\mu^1, \mu^2 \in \mathbb{R}$ be such that $\pi^T \mu^1 l^1 = 1$ and $\pi^T \mu^2 l^2 = 1$. Because l^1 and l^2 are distinct vectors of a basis, $\mu^1 l^1 \neq \mu^2 l^2$. We can construct $l' = \mu^1 l^1 - \mu^2 l^2$ in the lineality space of Q with $\pi^T l' = 0$, which contradicts the fact that $Q \cap H$ is pointed. (b). Let l be the lineality direction of Q. Again, $\pi^T l \neq 0$, otherwise $Q \cap H$ would not be pointed. Let $\mu \in \mathbb{R}$ be such that $\pi^T \mu l = 1$. As α^* is a ray of Q, we can construct $\alpha^1 := \frac{3}{2}\alpha^* - \frac{1}{2}\mu l$ and $\alpha^2 := \frac{1}{2}\alpha^* + \frac{1}{2}\mu l$. One can easily verify that $\alpha^1, \alpha^2 \in Q \cap H$, and that $\alpha^* = \frac{1}{2}\alpha^1 + \frac{1}{2}\alpha^2$, contradicting the fact that α^* is a vertex of $Q \cap H$. (c). Standard. □

Theorem 5. *Let P be a polyhedron in $\mathbb{R}^n$. If $x^* \in \text{aff}(P) \setminus \text{conv}(P)$ and (α, α_0) describes a valid inequality separating x^* obtained by optimizing over the linear problem* (5.9) *with the simplex method, then $\alpha^T x \geq \alpha_0$ is facet-defining for P.*

Proof. To simplify the proof, we consider P^+ again, and rewrite (5.9) as

$$\begin{aligned} \min \quad & \boldsymbol{\alpha}^T r^* \\ \text{s.t.} \quad & \boldsymbol{\alpha} \in Q \\ & \boldsymbol{\alpha}^T(\tilde{r} - r^*) = 1 \end{aligned} \tag{5.10}$$

where $r^* = (x^*, 1)$ and $\tilde{r} = (\tilde{x}, 1)$. By Proposition 21, we know that the linear programming problem (5.10) has a finite optimal objective function value. If the feasible region is not pointed, then in the solution α^* returned by the simplex method, some nonbasic free variables must be fixed at an arbitrary value (typically zero), with a zero reduced cost, and could not enter the basis at a finite value. Let us denote by K the index set of such variables, i.e. α_k^* is nonbasic and fixed to zero for all $k \in K$. We claim that α^* is a vertex of the pointed polyhedron

$$Q_{*K} := \left\{\alpha \in Q : \alpha^T(\tilde{r} - r^*) = 1 \text{ and } \alpha_k = 0, \text{ for all } k \in K\right\}$$

obtained by intersecting the feasible region with $\{\alpha_k = 0, \text{ for all } k \in K\}$. Indeed, it corresponds to a basic feasible solution for that polyhedron (in the classical sense of a basic feasible solution, i.e. where all free variables are basic). Note that since none of the nonbasic free variables could enter the basis at a finite value, $Q_{*K\setminus\{k\}}$ is not pointed, for any $k \in K$. Removing the normalization constraint, we obtain

$$Q_K := \{\alpha \in Q : \alpha_k = 0, \text{ for all } k \in K\}$$

which, by Lemma 16 has a lineality space of dimension zero or one. Lemma 16 also specifies that the point α^* is either an extreme ray of Q_K or a lineality direction of Q_K. The latter would imply that $\alpha^{*T} x = 0$ is a valid equality for P^+ that separates r^*, contradicting the hypothesis that $r^* \in \text{aff}(P^+)$. Hence, Q_K is pointed and α^* is an extreme ray of Q_K.

Now observe that Q_K is the polar of

$$P_K^+ = P^+ + \text{lin}(\{e_k : k \in K\})$$

which is full-dimensional since Q_K is pointed. As α^* is an extreme ray of Q_K, $\alpha^{*T} x \geq 0$ is facet-defining for P_K^+. By Lemma 15, it is also facet-defining for P^+. □

Note that we consider that to each vertex of the feasible region of a linear program, corresponds at least one basis. This is not true with textbook descriptions of the simplex method when there are redundant constraints. It is true however with most practical implementations, where the standard form typically includes one logical variable for each tableau row (that variable is fixed to zero in phase II).

Remark also that a specific implementation of the simplex method may not ensure that, as required by the above proof, none of the nonbasic free variables could enter the basis at a finite value. However, enforcing this condition can easily be implemented as a postprocessing, whenever finding facet-defining inequalities is desirable.

5.4 Separation and row-generation

We have assumed so far that we have an inner description of $P = \text{conv}(P_J)$. However, this assumption does not hold in practice since, in the context of mixed-integer programming, we are typically provided with a description of P_J as $P_J := P_{LP} \cap \{x_J \in \mathbb{Z}\}$, where $P_{LP} := \{x \in \mathbb{R}^n_+ : Ax = b\}$ is a *linear relaxation* of P_J. To overcome this issue, as in Chapter 3, we adopt a classic row-generation approach, where we optimize over a relaxation $Q(S,T)$ of Q and iteratively strengthen this relaxation by adding to it constraints of Q that are violated by its incumbent optimal solution.

We start by considering the following formulation of Q,

$$Q = \left\{ (\alpha, \alpha_0) \in \mathbb{R}^{n+1} \;\middle|\; \begin{array}{ll} \alpha^T x^i \geq \alpha_0 & \forall x^i \in P_J \\ \alpha^T r^j \geq 0 & \forall r^j \in \text{recc}(\text{conv}(P_J)) \end{array} \right\}, \tag{5.11}$$

which is easily seen to be equivalent to (5.5). The partial description $Q(S,T)$ of Q is given by

$$Q(S,T) := \left\{ (\alpha, \alpha_0) \in \mathbb{R}^{n+1} \;\middle|\; \begin{array}{ll} \alpha^T x^i \geq \alpha_0 & \forall x^i \in S \\ \alpha^T r^j \geq 0 & \forall r^j \in T \end{array} \right\}, \tag{5.12}$$

where $S \subseteq P_J$, and T is a subset of the recession cone of $\text{conv}(P_J)$. Both S and T are finite sets, and $Q(S,T)$ is a relaxation of Q. We can now find candidate (i.e. possibly invalid) inequalities separating x^* by solving for α, α_0 the optimization problem

$$\begin{array}{lll} \min & \boldsymbol{\alpha}^T x^* - \boldsymbol{\alpha}_0 & \\ \text{s.t.} & \boldsymbol{\alpha}^T(\tilde{x} - x^*) = 1 & \\ & \boldsymbol{\alpha}^T \tilde{x} \geq \boldsymbol{\alpha}_0 & \\ & \boldsymbol{\alpha}^T x^i \geq \boldsymbol{\alpha}_0 & \forall x^i \in S \\ & \boldsymbol{\alpha}^T r^j \geq 0 & \forall r^j \in T \end{array} \tag{5.13}$$

which we call *master* problem. The elements of S and T are added iteratively as provided by the solution of a *slave* problem (5.14). Indeed, given a candidate inequality $(\bar{\alpha}, \bar{\alpha}_0)$, solving

$$\begin{array}{ll} \min & \bar{\alpha}^T \boldsymbol{y} \\ \text{s.t.} & \boldsymbol{y} \in P_J \end{array} \tag{5.14}$$

for y can have three possible outcomes. If the slave has an optimal solution y^* such that $\bar{\alpha}^T y^* \geq \alpha_0$, then we know that $(\bar{\alpha}, \bar{\alpha}_0)$ is a valid inequality for P_J. If the slave has an optimal solution y^* such that

$\bar{\alpha}^T y^* < \alpha_0$, then $(\bar{\alpha}, \bar{\alpha}_0)$ is not a valid inequality for P_J. But as $y^* \in P_J$, we can add the constraint $\alpha^T y^* \geq \alpha$ to $Q(S,T)$, which is currently violated by $(\bar{\alpha}, \bar{\alpha}_0)$. Finally, if the slave is unbounded, there must exist a mixed-integer infinite direction r^* of P_J such that $\bar{\alpha}^T r^* < 0$. We thus add to $Q(S,T)$ the constraint $\alpha^T r^* \geq 0$, which is valid for Q and cuts off the line $\{(\bar{\alpha}, \lambda) : \lambda \in \mathbb{R}\}$ including the point $(\bar{\alpha}, \bar{\alpha}_0)$.

For technical reasons, MIP solvers are not well-suited for finding mixed-integer infinite directions in problems. However, such direction can be found easily by first optimizing over the linear programming relaxation of P_J, looking for rays if it is not bounded. The simplex method permits this, and Corollary 8 shows that the method yields rays of $\text{conv}(P_J)$.

Lemma 17. *Let P_{LP} be a linear relaxation of P_J and r be a rational direction, the following statements are equivalent.*

1. There exists $\lambda > 0$ such that λr is a mixed-integer infinite direction of P_J.

2. r is a ray of $\text{conv}(P_J)$*.*

3. r is a ray of P_{LP}.

Proof. $(1 \Rightarrow 3)$ This follows from $P_J \subseteq P_{LP}$. $(3 \Rightarrow 1)$ Since r is rational, there exists $\lambda > 0$ such that $\lambda r_j \in \mathbb{Z}$ for all $j \in J$. Thus for every $x \in P_J$ and $y = x + \lambda r$, $y \in P_{LP}$ and $y_j \in Z$ for all $j \in J$. Therefore $y \in P_J$. $(1 \Rightarrow 2)$ This follows from $P_J \subseteq \text{conv}(P_J)$. $(2 \Rightarrow 1)$ This follows from $\text{conv}(P_J) \subseteq P_{LP}$. $\square$

Corollary 8. *The sets* $\text{conv}(P_J)$ *and P_{LP} have the same recession cone and share a same set of extreme rays.*

Proof. Since we assume in this text that we consider only rational data, Lemma 17 shows that r is a ray of $\text{conv}(P_J)$ iff it is a ray of P_{LP}, proving the claim. $\square$

Note on the other hand that in (5.12), the explicit inclusion of the constraint $\alpha^T \tilde{x} \geq \alpha_0$, where $\tilde{x} \in P_J$ is the point used in the normalization, ensures that the master is always bounded. Indeed, the proof of Proposition 21 equally applies to the problem

$$\begin{aligned} \bar{s} = \quad \min \quad & \boldsymbol{\alpha}^T x^* - \boldsymbol{\alpha}_0 \\ \text{s.t.} \quad & \boldsymbol{\alpha}^T (\tilde{x} - x^*) = 1 \\ & \boldsymbol{\alpha}^T \tilde{x} \geq \boldsymbol{\alpha}_0. \end{aligned} \tag{5.15}$$

The same row-generation method is used by Perregaard and Balas in [69], except that they limit to linear programming slaves, in order to obtain fast separation. As the objective here is to have an exact separator, we can not make this assumption.

An overview of the method is presented in Algorithm 4.

```
Step 0  Input: a set $P_J$ and a point $x^*$ to separate.
        Initialize $S := \emptyset$ and $T := \emptyset$.
Step 1  Solve the master problem (5.13).
        Let $(\bar{\alpha}, \bar{\alpha}_0)$ be an optimal solution (if such solution exists).
        If the master problem is infeasible or if $\bar{\alpha}^T x^* \geq \bar{\alpha}_0$:
            There is no valid inequality for $P_J$ separating $x^*$. Stop
Step 2  Solve the slave problem (5.14).
        If the slave problem is infeasible:
            Any inequality is valid for $P_J$. Consider $0 \geq 1$ and stop.
        If the slave problem is unbouned:
            Let $r$ be an extreme ray of its LP relaxation $P_{LP}$ that
            shows unboundedness (i.e. $\bar{\alpha}^T r < 0$).
            $T := T \cup \{r\}$. Go to Step 1.
        If the slave problem has an optimal solution $y^*$:
            If $\bar{\alpha}^T y^* < 0$:
                $S := S \cup \{y^*\}$. Go to Step 1.
            If $\bar{\alpha}^T y^* \geq 0$:
                $\bar{\alpha}^T x \geq \bar{\alpha}_0$ is a valid inequality for $P_J$
                separating $x^*$. Stop.
```

Algorithm 4: Row-generation algorithm for optimizing over Q

5.5 Faster separation through lifting

The algorithm described in the previous section computes a point of P_J or a ray of P_{LP} at each iteration. That involves solving a linear programming problem, then a mixed-integer problem whenever the former is bounded. It is thus naturally slow in practice, but can be implemented. In order to illustrate the time required to run the method, we leave aside a few implementation issues that will be covered later, and present the results obtained in a benchmark with this algorithm. We test it on the problem instances from MIPLIB 3.0 [23], with a heuristic selection of five-row models (see Section 5.8). The results are shown in Table 5.1. Without delving into the details, we very quickly describe the data it presents. Before the start of our computations, the instances are preprocessed and strengthened with a round of GMIs. The columns *GMIs* and *GMIs %gc* show the number of GMIs generated and the percentage of gap closed as a result. Our algorithm then separates several rounds of five-row cuts. The total number of such cuts is indicated in the column *cuts*, and *tight* indicates the number of them that are tight at the end. Finally, the column *%gc* indicates the percentage of gap closed by all the cuts (five-row and GMI), and a "Y" in the column *ex.* indicates that the separation is *exact* (i.e. we can prove that no more gap could be closed with cuts from the five-row models). The column *time* shows the total wall time taken by the experiment in seconds, and *cg_iter* (standing for cut-generation iterations) indicates the total number of iterations Algorithm 4 performs, i.e. the total number of five-row MIPs solved in the process.

We observe in Table 5.1 that the test completes before hitting the time limit of four hours for only 13 of the 66 instances. Furthermore, for only 5 among them is the separation exact, and we even detect numerical difficulties in one instance (marked by a ! in the *ex.* column). This means that, for at least one of the cuts, the optimal solution returned by the LP solver does not satisfy all the constraints it should. The experiment takes over an hour on average, with a geometric mean of almost 600 seconds.

	Base algorithm							
name	time	cg_iter	GMIs	GMIs %gc	cuts	tight	%gc	ex.
bell3a	1391.20	45452	14	39.03	32	18	49.45	_
bell5	1283.30	172368	17	14.53	64	17	21.40	!
egout	2.42	1401	8	31.94	44	30	100.00	Y
flugpl	12.67	705	7	10.75	14	8	100.00	Y
lseu	12834.91	241012	5	20.48	41	33	76.64	_
misc03	2919.36	120048	4	8.62	0	7	8.62	_
mod008	7385.68	7503	5	21.62	0	1	21.62	_
p0033	12.46	1575	4	34.42	45	40	100.00	Y
p0201	14400.02	896899	14	0.39	41	26	17.83	_
pp08a	48.03	35015	53	51.44	157	104	83.47	Y
set1ch	7668.69	227506	121	28.20	261	207	90.46	_
vpm1	198.51	67630	15	6.18	151	51	41.59	Y
vpm2	6404.89	517514	21	7.31	215	114	41.02	_
average	4197.088	179586.769	22.154	21.147	81.923	50.462	57.854	**46%**
geometric	**598.813**	**39956.226**	12.321	13.793	0.000	25.768	45.390	-

Table 5.1: Running time for Algorithm 4

Although we do not aim at having a separator that is fast enough to be useful for speeding up the resolution of MIPs, the huge computing times involved on such small instances would make the cost of the computation prohibitive on any reasonable set of medium-sized benchmark instances. It is therefore necessary for us to come up with ways to make our separator faster.

A first possible way is to take advantage of the fact that the point x^* typically has a lot of components that are at one of their bounds in the formulation of P_{LP}. In such a case, it is possible to first focus on the face $\tilde{F} = P \cap \{x_j = x^*_j,$ for all j such that x^*_j is at a bound$\}$ of P, and find a valid inequality for $\tilde{F}$ separating x^*. One can then *lift* that inequality in order for it to be valid for the whole set P. A similar idea was exploited by Perregaard and Balas in [69].

Theorem 6 shows, in the case of one component at a bound, that the lifting is always feasible and yields a valid inequality for P that also separates x^*. The conclusion applies to any number of bounds by using Theorem 6 repeatedly, forming a feasible sequential lifting. Note that the bound constraint is presented in general form $f^T x \geq f_0$ to handle lower or upper bounds equivalently.

Theorem 6. *Given $P = \text{conv}(\{x^i\}) + \text{cone}(\{r^j\}) \neq \emptyset$, consider an inequality $f^T x \geq f_0$ that is valid for P and let $F = P \cap \{x : f^T x = f_0\}$. If $F \neq \emptyset$ and $\alpha^T x \geq \alpha_0$ is a valid inequality for F, then there always exists a finite coefficient μ such that $\alpha x + \mu(f^T x - f_0) \geq \alpha_0$ is valid for P.*

Proof. Since $P = \text{conv}(\{x^i\}) + \text{cone}(\{r^j\})$, an inequality $\gamma^T x \geq \gamma_0$ is valid for P if and only if $\gamma^T x^i \geq \gamma_0$ for i and $\gamma^T r^j \geq 0$ for all j. Let us consider the vertices x^i first. For each vertex x^i, we must satisfy the condition

$$\alpha^T x^i + \mu(f^T x^i - f_0) \geq \alpha_0.$$

We have two cases:

- $f^T x^i = f_0$. In this case, we have that $x^i \in F$ and the inequality $\alpha^T x^i + \mu(f^T x^i - f_0) \geq \alpha_0$ is satisfied for every value of μ (because $\alpha^T x \geq \alpha_0$ is valid for F and μ vanishes).
- $f^T x^i > f_0$. In this case, μ must satisfy the condition

$$\mu \geq \frac{\alpha_0 - \alpha^T x^i}{f^T x^i - f_0}.$$

Let us consider the rays r^j. For each ray r^j, we must satisfy the condition

$$\alpha^T r^j + \mu f^T r^j \geq 0.$$

Again, we have two cases:

- $f^T r^j = 0$. In this case, it is easy to check that r^j is a ray also for F, and hence $\alpha^T r^i \geq 0$, because $\alpha^T x \geq \alpha_0$ is valid for F. Since $\alpha^T r^i \geq 0$ and μ vanishes, the condition is satisfied for every value of μ.

- $f^T r^j > 0$. In this case, μ must satisfy the condition:

$$\mu \geq -\frac{\alpha^T r^j}{f^T r^j}$$

Since we have a finite number of conditions $\mu \geq z_k$ where z_k is finite, taking the maximum value for the right-hand side suffices to find a value of μ that yields a valid inequality for P. □

Corollary 9. *If $F \neq \emptyset$, $\alpha^T x \geq \alpha_0$ is facet-defining for F, and $\mu = \max\{z_k\}$, then $\alpha x + \mu(f^T x - f_0) \geq \alpha_0$ is facet-defining for P.*

Proof. If there are no lower bounds z_k on μ, then $F = P$ and the result is immediate. Otherwise, the largest lower bounds on μ corresponds to (at least) one vertex x^i of P such that $f^T x^i > f_0$, or one extreme ray r^j of P such that $f^T r^j > 0$. In the case of a vertex, x^i is tight on the lifted inequality, and it is affinely independent from the points in F. In the case of an extreme ray, let $\tilde{x} \in F$, the same holds $\tilde{x} + r^j$. In both cases, $\dim(P \cap \{x : \alpha x + \mu(f^T x - f_0) = \alpha_0\}) = \dim(F) + 1$, so the lifted inequality is facet-defining for P. □

In case F is empty, i.e. if it is not a proper face of P, then Theorem 7 provides a similar result, showing again how to lift to obtain a valid inequality for P.

Theorem 7. *If $F = \emptyset$, there always exists a finite coefficient μ such that $\mu(f^T x - f_0) \geq 1$ is valid for P, i.e. we can always lift the inequality $0^T x \geq 1$.*

Proof. The proof is similar to the one of the previous theorem. Let us consider the vertices first. Since $F = \emptyset$, then it cannot be that $f^T x^i = f_0$, otherwise x^i would be a feasible point in F. So we must have $f^T x^i > f_0$, and hence the condition:

$$\mu \geq \frac{1}{f^T x^i - f_0}$$

Let us consider the rays. We must prove that $\mu f^T r^j \geq 0$. Since $f^T r^j \geq 0$, we get the condition $\mu \geq 0$ (the same per all rays). Taking again the largest right-hand side of the constraints on μ proves the claim. □

In general, if $f^T x \geq f_0, \forall x \in P$, Theorem 6 shows that an inequality (α, α_0) valid only for the nonempty set $P \cap \{f^T x = f_0\}$ can always be lifted so as to be valid for P. Theorem 7 indicates that this is also true when $P \cap \{f^T x = f_0\}$ is empty provided that $\alpha = 0$ and $\alpha_0 = 1$. Note that any inequality is valid for the empty set, but we proved that for the special choice of the infeasible constraint $0 \geq 1$, the lifting problem is always feasible.

In all cases, if $f^T x^* = f_0$, then the slack of the lifted inequality at x^*,

$$\alpha^T x^* + \mu(f^T x^* - f_0) - \alpha_0$$

is equal to the slack of the initial inequality at x^*,

$$\alpha^T x^* - \alpha_0.$$

In other words, lifting a valid inequality that is violated at x^* from a face that is tight at x^* yields a new valid inequality that is violated by the same amount at x^*.

As mentioned earlier, one family of inequalities defining faces of P is readily available: the constraints of the linear programming relaxation of the mixed-integer set P. In standard form, these are simply the bounds on the variables. Furthermore, if the point x^* that we want to separate is a vertex of the LP relaxation, then we know that a number of its components (namely, the nonbasic variables) are tight at their respective bounds. And we just showed that lifting from tight constraints does not affect violation.

In practice, for a lower bound constraint $x_j \geq l_j$, the lifted inequality takes the form $\alpha^T x + \mu(x_j - l_j) \geq \alpha_0$ and for an upper bound constraint $x_j \leq u_j$, it becomes $\alpha^T x + \mu(u_j - x_j) \geq \alpha_0$. Since in both cases the bound is tight at x_j^*, we can replace l_j and u_j by x_j^*. Denoting by α_j the cut coefficient for x_j, we obtain the common formula

$$\alpha^T x + \alpha_j x_j \geq \alpha_0 + \alpha_j x_j^*.$$

We can now apply these concepts to design a two-phase process. Let

$$P := \{x \in \mathbb{R}^n : Ax = b,\ l \leq x \leq u,\ x_J \in \mathbb{Z}\}$$

and N be the index set of components x^* that are at a bound, i.e. either $x_j^* = l_j$ or $x_j^* = u_j$, for all $j \in N$. We denote by B the complement set, $B := \{1, \dots, n\} \setminus N$.

In a first phase, we find a valid inequality $(\bar{\alpha}_B, \bar{\alpha}_0)$ for $F := P \cap \{x_N = x_N^*\}$. If no such inequality exists, then $x_B^* \in F \subseteq P$, which means that there is no valid inequality for P that separates x^*. In the special case where F is empty, we consider $(\bar{\alpha}_B, \bar{\alpha}_0) = (0, 1)$, as Theorem 7 suggests.

In a second phase, we find a valid inequality $(\alpha_N, \alpha_B, \alpha_0)$ for P. In that process, we can fix $\alpha_B = \bar{\alpha}_B$ and $\alpha_0 = \bar{\alpha}_0 + \alpha_N^T x_N^*$. The above reasoning guarantees that a feasible solution exists for α_N and that the violation at x^* of the initial cut obtained is conserved through the second phase. Remark also that in the second phase, the normalization constraint (5.8) is always satisfied, hence redundant.

Our proposed approach is summarized in Algorithm 5, where the inequalities (α, α_0) in Phases 1 and 2 are found using Algorithm 4.

The results from Table 5.2 show that this method yields a decrease in computation time by a factor 6 in geometric mean over the 13 instances. More importantly, we now have exact separation for more than

```
Init      Let $N, B$ be such that $x_N = x^*_N$ and $x_j \neq x^*_j, \forall j \in B$
Phase 1   Let $P_1 := P \cap \{x_N = x^*_N\}$.
          If $P_1 = \emptyset$,
                $(\bar{\alpha}_B, \bar{\alpha}_0) := (0, 1)$. Go to Lifting.
          Find a valid inequality $(\bar{\alpha}_B, \bar{\alpha}_0)$ for $P_1$.
          If no such inequality exist,
                No valid inequality for $P$ separates $x^*$. Stop.
Phase 2   Find a valid inequality $(\alpha_N, \bar{\alpha}_B, \alpha_0)$ for $P$.
```

Algorithm 5: Two-phase process

				Two-phase				
name	time	cg_iter	GMIs	GMIs %gc	cuts	tight	%gc	ex.
bell3a	347.56	31883	14	39.03	39	22	55.76	–
bell5	158.76	38588	17	14.53	106	15	20.50	–
egout	1.50	863	8	31.94	77	41	100.00	Y
flugpl	5.21	272	7	10.75	22	11	100.00	Y
lseu	11741.74	156057	5	20.48	109	54	82.13	–
misc03	383.11	27610	4	8.62	0	7	8.62	–
mod008	242.83	61	5	21.62	0	1	21.62	–
p0033	4.18	672	4	34.42	45	28	100.00	Y
p0201	5151.84	199654	14	0.39	61	28	17.83	Y
pp08a	19.69	11375	53	51.44	190	119	83.47	Y
set1ch	179.56	26303	121	28.20	318	208	90.46	Y
vpm1	29.83	9286	15	6.18	171	50	41.59	Y
vpm2	193.88	45999	21	7.31	304	130	41.02	–
average	1419.976	42201.769	22.154	21.147	110.923	54.923	58.692	**54%**
geometric	**105.772**	**8548.323**	12.321	13.793	0.000	28.189	45.904	-

Table 5.2: Running time for Algorithm 5: two-phase approach

half of the instances. And we will see in the next section that the same reasoning can be carried on to obtain further improvements. Note the we leave aside for now the columns presenting the number of cuts and the amount of gap closed, as we focus only on the speed of cut generation. They are merely provided for completeness, and to check the consistency of our results.

5.6 Lifting binary variables

In the previous section, we only proved that under certain condition, it is possible to lift a valid inequality. We did not provide a way to compute the lifting coefficient (μ in Theorem 6 and Theorem 7) as we simply rely on Algorithm 4 to find it.

It may be interesting however to examine the lifting problem itself. For simplicity, we assume for the time being that we want to lift the cut coefficient of only one variable x_k, i.e. $N = \{k\}$. Let us consider a valid inequality $\bar{\alpha}_B \geq \bar{\alpha}_0$ for $F = P \cap \{x_k = x^*_k\}$. We want to find α_k such that

$$\alpha_k x_k + \bar{\alpha}_B x_B \geq \bar{\alpha}_0 + \alpha_k x^*_k$$

is satisfied for all (x_k, x_B) in P. We know that it is always true if $x_k = x^*_k$. For $x_k \neq x^*_k$, we need

$$\alpha_k(x^*_k - x_k) \leq \bar{\alpha}_B x_B - \bar{\alpha}_0$$

i.e.

$$\begin{cases} \alpha_k \le \dfrac{\bar{\alpha}_B x_B - \bar{\alpha}_0}{x_k^* - x_k} & \forall x \in P : x_k < x_k^* \\ \alpha_k \ge \dfrac{\bar{\alpha}_B x_B - \bar{\alpha}_0}{x_k^* - x_k} & \forall x \in P : x_k > x_k^*. \end{cases} \tag{5.16}$$

For a given k, only one of the two types of conditions in (5.16) can occur, since x_k^* is the value of a bound on x_k. Therefore, we can find α_k if we can solve the optimization problem

$$\alpha_k = \min \left\{ \frac{\bar{\alpha}_B \boldsymbol{x}_B - \bar{\alpha}_0}{x_k^* - \boldsymbol{x}_k} : \boldsymbol{x} \in P \text{ and } \boldsymbol{x}_k < x_k^* \right\} \tag{5.17}$$

if x_k^* is at upper bound or

$$\alpha_k = \max \left\{ \frac{\bar{\alpha}_B \boldsymbol{x}_B - \bar{\alpha}_0}{x_k^* - \boldsymbol{x}_k} : \boldsymbol{x} \in P \text{ and } \boldsymbol{x}_k > x_k^* \right\} \tag{5.18}$$

if x_k^* is at lower bound. Unfortunately, we can not tackle (5.17) or (5.18) in practice, be it only because the constraints $x_k < x_k^*$ and $x_k > x_k^*$ may make the feasible region an open set. However, if $k \in J$, i.e. if x_k is an integer-constrained variable, the problems become

$$\alpha_k = \min \left\{ \frac{\bar{\alpha}_B \boldsymbol{x}_B - \bar{\alpha}_0}{x_k^* - \boldsymbol{x}_k} : \boldsymbol{x} \in P \text{ and } \boldsymbol{x}_k \le x_k^* - 1 \right\} \tag{5.19}$$

and

$$\alpha_k = \max \left\{ \frac{\bar{\alpha}_B \boldsymbol{x}_B - \bar{\alpha}_0}{x_k^* - \boldsymbol{x}_k} : \boldsymbol{x} \in P \text{ and } \boldsymbol{x}_k \ge x_k^* + 1 \right\}, \tag{5.20}$$

respectively. While the objective functions of (5.19) and (5.20) are still nonlinear, their feasible regions are mixed-integer sets. And if we further specify that x_k is a binary variable, then the problems reduce to

$$\alpha_k = \min \left\{ \bar{\alpha}_B \boldsymbol{x}_B - \bar{\alpha}_0 : \boldsymbol{x} \in P \text{ and } \boldsymbol{x}_k = 0 \right\} \tag{5.21}$$

and

$$\alpha_k = -\min \left\{ \bar{\alpha}_B \boldsymbol{x}_B - \bar{\alpha}_0 : \boldsymbol{x} \in P \text{ and } \boldsymbol{x}_k = 1 \right\}, \tag{5.22}$$

which are mixed-integer programming problems.

Therefore, for a binary variable, it is possible to compute a lifted cut coefficient by solving a single MIP instead of resorting to the row-generation Algorithm 4 which may need to solve many MIPs of the same size. In general for $|N| \ge 1$, we can compute sequentially a valid lifted coefficient α_k for all the binary variables, by solving for each k the mixed-integer problem

$$\alpha_k = (1 - 2x_k^*) \min_{\boldsymbol{x} \in P} \left\{ \bar{\alpha}_B^T \boldsymbol{x}_B - \bar{\alpha}_0 \ : \ \boldsymbol{x}_{N \setminus \{k\}} = x^*_{N \setminus \{k\}},\ \boldsymbol{x}_k = 1 - x_k^* \right\} \tag{5.23}$$

then setting $N := N \setminus \{k\}$ and $B := B \cup \{k\}$. Note that such a lifting is not unique. In particular, it is sequence-dependent, i.e. it depends on the order according to which it is applied on variables.

On our benchmark set, the sequential lifting of binary variables lets us further decrease the computational times, at 74 seconds in geometric mean, down from 106 seconds (Table 5.3). Moreover, the separation is now exact for all but three instances.

				Lifting binaries				
name	time	cg_iter	GMIs	GMIs %gc	cuts	tight	%gc	ex.
bell3a	122.04	17572	14	39.03	68	25	56.77	-
bell5	131.01	27178	17	14.53	129	15	20.50	Y
egout	0.79	431	8	31.94	57	23	100.00	Y
flugpl	7.48	272	7	10.75	22	11	100.00	Y
lseu	2592.50	16016	5	20.48	157	54	82.13	Y
misc03	75.24	2277	4	8.62	59	8	8.62	Y
mod008	608.23	51	5	21.62	0	1	21.62	-
p0033	3.77	174	4	34.42	36	21	100.00	Y
p0201	1514.24	23076	14	0.39	119	41	17.83	Y
pp08a	14.32	11112	53	51.44	167	113	83.47	Y
set1ch	315.11	31078	121	28.20	292	198	90.46	Y
vpm1	21.33	9744	15	6.18	174	43	41.59	Y
vpm2	343.53	61836	21	7.31	325	102	41.01	-
average	442.276	15447.462	22.154	21.147	123.462	50.385	58.769	**77%**
geometric	**74.334**	**4080.668**	12.321	13.793	0.000	26.675	45.967	-

Table 5.3: Running time: lifting binary variables

5.7 Computational tricks

Recall that, given a candidate inequality $(\bar{\alpha}, \bar{\alpha}_0)$, we solve the slave MIP (5.14)

$$\begin{aligned} \min \quad & \bar{\alpha}^T \boldsymbol{y} \\ \text{s.t.} \quad & \boldsymbol{y} \in P \end{aligned}$$

to check whether $(\bar{\alpha}, \bar{\alpha}_0)$ belongs to Q, i.e. whether it is a valid inequality for P. Let y^* be the optimal solution, if $\bar{\alpha}^T y^* \geq \alpha_0$, then $(\bar{\alpha}, \bar{\alpha}_0)$ is valid. If $\bar{\alpha}^T y^* < \alpha_0$, we can add the constraint $\alpha^T y^* \geq \alpha$ to $Q(S,T)$.

In the first case, we do not need the exact value of y^*. Therefore, during the branch-and-bound process, if the global LP bound becomes larger then α_0, then we already know that $\bar{\alpha}^T y \geq \alpha_0$ for all $y \in P$, so $(\bar{\alpha}, \bar{\alpha}_0)$ is valid. Note that the global LP bound is the lowest of the LP bounds of the individual branch-and-bound nodes that are open at a given time of the branch-and-bound algorithm.

In the second case, we do not strictly need y^* to be optimal as long as it provides a constraint of Q that is violated at $(\bar{\alpha}, \bar{\alpha}_0)$. Therefore, if a feasible solution $\tilde{y} \in P$ becomes available during the branch-and-bound such that $\bar{\alpha}^T \tilde{y} < \alpha_0$, then it can already be added to the formulation of Q.

Based on these observation, we take advantage of the callback facilities provided by the solver (CPLEX), and implement a function to be called at each new node of the branch-and-bound. If this function detects that the global LP bound raises above α_0, the branch-and-bound is interrupted, and we mark $(\bar{\alpha}, \bar{\alpha}_0)$ as a valid inequality for P. If it detects that a suitable feasible solution $\tilde{y}$ is available, then it decreases the iteration limit that we put on MIP resolution. This way, the branch-and-bound search for a more-violated constraint of Q continues, but we avoid spending too much computing time on it, as it is not strictly necessary.

By implementing this on top of the techniques described earlier, we manage to further reduce the computation time. Table 5.4 shows that the geometric mean of the time spent is now 57 seconds, instead of 74 previously.

A second computational technique we explore is closely related to the lifting described in Section 5.6.

				Callbacks				
name	time	cg_iter	GMIs	GMIs %gc	cuts	tight	%gc	ex.
bell3a	93.97	15068	14	39.03	69	24	56.82	-
bell5	93.79	27272	17	14.53	129	15	20.50	Y
egout	0.76	436	8	31.94	57	23	100.00	Y
flugpl	6.93	272	7	10.75	22	11	100.00	Y
lseu	2135.36	13492	5	20.48	134	28	77.76	-
misc03	57.46	2277	4	8.62	56	8	8.62	Y
mod008	653.68	511	5	21.62	1	2	21.70	-
p0033	3.46	174	4	34.42	36	21	100.00	Y
p0201	296.92	16674	14	0.39	115	39	17.83	Y
pp08a	13.63	11112	53	51.44	167	113	83.47	Y
set1ch	219.20	31078	121	28.20	292	198	90.46	Y
vpm1	19.72	9744	15	6.18	174	43	41.59	Y
vpm2	291.17	61776	21	7.31	324	103	41.01	-
average	298.927	14606.615	22.154	21.147	121.231	48.308	58.443	**69%**
geometric	**56.905**	**4639.462**	12.321	13.793	69.913	26.584	45.790	-

Table 5.4: Running time: MIP solver callbacks

Upon closer look at the condition (5.16), it appears that while we can not solve the derived optimization problems (5.17) and (5.18) through mixed-integer linear programming, we can at least provide finite bounds on the coefficient α_k by solving a MIP. Specifically, (5.16) gives for any $\epsilon > 0$

$$\alpha_k \leq \min\left\{\frac{\bar{\alpha}_B \boldsymbol{x}_B - \bar{\alpha}_0}{x_k^* - \boldsymbol{x}_k} : \boldsymbol{x} \in P \text{ and } \boldsymbol{x}_k = x_k^* - \epsilon\right\} \tag{5.24}$$

and

$$\alpha_k \geq \max\left\{\frac{\bar{\alpha}_B \boldsymbol{x}_B - \bar{\alpha}_0}{x_k^* - \boldsymbol{x}_k} : \boldsymbol{x} \in P \text{ and } \boldsymbol{x}_k = x_k^* + \epsilon\right\}. \tag{5.25}$$

Note that solving (5.24) or (5.25) for all $\epsilon > 0$ would provide a necessary and sufficient condition for the validity of α_k. Solving them for a fixed value of ϵ instead only provides a necessary condition

$$\alpha_k \leq \frac{1}{\epsilon} \min\left\{\bar{\alpha}_B \boldsymbol{x}_B - \bar{\alpha}_0 : \boldsymbol{x} \in P \cap \{x_k = x_k^* - \epsilon\}\right\} \tag{5.26}$$

if x^* is at upper bound and

$$\alpha_k \geq -\frac{1}{\epsilon} \min\left\{\bar{\alpha}_B \boldsymbol{x}_B - \bar{\alpha}_0 : \boldsymbol{x} \in P \cap \{x_k = x_k^* + \epsilon\}\right\} \tag{5.27}$$

if x^* is at lower bound, that we can compute by solving a MIP. These bounds are valid constraints for the master problem, but there is no formal argument for claiming that adding them to our representation of Q will make separation faster. Indeed, the most part of our computing time is devoted to solving the slave MIP, whose difficulty is not directly affected by the state of the master. However, the intuition suggests that we may decrease the number of iterations in Algorithm 4 by refining our initial representation of Q.

In practice, we need to choose a value for $\epsilon > 0$ to solve (5.26) or (5.27) as mixed-integer linear problems. We would prefer to obtain a strong bound on α_k, but we do not have any way of predicting what value of ϵ will provide such good bound. We only know that very small and very large values of ϵ are likely to cause numerical trouble. We thus choose to fix $\epsilon = 1$ in our computations. Further motivation is provided in the special case where x_k is an integer-constrained variable. Then, the exact bound is provided by solving (5.26) or (5.27) for all $\epsilon \in \{1, 2, \ldots\}$, and $\epsilon = 1$ has a special status as the smallest possible value in the set.

Table 5.5 shows that the intuition is wrong and that computing bounds on α_k by solving MIPs does not decrease the number of iterations in Algorithm 4. On most instances, it seems to be counter produc-

				Bounds on α				
name	time	cg_iter	GMIs	GMIs %gc	cuts	tight	%gc	ex.
bell3a	94.96	15068	14	39.03	69	24	56.82	-
bell5	91.88	27272	17	14.53	129	15	20.50	Y
egout	0.56	436	8	31.94	57	23	100.00	Y
flugpl	5.46	272	7	10.75	22	11	100.00	Y
lseu	2005.63	74796	5	20.48	217	103	82.13	Y
misc03	57.82	2277	4	8.62	56	8	8.62	Y
mod008	658.90	511	5	21.62	1	2	21.70	-
p0033	9.69	174	4	34.42	36	21	100.00	Y
p0201	2259.56	32379	14	0.39	122	35	17.83	Y
pp08a	22.92	11112	53	51.44	167	113	83.47	Y
set1ch	340.94	31078	121	28.20	292	198	90.46	Y
vpm1	33.95	9744	15	6.18	174	43	41.59	Y
vpm2	469.82	61776	21	7.31	324	103	41.01	-
average	465.545	20530.385	22.154	21.147	128.154	53.769	58.779	**77%**
geometric	**80.073**	**5569.988**	12.321	13.793	72.884	29.142	45.983	-

Table 5.5: Running time: computing bounds on α coefficients

tive, as suggested by the increased computing times. Therefore, this technique is not used in the next computation.

A third computational technique also exploits an idea related to lifting. Our empirical experience has shown that while phase-1 and lifting are usually fast, phase-2 consumes the largest share of our computing time. To try to obtain a phase-2 implementation that more closely resembles the lifting, we can look for the α_k coefficients, still via row-generation, but one by one. Indeed, phase-2 solves the problem of lifting all the α_k for $k \in N$ simultaneously. We can instead find the values of each α_k sequentially. By doing so, we lose a degree of freedom: In a simultaneous lifting, we can find the "best" coefficients according to a given measure. In a sequential lifting, this is not possible, as the result mainly depends on the particular order according to which we treat the variables. However, we are not using any measure for the quality of the cuts in Algorithm 5. In particular, the objective function is zero in phase-2, as the violation of the inequality we look for is already fixed at the violation found in phase-1. As was the case when we computed bounds on α, we do not have any formal argument for claiming that the sequential lifting would speed the computation up. The intuition is that we will apply row-generation many times, but to solve simpler problems, and we can only verify it through experimentation.

In this new implementation, after phase-1, we try to fix as many α_k coefficient as possible via sequential lifting. We call this phase-S, and it is performed with a reduced iteration limit for Algorithm 4. For all the remaining coefficients that are not found in phase-S, we then fall back to phase-2.

The data in Table 5.6 shows that phase-S does not improve the speed of our separator. The computing times are larger, and the overall number of row-generation iterations is not consistently decreased. However, by combining the addition of bounds on α and phase-S (Table 5.7), we see that we obtain results that are competitive with those of Table 5.4, beating them by a large margin except for two instances.

Table 5.8 summarizes all the techniques we have tested, and the resulting computing times.

				Phase-S				
name	time	cg_iter	GMIs	GMIs %gc	cuts	tight	%gc	ex.
bell3a	78.86	10988	14	39.03	82	28	56.83	-
bell5	66.88	13544	17	14.53	115	15	20.50	Y
egout	11.73	3450	8	31.94	64	22	100.00	Y
flugpl	9.14	359	7	10.75	26	7	100.00	Y
lseu	3787.37	52255	5	20.48	172	81	82.13	Y
misc03	226.26	2311	4	8.62	67	10	8.62	Y
mod008	1088.27	2182	5	21.62	0	1	21.62	-
p0033	5.86	520	4	34.42	53	38	100.00	Y
p0201	291.55	8620	14	0.39	121	46	17.83	Y
pp08a	11.44	9350	53	51.44	164	110	83.47	Y
set1ch	25.01	12157	121	28.20	291	191	90.46	Y
vpm1	14.71	5361	15	6.18	188	28	41.59	Y
vpm2	390.63	61467	21	7.31	318	104	40.40	-
average	462.132	14043.385	22.154	21.147	127.769	52.385	58.727	**77%**
geometric	**72.323**	**5786.490**	12.321	13.793	0.000	27.681	45.917	-

Table 5.6: Running time: phase-S (i.e. sequential phase-2)

				Phase-S + Bounds on α				
name	time	cg_iter	GMIs	GMIs %gc	cuts	tight	%gc	ex.
bell3a	65.21	15098	52.85	39.03	61	29	56.72	-
bell5	47.30	27607	14.53	14.53	91	15	20.50	Y
egout	0.97	966	90.81	31.94	64	22	100.00	Y
flugpl	4.34	361	12.54	10.75	27	9	100.00	Y
lseu	1859.27	27700	52.48	20.48	176	35	80.62	-
misc03	49.00	2473	8.62	8.62	55	9	8.62	Y
mod008	632.03	2182	21.62	21.62	0	1	21.62	-
p0033	4.41	438	71.11	34.42	45	30	100.00	Y
p0201	1386.99	28627	38.08	0.39	150	48	17.83	Y
pp08a	8.94	10552	51.44	51.44	167	112	83.47	Y
set1ch	22.76	13143	38.11	28.20	301	201	90.46	Y
vpm1	6.81	5590	26.99	6.18	181	36	41.59	Y
vpm2	187.12	63974	37.23	7.31	327	101	41.01	-
average	328.858	15285.462	39.724	21.147	126.538	49.846	58.649	**69%**
geometric	**41.790**	**6015.653**	32.527	13.793	0.000	26.513	45.898	-

Table 5.7: Running time: phase-S and bounds on α

5.8 Experimental setup

In this chapter, we aim at obtaining a reasonable evaluation of the percentage of gap closed by cuts from various relaxations. The general layout of our experiment is summarized in Algorithm 6. We work on the 66 instances from MIPLIB 3 [23], which are first preprocessed using the default MIP preprocessing parameters of the solver (CPLEX 12.4). For each instance, we start by optimizing over the LP relaxation and computing the optimal LP tableau. From that tableau, we build the multi-row relaxations that we will use throughout the computation, and compute a round of GMI cuts. After adding these GMIs to the formulation, we enter in the main loop. In the main loop, we try to separate the current LP optimal point with each multi-row model, and add the separating cuts to the formulation.

We now describe how we construct the multi-row models. From a theoretical perspective, closures are a useful tool to evaluate the strength of a family of cutting planes. For a given family of cuts, the first closure is the polyhedron obtained by adding all the cuts of that family that arise from models that are constructed from the LP relaxation of a MIP. In practice, this means that each row of the underlying models can be a linear combination of any set of rows from any tableau of the LP relaxation. The ith closure is obtained by adding cuts from models constructed upon the $i-1$th closure. A cut is said to be rank-i with respect to its family if it is valid for the ith closure.

	Base algorithm				Two-phase				Lifting binaries			
name	time	cg_iter	%gc	ex.	time	cg_iter	%gc	ex.	time	cg_iter	%gc	ex.
`bell3a`	1391.20	45452	49.45	_	347.56	31883	55.76	_	122.04	17572	56.77	_
`bell5`	1283.30	172368	21.40	!	158.76	38588	20.50	_	131.01	27178	20.50	Y
`egout`	2.42	1401	100.00	Y	1.50	863	100.00	Y	0.79	431	100.00	Y
`flugpl`	12.67	705	100.00	Y	5.21	272	100.00	Y	7.48	272	100.00	Y
`lseu`	12834.91	241012	76.64	_	11741.74	156057	82.13	_	2592.50	16016	82.13	Y
`misc03`	2919.36	120048	8.62	_	383.11	27610	8.62	_	75.24	2277	8.62	Y
`mod008`	7385.68	7503	21.62	_	242.83	61	21.62	_	608.23	51	21.62	_
`p0033`	12.46	1575	100.00	Y	4.18	672	100.00	Y	3.77	174	100.00	Y
`p0201`	14400.02	896899	17.83	_	5151.84	199654	17.83	Y	1514.24	23076	17.83	Y
`pp08a`	48.03	35015	83.47	Y	19.69	11375	83.47	Y	14.32	11112	83.47	Y
`set1ch`	7668.69	227506	90.46	_	179.56	26303	90.46	Y	315.11	31078	90.46	Y
`vpm1`	198.51	67630	41.59	Y	29.83	9286	41.59	Y	21.33	9744	41.59	Y
`vpm2`	6404.89	517514	41.02	_	193.88	45999	41.02	_	343.53	61836	41.01	_
avg.	4197.09	179586.8	57.85	46%	1419.98	42201.8	58.69	54%	442.28	15447.5	58.77	77%
geom.	598.81	39956.2	45.39	-	105.77	8548.3	45.90	-	74.33	4080.7	45.97	-

	Callbacks				Bounds on α				Phase-S			
name	time	cg_iter	%gc	ex.	time	cg_iter	%gc	ex.	time	cg_iter	%gc	ex.
`bell3a`	93.97	15068	56.82	_	94.96	15068	56.82	_	65.21	15098	56.72	_
`bell5`	93.79	27272	20.50	Y	91.88	27272	20.50	Y	47.30	27607	20.50	Y
`egout`	0.76	436	100.00	Y	0.56	436	100.00	Y	0.97	966	100.00	Y
`flugpl`	6.93	272	100.00	Y	5.46	272	100.00	Y	4.34	361	100.00	Y
`lseu`	2135.36	13492	77.76	_	2005.63	74796	82.13	Y	1859.27	27700	80.62	_
`misc03`	57.46	2277	8.62	Y	57.82	2277	8.62	Y	49.00	2473	8.62	Y
`mod008`	653.68	511	21.70	_	658.90	511	21.70	_	632.03	2182	21.62	_
`p0033`	3.46	174	100.00	Y	9.69	174	100.00	Y	4.41	438	100.00	Y
`p0201`	296.92	16674	17.83	Y	2259.56	32379	17.83	Y	1386.99	28627	17.83	Y
`pp08a`	13.63	11112	83.47	Y	22.92	11112	83.47	Y	8.94	10552	83.47	Y
`set1ch`	219.20	31078	90.46	Y	340.94	31078	90.46	Y	22.76	13143	90.46	Y
`vpm1`	19.72	9744	41.59	Y	33.95	9744	41.59	Y	6.81	5590	41.59	Y
`vpm2`	291.17	61776	41.01	_	469.82	61776	41.01	_	187.12	63974	41.01	_
avg.	298.93	14606.6	58.44	69%	465.55	20530.4	58.78	77%	328.86	15285.5	58.65	69%
geom.	56.91	4639.5	45.79	-	80.07	5570.0	45.98	-	41.79	6015.7	45.90	-

Table 5.8: Running time: summary

```
**Input:** a mixed-integer problem $P$, its linear relaxation $P_{LP}$

Optimize over $P_{LP}$.
Generate a round of GMIs from the optimal tableau.
Build multi-row models from the optimal tableau.
Add the GMIs to $P_{LP}$

**do**
        Optimize over $P_{LP}$. Let $x^*$ be the optimal solution.

        **for each** multi-row model,
                generate a cut, trying to separate $x^*$.
        **end for**
        Add the separating cuts to $P_{LP}$.
**while** at least one cut was added.
```

Algorithm 6: Main cut-generation loop

Because our separator is still too slow for that purpose, we can not claim to be even approaching the computation of a first closure for the various types of cuts we test. We thus have to pick a way to select the relaxations from which to generate cuts. Our first attempt is to fix one optimal tableau, and build models from all combinations of rows that can be directly read from that tableau. This is a natural choice since in practice, most cutting planes are generated from simplex tableaux, and intersection cuts from an *optimal* tableau are guaranteed to separate the corresponding optimal vertex. This yields $\binom{m}{k}$ k-row models for a MIP with m rows.

Remark that all the relaxations that we consider are strengthenings of the intersection cut model P_I. We assume that in a practical context, the corresponding cuts would be generated with variants of the intersection cut method. For that reason, we will consider only models for which all basic variables are integer-constrained and at least one component of f is fractional.

Throughout this chapter, we build all the models directly from data of the initial problem, hence generating only rank-1 cuts.

Various computational issues can arise when computing cuts with our separator:

(a). The computation of a slave MIP fails:

- (i). The iteration limit is hit:
  - 10^6 branch-and-bound nodes when looking for the point $\tilde{x}$ used in the normalization (see Section 5.3),
  - 10000 nodes when solving a regular slave MIP,
  - 1000 nodes when a feasible solution that corresponds to a violated master constraint is already known (see Section 5.7),
  - 10000 nodes when lifting binary variables or finding bounds on α coefficients (see Section 5.6).
- (ii). The MIP solver detects unboundedness despite the LP relaxation being bounded.
- (iii). The MIP solver claims the problem to be infeasible while we found a feasible solution at a previous iteration.
- (iv). The MIP solver encounters variables whose value is too large to be represented internally with a 32-bit integer.

(b). The computation of the master LP fails:

- (i). The solver claims the master to be unbounded, which is not possible by construction (see Section 5.3).
- (ii). The solver claims the master to be infeasible while lifting a valid inequality, which should not happen either (see Section 5.5).
- (iii). The solver returns a solution that does not satisfy all the constraints provided (violation larger than $\frac{1}{2}10^{-7}$).

(c). Algorithm 4 fails:

- (i). The iteration limit (2500) is hit.
- (ii). The global time limit (4 hours) is hit.

(d). A generated inequality has bad numerical properties or does not separate the point x^*:

- (i). its violation at x^* is less than 10^{-4},
- (ii). its dynamism (i.e. the quotient of the largest and the smallest nonzero cut coefficient) is larger than 10^6, or
- (iii). its efficacy (i.e. the violation divided by the dynamism) is less than 10^{-5}.

(e). We run out of memory.

In most of the above cases, the computation fails for the concerned model but can continue with other models. However, case (b.iii) indicates that we could not trust the solver to provide valid inequalities and we consider that we should abort the computation for the whole instance if it happens. And we can not recover from memory exhaustion (case (e)).

In practice, issues arise for a number of models in almost all of the instances. But as long as, at each iteration of Algorithm 6, one or more models generate separating inequalities, the process can continue. Only the last iteration of the loop matters to determine whether we have exact separation or not. If, at the last iteration, no issues occur, then we have proven that the last point x^* could not be separated by any of the multi-row models, i.e. we have exact separation. Otherwise, we only provide a lower bound on the amount of gap that can be closed with the given selection of multi-row models.

Table 5.9 present the results obtained by running Algorithm 6 with the selection of 2-row models described above. We run our test on 62 of the 65 MIPLIB 3 instances (`dsbmip` and `enigma` have no integrality gap and no optimal solution is known for `dano3mip`). We have results from 55 of the 62 instances (issue (b.iii) occurs four times, issue (c) occurs three times).

We observe that, on average, the amount of gap closed by the two-row cuts (37.493%) is significantly higher than the gap closed by the GMIs (22.596%). This is not surprising however, as we consider full two-row models, i.e. we do not drop any integrality or bound constraint on the variables. The main lesson from Table 5.9 is that the running time is not satisfactory. We hit the global time limit (4 hours) in most instances, and we have exact separation for only 13 of them.

These excessive computing times can be blamed in large part on the sheer number of two-row models we select when taking all suitable pairs of rows from an optimal tableau. Moreover, this problem is bound to get worse when considering relaxations with more than two rows. In order to alleviate the computational burden caused by this policy, we now look for a more restricted selection of models that could nevertheless provide similar results.

As in Chapter 3, we try to construct models whose rows have similar supports. To that end, we design the heuristic row selector summarized in Algorithm 7. It first builds, for each row of the tableau, a cluster

			All 2-row models			
name	GMIs	GMI %gc	time	cuts (t)	%gc	ex.
10teams	100	57.14	14417.29	4	57.14	-
air03	3	100	0.699894	32	100	Y
air04	100	8.60	14400.84	15	8.60	-
air05	100	4.63	14400.98	8	4.63	-
arki001	7	0.00	14400.16	7	0.00	-
bell3a	14	39.03	513.98	120	59.02	Y
bell5	17	14.53	1504.47	34	91.23	Y
blend2	5	16.04	3100.16	39	28.20	Y
cap6000	2	39.91	14400.96	2	39.91	-
danoint	31	0.26	14400.06	65	1.74	-
egout	8	31.94	5.29	26	100.00	Y
fiber	22	69.30	14400.03	704	86.48	-
fixnet6	11	22.26	14400.82	27	33.41	-
flugpl	7	10.75	17.21	8	44.52	Y
gen	6	1.27	14400.41	9	37.81	-
gesa2_o	70	30.68	14400.02	191	45.79	-
gesa3	37	20.48	14400.01	246	51.34	-
gesa3_o	64	50.54	14402.15	228	59.64	-
gt2	11	47.22	14400.49	104	58.33	-
harp2	22	22.81	14404.91	20	22.81	-
khb05250	19	73.16	14412.45	177	91.09	-
l152lav	7	2.01	14400.03	7	3.06	-
lseu	5	20.48	6250.38	54	70.25	Y
markshare1	6	0.00	3737.98	1	0.00	-
markshare2	7	0.00	14400.40	1	0.00	-
mas74	11	6.67	14423.24	11	6.96	-
mas76	10	6.42	14400.15	7	6.44	-
misc03	4	8.62	7798.57	80	9.79	-
misc06	16	28.48	14400.01	191	75.39	-
misc07	5	0.72	6669.73	113	0.88	-
mkc	34	0.96	14400.04	53	24.39	-
mod008	5	21.62	14400.64	12	22.89	-
mod010	5	100.00	14401.19	20	100.00	-
mod011	21	30.69	14448.30	19	30.69	-
noswot	14	0.00	506.48	28	0.00	Y
nw04	2	29.75	14400.17	4	29.90	-
p0033	4	34.42	13.19	65	100.00	Y
p0201	14	0.39	4197.12	74	17.95	Y
p0282	23	3.19	14400.01	140	42.31	-
p0548	31	60.67	1672.68	115	99.88	-
p2756	80	56.96	14400.84	435	95.06	-
pk1	15	0.00	14400.02	3	0.00	-
pp08a	53	51.44	615.71	98	80.41	Y
pp08acuts	41	31.52	14400.01	50	47.71	-
qiu	25	1.69	14408.77	16	1.74	-
qnet1	19	7.18	14400.94	9	7.18	-
qnet1_o	10	26.21	14402.68	11	26.46	-
rentacar	13	4.97	14400.60	13	5.62	-
rgn	12	5.02	3468.94	38	41.96	-
rout	30	1.40	14416.56	8	1.40	-
set1ch	121	28.20	14400.00	165	62.84	-
stein27	21	0.00	2051.46	5	0.00	Y
stein45	16	0.00	14400.01	11	0.00	-
swath	13	6.37	14400.93	17	12.21	-
vpm1	15	6.18	648.42	41	17.06	Y
average	24.800	22.596	10467.629	72.382	37.493	24%
geometric	14.983	0.000	4961.177	27.222	0.000	-

Table 5.9: Cuts from all full two-row models directly read from an optimal tableau

```
Input: A $m$-row simplex tableau and $k$, the target number of rows per model

for each row $i$ of the tableau,
    build a cluster of $5k - 1$ rows having a support most similar to row $i$
end for

$i := 1$.
while models < MODELS_MAX,
    Find a subset of $k - 1$ rows in the cluster $i$ such that the $k$-row model
    built from that subset and $i$ does not already form a selected model.

    if such a subset exist,
        build a $k$-row model from these rows and row $i$
    end if

    $i := (i + 1) \mod m$.
end
```

Algorithm 7: Heuristic selection of multi-row models

of other rows that have a similiar support. Then, it loops through the clusters constructing one multi-row model with the rows in each one. Our intent with this heuristic is to favor models from rows with similar supports, while covering all rows of the tableau. Note that the constraints mentioned earlier for each row to have an integer-constrained basic variable, and for each model to have a fractional-valued basic variable, are incorporated into the heuristic.

At line 4 of Algorithm 7, we need to define a measure for the similarity of the supports of two rows. We test two options. Let S_c be the number of columns in which both rows have nonzero coefficients, and S_d the number of columns in which exactly one row has a nonzero coefficients. Our first option is to maximize $S_c/(S_c + S_d)$, i.e. the cardinality of the intersection of the supports divided by the cardinality of the union of the supports. The second option is simply to maximize $S_c - S_d$, i.e. score positively columns where both rows have nonzero coefficients and penalize columns where one row has a zero and the other not.

As Table 5.10 suggests, we adopt $S_c - S_d$ in the rest of this chapter. Indeed, compared to $S_c/(S_c + S_d)$ it yields more percentage of gap closed, more instances with exact separation, and offers reduced computing times. Compared to when reading all two-row models from an optimal tableau, the average percentage of gap closed drops from 37.493% to 35.193%, but the running times are divided by a factor 4 in geometric mean, and the number of instances with exact separation almost doubles. Table 5.11 presents detailed results for $S_c - S_d$.

In Table 5.10 and Table 5.11, `MODELS_MAX` is set to the number m of rows in the formulation of the problem. In all later tables, `MODELS_MAX` $= \frac{m}{2}$. Note that in all the tables in the remainder of this chapter, each row corresponds to an instance for which every test presented succeeded. For example, in Table 5.10, the `vpm2` does not appear because the computation using all 2-row models failed on that instance. But in Table 5.11, where the experiment with all 2-row models is not presented, `vpm2` appears with the corresponding data. This may result in slight variations in the testset, and different average

	All 2-row models				Heuristic selection, $S_c/(S_c+S_d)$				Heuristic selection, $S_c - S_d$			
name	time	cuts (t)	%gc	ex.	time	cuts (t)	%gc	ex.	time	cuts (t)	%gc	ex.
10teams	14417.29	4	57.14	_	14431.48	4	57.14	_	14460.59	4	57.14	_
air03	0.699894	32	100	Y	1.01285	31	100	Y	1.08284	32	100	Y
air04	14400.84	15	8.60	_	14400.97	13	7.90	_	14400.55	15	8.60	_
air05	14400.98	8	4.63	_	14400.98	8	4.63	_	14400.97	8	4.63	_
arki001	14400.16	7	0.00	_	14401.20	7	0.00	_	14400.50	7	0.00	_
bell3a	513.98	120	59.02	Y	30.79	25	50.49	Y	23.37	29	55.69	Y
bell5	1504.47	34	91.23	Y	488.96	18	20.80	Y	93.83	20	19.32	Y
blend2	3100.16	39	28.20	Y	315.29	10	20.00	Y	337.78	10	20.00	Y
cap6000	14400.96	2	39.91	_	14400.95	2	39.91	_	14400.96	2	39.91	_
danoint	14400.06	65	1.74	_	12124.39	56	1.74	_	11614.80	56	1.74	_
egout	5.29	26	100.00	Y	2.77	29	84.99	Y	1.75	19	98.36	Y
fiber	14400.03	704	86.48	_	7322.20	383	82.65	_	3675.17	473	82.65	Y
fixnet6	14400.82	27	33.41	_	14400.02	43	37.91	_	8876.15	54	36.54	_
flugpl	17.21	8	44.52	Y	5.73	7	12.74	Y	10.95	8	44.52	Y
gen	14400.41	9	37.81	_	14400.18	10	37.81	_	14400.59	9	37.81	_
gesa2_o	14400.02	191	45.79	_	6274.83	270	68.37	Y	4503.62	250	47.96	_
gesa3	14400.01	246	51.34	_	14403.57	100	57.53	_	5002.46	178	56.11	_
gesa3_o	14402.15	228	59.64	_	14400.15	107	55.96	_	13677.70	287	77.81	Y
gt2	14400.49	104	58.33	_	14400.15	15	56.78	_	14400.03	92	58.25	_
harp2	14404.91	20	22.81	_	14402.02	20	22.81	_	14401.42	20	22.81	_
khb05250	14412.45	177	91.09	_	14403.60	54	80.13	_	14408.54	177	91.09	_
l152lav	14400.03	7	3.06	_	14400.01	4	3.06	_	14400.36	8	2.68	_
lseu	6250.38	54	70.25	Y	14400.72	16	50.82	_	5937.95	47	70.25	Y
markshare1	3737.98	1	0.00	_	1182.30	1	0.00	_	1332.16	1	0.00	_
markshare2	14400.40	1	0.00	_	3338.39	1	0.00	_	3884.63	1	0.00	_
mas74	14423.24	11	6.96	_	14400.09	11	6.96	_	14400.34	11	6.96	_
mas76	14400.15	7	6.44	_	6141.47	6	6.42	_	5773.43	6	6.42	_
misc03	7798.57	80	9.79	_	61.39	41	8.62	Y	87.83	9	8.62	Y
misc06	14400.01	191	75.39	_	30.21	13	28.66	Y	142.28	22	48.81	Y
misc07	6669.73	113	0.88	_	5.24	9	0.72	Y	8.06	13	0.72	Y
mkc	14400.04	53	24.39	_	14400.74	37	0.96	_	14403.26	53	24.39	_
mod008	14400.64	12	22.89	_	201.26	1	21.62	_	191.87	1	21.62	_
mod010	14401.19	20	100.00	_	14400.19	19	100.00	_	14400.54	22	100.00	_
mod011	14448.30	19	30.69	_	14409.35	16	30.69	_	14463.60	19	30.69	_
noswot	506.48	28	0.00	Y	10.11	16	0.00	Y	8.00	15	0.00	Y
nw04	14400.17	4	29.90	_	14406.33	2	29.75	_	14400.42	4	29.90	_
p0033	13.19	65	100.00	Y	8.70	61	100.00	Y	13.89	65	100.00	Y
p0201	4197.12	74	17.95	Y	2814.48	119	17.83	Y	2767.20	34	16.00	Y
p0282	14400.01	140	42.31	_	8275.55	196	37.60	_	6508.40	152	41.92	_
p0548	1672.68	115	99.88	_	164.52	158	99.88	Y	30.96	116	99.88	Y
p2756	14400.84	435	95.06	_	14400.05	223	77.13	_	1295.19	374	77.21	Y
pk1	14400.02	3	0.00	_	12488.11	1	0.00	_	12718.23	1	0.00	_
pp08a	615.71	98	80.41	Y	46.54	77	70.16	Y	24.91	88	76.19	Y
pp08acuts	14400.01	50	47.71	_	5486.96	41	37.07	_	4148.46	44	46.86	_
qiu	14408.77	16	1.74	_	14401.84	16	1.69	_	14400.21	29	1.71	_
qnet1	14400.94	9	7.18	_	14400.19	9	7.18	_	14401.49	6	15.76	_
qnet1_o	14402.68	11	26.46	_	14402.14	6	26.21	_	14402.62	11	26.46	_
rentacar	14400.60	13	5.62	_	14419.49	11	5.23	_	14400.10	13	5.23	_
rgn	3468.94	38	41.96	_	1445.34	34	25.10	Y	2044.03	31	25.10	Y
rout	14416.56	8	1.40	_	14400.95	6	1.41	_	14400.56	9	1.40	_
set1ch	14400.00	165	62.84	_	312.62	140	58.84	Y	174.68	153	61.75	Y
stein27	2051.46	5	0.00	Y	10.01	4	0.00	Y	37.44	4	0.00	Y
stein45	14400.01	11	0.00	_	259.44	3	0.00	Y	256.55	3	0.00	Y
swath	14400.93	17	12.21	_	14421.51	18	14.01	_	14400.95	17	12.21	_
vpm1	648.42	41	17.06	Y	17.92	31	15.95	Y	13.91	26	15.95	Y
average	10467.629	72.382	37.493	24%	8061.371	46.527	32.434	36%	7232.133	57.418	35.193	42%
geometric	4961.177	27.222	0.000	-	1570.310	17.359	0.000	-	1387.383	19.901	0.000	-

Table 5.10: Heuristic selection of models, two options for defining "similar" supports

values, e.g. for the number of GMIs generated, which would not change otherwise.

5.9 Strengthened intersection cuts

Armed with a reasonable heuristic model selector, we can now compare the strength of various relaxations of a given set of two-row models. In this section we again limit ourselves to two-row models, and test in more detail the classes of models presented in Chapter 4.

			Heuristic selection, $S_c - S_d$			
name	GMIs	GMI %gc	time	cuts (t)	%gc	ex.
10teams	100	57.14	14460.59	4	57.14	-
air03	3	100	1.08284	32	100	Y
air04	100	8.60	14400.55	15	8.60	-
air05	100	4.63	14400.97	8	4.63	-
arki001	7	0.00	14400.50	7	0.00	-
bell3a	14	39.03	23.37	29	55.69	Y
bell5	17	14.53	93.83	20	19.32	Y
blend2	5	16.04	337.78	10	20.00	Y
cap6000	2	39.91	14400.96	2	39.91	-
danoint	32	0.26	11614.80	56	1.74	-
dcmulti	36	47.80	2371.66	200	59.75	Y
egout	8	31.94	1.75	19	98.36	Y
fiber	22	69.30	3675.17	473	82.65	Y
fixnet6	11	22.26	8876.15	54	36.54	-
flugpl	7	10.75	10.95	8	44.52	Y
gen	6	1.27	14400.59	9	37.81	-
gesa2_o	70	30.68	4503.62	250	47.96	-
gesa3	37	20.48	5002.46	178	56.11	-
gesa3_o	64	50.54	13677.70	287	77.81	Y
gt2	11	47.22	14400.03	92	58.25	-
harp2	22	22.81	14401.42	20	22.81	-
khb05250	19	73.16	14408.54	177	91.09	-
l152lav	7	2.01	14400.36	8	2.68	-
lseu	5	20.48	5937.95	47	70.25	Y
markshare1	6	0.00	1332.16	1	0.00	-
markshare2	7	0.00	3884.63	1	0.00	-
mas74	11	6.67	14400.34	11	6.96	-
mas76	10	6.42	5773.43	6	6.42	-
misc03	4	8.62	87.83	9	8.62	Y
misc06	16	28.48	142.28	22	48.81	Y
misc07	5	0.72	8.06	13	0.72	Y
mkc	34	0.96	14403.26	53	24.39	-
mod008	5	21.62	191.87	1	21.62	-
mod010	5	100.00	14400.54	22	100.00	-
mod011	21	30.69	14463.60	19	30.69	-
modglob	28	17.28	261.52	35	31.92	-
noswot	14	0.00	8.00	15	0.00	Y
nw04	2	29.75	14400.42	4	29.90	-
p0033	4	34.42	13.89	65	100.00	Y
p0201	14	0.39	2767.20	34	16.00	Y
p0282	23	3.19	6508.40	152	41.92	-
p0548	31	60.67	30.96	116	99.88	Y
p2756	80	56.96	1295.19	374	77.21	Y
pk1	15	0.00	12718.23	1	0.00	-
pp08a	53	51.44	24.91	88	76.19	Y
pp08acuts	41	31.52	4148.46	44	46.86	-
qiu	23	1.70	14400.21	29	1.71	-
qnet1	17	15.76	14401.49	6	15.76	-
qnet1_o	10	26.21	14402.62	11	26.46	-
rentacar	13	4.97	14400.10	13	5.23	-
rgn	12	5.02	2044.03	31	25.10	Y
rout	30	1.40	14400.56	9	1.40	-
set1ch	121	28.20	174.68	153	61.75	Y
stein27	21	0.00	37.44	4	0.00	Y
stein45	16	0.00	256.55	3	0.00	Y
swath	13	6.37	14400.95	17	12.21	-
vpm1	15	6.18	13.91	26	15.95	Y
vpm2	21	7.31	71.02	47	33.59	Y
average	24.931	22.823	6904.681	59.310	35.533	43%
geometric	15.423	0.000	1292.591	21.223	0.000	-

Table 5.11: Heuristic selection of models ($S_c - S_d$)

	basic		nonbasic	
	$\in \mathbb{Z}$	bnd.	$\in \mathbb{Z}$	bnd.
P_I	$\surd$	$\times$	$\times$	`T`
S-free	$\surd$	$\surd$	$\times$	`T`
lifting	$\surd$	$\times$	$\surd$	`T`
P_{IU}	$\surd$	$\times$	$\times$	$\surd$

Table 5.12: Relaxations; $\surd$: keep, `T`: keep tight, $\times$: drop

Recall that P_I is the intersection cut model for which we developed a separator in Chapter 3 in the two-row case, where in particular

$$P_I = \{(x, s) \in \mathbb{Z}^2 \times \mathbb{R}^n_+ : x = f + Rs\}.$$

We can strengthen the intersection cut model by reintroducing bounds on the basic variables x, yielding

$$P_{S\text{-free}} = \{(x, s) \in S \times \mathbb{R}^n_+ : x = f + Rs\}.$$

We can also exploit the integrality of the nonbasic variables by solving the so-called lifting problem, and obtain valid inequalities for

$$P_{\text{lifting}} = \{(x, s) \in \mathbb{Z}^2 \times \mathbb{R}^n_+ : x = f + Rs,\ s_j \in \mathbb{Z} \text{ for all } j \in J\}.$$

Finally, taking advantage of upper bounds on the nonbasic variables, we can generate inequalities that are valid for

$$P_{IU} = \{(x, s) \in \mathbb{Z}^2 \times \mathbb{R}^n_+ : x = f + Rs, s \leq U\}.$$

If we denote by P_{full} a two-row model where we keep all integrality and bound constraints on the variables, P_I consists in dropping from P_{full} bounds on the two basic variables, integrality on nonbasic variables, and any non-binding bound on the nonbasic variables. This is shown in Table 5.12, along with the constraints that are reintroduced in $P_{S\text{-free}}$, P_{lifting} and P_{IU}.

Before we get to its various strengthenings, we briefly focus on the two-row model P_I for which we already presented results in Chapter 3. Using a generic separator on the same model will let us crosscheck our implementations, and will provide a useful comparison point for confronting the speed of our generic separator with a fast, single-purpose algorithm. Table 5.13 enables such comparison.

We observe that despite the two implementations using different sets of tolerances and thresholds, the results are consistent within 0.05% of gap closed. That is, within the tolerance, when one of the methods provides exact separation at G %gc, the other does not exceed G %gc, and when both method provide exact separation, they both close the same amount of gap.

One exception is `seymour`, where the separator presented in this chapter does not close any gap on top of GMIs, and claims that none more could be closed with valid inequalities, while our intersection cut separator closes four more percents of gap. This could be caused by an invalid inequality being generated by our intersection cut separator due to numerically unsafe coefficients, or a valid inequality being discarded by our generic separator because of numerical safeguards.

			P_I via intersection cut separator				P_I via generic separator			
name	GMIs	GMI %gc	time	cuts (t)	%gc	ex.	time	cuts (t)	%gc	ex.
10teams	100	57.14	3.23	4	57.14	Y	7.67	4	57.14	Y
air04	100	8.60	61.16	18	8.66	_	14401.86	16	8.60	_
air05	100	4.63	45.72	8	4.78	Y	3756.06	8	4.78	Y
arki001	7	0.00	4.23	12	0.00	_	14402.53	13	3.97	_
bell3a	14	39.03	0.05	9	52.48	Y	12.51	10	52.53	Y
bell5	17	14.53	0.05	8	14.92	Y	7.71	8	14.92	Y
blend2	5	16.04	0.06	3	16.90	Y	94.51	3	16.90	Y
cap6000	2	39.91	1.15	6	40.07	Y	14400.65	3	40.07	_
danoint	31	0.26	5.69	39	1.74	_	14400.03	39	1.74	_
dcmulti	36	47.80	0.37	44	49.92	Y	261.05	40	49.93	_
egout	8	31.94	0.04	8	60.54	Y	7.31	8	60.54	Y
fiber	22	69.30	0.19	26	71.08	Y	353.75	19	69.31	_
fixnet6	11	22.26	0.16	18	30.64	Y	302.24	18	30.57	_
flugpl	7	10.75	0.02	7	13.02	Y	4.05	7	13.04	Y
gen	6	1.27	0.20	13	9.53	_	714.17	29	6.82	_
gesa2	40	27.56	1.38	91	56.48	_	8003.63	108	56.58	_
gesa2_o	70	30.68	1.44	136	32.87	Y	500.50	126	32.87	Y
gesa3	37	20.48	1.38	142	47.06	Y	3057.59	110	36.68	_
gesa3_o	64	50.54	1.54	146	69.36	Y	2345.12	123	61.78	_
gt2	11	47.22	0.03	8	47.22	Y	1.74	8	47.22	Y
harp2	22	22.81	0.39	20	25.20	_	1407.92	22	25.49	Y
khb05250	19	73.16	0.04	15	73.16	Y	1.00	15	73.16	Y
l152lav	7	2.01	1.05	14	2.04	_	3795.00	12	2.04	_
lseu	5	20.48	0.02	8	21.01	Y	6.23	8	21.01	Y
mas74	11	6.67	0.03	7	6.67	Y	0.25	7	6.67	Y
mas76	10	6.42	0.03	6	6.42	Y	0.23	6	6.42	Y
misc03	4	8.62	0.07	7	8.62	Y	0.35	7	8.62	Y
misc06	16	28.48	0.28	17	48.27	_	116.79	16	47.44	_
misc07	5	0.72	0.08	9	0.72	Y	0.85	9	0.72	Y
mitre	101	50.71	4.09	476	81.71	_	3802.97	490	81.71	_
mkc	34	0.96	2.73	89	2.64	Y	14400.01	63	2.61	_
mod008	5	21.62	0.12	1	21.62	Y	0.91	1	21.62	Y
mod011	21	30.69	3.74	41	36.06	Y	14400.16	24	32.24	_
modglob	28	17.28	0.30	29	26.13	Y	1654.06	27	25.82	_
nw04	2	29.75	1.87	2	29.75	Y	26.69	2	29.75	Y
p0033	4	34.42	0.01	5	34.81	Y	0.78	5	34.81	Y
p0201	14	0.39	0.06	6	6.52	Y	92.69	22	0.39	_
p0282	23	3.19	0.22	22	4.63	Y	375.36	17	4.63	_
p0548	31	60.67	0.07	33	60.72	Y	36.07	33	60.72	_
p2756	80	56.96	0.53	80	58.33	Y	29.93	80	58.33	_
pp08a	53	51.44	0.08	78	75.18	Y	538.10	61	60.57	_
pp08acuts	41	31.52	0.52	40	38.78	Y	507.79	36	35.02	_
qiu	25	1.69	1.63	22	1.87	_	14400.20	16	1.74	_
qnet1	19	7.18	4.23	18	14.28	Y	3512.39	21	10.44	_
qnet1_o	10	26.21	0.44	15	31.49	Y	653.89	18	30.66	_
rentacar	13	4.97	11.72	11	5.45	_	14402.83	10	4.97	_
rgn	12	5.02	0.02	8	5.02	Y	129.28	5	5.02	_
rout	30	1.40	3.69	8	1.51	Y	200.03	8	1.51	Y
set1ch	121	28.20	0.17	126	60.63	Y	15.68	126	60.64	Y
seymour	60	4.98	13.33	11	8.43	Y	884.90	19	4.98	Y
swath	13	6.37	1.24	40	6.65	Y	14400.06	28	6.50	_
vpm1	15	6.18	0.09	18	14.56	Y	140.71	18	14.56	_
vpm2	21	7.31	0.13	22	17.15	Y	9790.34	21	17.15	_
average	29.491	22.612	3.417	38.679	28.688	79%	3335.078	36.849	27.622	42%
geometric	18.150	0.000	0.389	17.524	0.000	-	194.223	17.092	15.266	-

Table 5.13: crosschecking the separators

				P_I			$P_{S\text{-free}}$			P_{lifting}	
name	GMIs	GMI %gc	cuts (t)	%gc	ex.	cuts (t)	%gc	ex.	cuts (t)	%gc	ex.
`10teams`	100	57.14	4	57.14	Y	4	57.14	Y	4	57.14	_
`air04`	100	8.60	18	8.66	_	23	8.78	_	15	8.60	_
`air05`	100	4.63	8	4.78	Y	12	5.12	_	8	4.63	_
`bell3a`	14	39.03	9	52.48	Y	38	54.90	Y	10	52.45	_
`bell5`	17	14.53	8	14.92	Y	10	14.93	Y	10	15.11	Y
`blend2`	5	16.04	3	16.90	Y	5	17.03	Y	4	17.22	Y
`cap6000`	2	39.91	6	40.07	Y	6	40.07	_	2	39.91	_
`danoint`	32	0.26	39	1.74	_	64	1.74	_	63	1.74	_
`dcmulti`	36	47.80	44	49.92	Y	50	54.83	Y	50	51.13	_
`egout`	8	31.94	8	60.54	Y	9	61.67	Y	8	60.54	Y
`fiber`	22	69.30	26	71.08	Y	27	71.09	Y	15	69.30	_
`fixnet6`	11	22.26	18	30.64	Y	19	31.18	Y	18	30.57	_
`flugpl`	7	10.75	7	13.02	Y	7	13.00	_	6	42.04	Y
`gen`	6	1.27	13	9.53	_	18	24.56	Y	17	8.13	_
`gesa2`	40	27.56	91	56.48	_	134	56.87	_	59	33.99	_
`gesa2_o`	70	30.68	136	32.87	Y	141	32.75	_	92	35.17	_
`gesa3`	37	20.48	142	47.06	Y	252	54.23	_	40	32.82	_
`gesa3_o`	64	50.54	146	69.36	Y	170	72.88	_	103	58.68	_
`gt2`	11	47.22	8	47.22	Y	11	49.63	Y	9	51.11	_
`harp2`	22	22.81	20	25.20	_	23	26.65	Y	20	22.81	_
`khb05250`	19	73.16	15	73.16	Y	15	73.16	Y	15	73.16	Y
`l152lav`	7	2.01	14	2.04	_	19	8.52	_	5	2.01	_
`lseu`	5	20.48	8	21.01	Y	8	21.04	Y	7	21.06	_
`mas74`	11	6.67	7	6.67	Y	7	6.67	Y	7	6.67	_
`mas76`	10	6.42	6	6.42	Y	6	6.42	Y	6	6.42	_
`misc03`	4	8.62	7	8.62	Y	7	8.62	Y	7	8.62	Y
`misc06`	16	28.48	17	48.27	_	18	48.81	Y	16	47.44	_
`misc07`	5	0.72	9	0.72	Y	12	0.72	Y	9	0.72	Y
`mkc`	34	0.96	89	2.64	Y	107	26.99	Y	35	0.96	_
`mod008`	5	21.62	1	21.62	Y	1	21.62	Y	2	21.79	_
`mod011`	21	30.69	41	36.06	Y	16	30.69	_	22	32.24	_
`modglob`	28	17.28	29	26.13	Y	35	26.48	_	29	26.61	_
`nw04`	2	29.75	2	29.75	Y	2	29.75	_	2	29.75	Y
`p0033`	4	34.42	5	34.81	Y	7	44.28	Y	6	45.83	_
`p0201`	14	0.39	6	6.52	Y	4	8.49	Y	10	1.34	_
`p0282`	23	3.19	22	4.63	Y	24	13.82	Y	8	4.41	_
`p0548`	31	60.67	33	60.72	Y	31	61.01	Y	35	64.30	_
`p2756`	80	56.96	80	58.33	Y	79	67.06	_	79	59.44	_
`pp08a`	53	51.44	78	75.18	Y	83	75.18	_	61	60.57	_
`pp08acuts`	41	31.52	40	38.78	Y	46	41.02	_	34	38.31	_
`qiu`	25	1.69	22	1.87	_	40	2.05	_	16	1.74	_
`qnet1`	19	7.18	18	14.28	Y	70	24.22	_	7	7.50	_
`qnet1_o`	10	26.21	15	31.49	Y	52	39.35	Y	9	28.87	_
`rentacar`	13	4.97	11	5.45	_	15	5.47	Y	10	4.97	_
`rgn`	12	5.02	8	5.02	Y	9	5.02	_	5	5.02	_
`rout`	30	1.40	8	1.51	Y	61	6.87	_	7	1.51	_
`set1ch`	121	28.20	126	60.63	Y	143	60.72	Y	126	60.64	Y
`seymour`	57	8.05	11	8.43	Y	20	11.37	_	11	8.43	Y
`swath`	13	6.37	40	6.65	Y	47	6.66	_	16	6.37	_
`vpm1`	15	6.18	18	14.56	Y	19	14.95	Y	18	14.56	_
`vpm2`	21	7.31	22	17.15	Y	27	22.00	Y	21	17.15	_
average	28.490	22.564	30.627	28.210	82%	40.255	30.747	57%	23.412	27.480	20%

Table 5.14: Strengthenings of P_I, part 1

Regarding running times, the intersection cut separator from Chapter 3 is almost three orders of magnitude faster than the generic separator presented here, while performing exact separation on almost three times as many instances.

Table 5.14 and Table 5.15 show the gap closed with the same selection of two-row models, but different relaxations, namely P_I, $P_{S\text{-free}}$, P_{lifting}, P_{IU} and P_{full}. Unfortunately, it is extremely hard for Algorithm 4 to separate from P_{lifting}, as indicated by the very low number of instances with exact separation for these models. Intuitively this can be explained by the fact that to build P_{lifting} from P_{full}, we drop a number of bound constraints on the variables, but keep all integrality constraints (see Table 5.12). And while all the difficulty in solving MIPs comes from the integrality of the variables, the bound constraints are

			P_I			P_{IU}			P_{full}		
name	GMIs	GMI %gc	cuts (t)	%gc	ex.	cuts (t)	%gc	ex.	cuts (t)	%gc	ex.
`10teams`	101	57.14	4	57.14	Y	4	57.14	Y	4	57.14	_
`air04`	100	8.60	18	8.66	_	19	8.67	_	15	8.60	_
`air05`	100	4.63	8	4.78	Y	8	4.78	Y	8	4.63	_
`arki001`	7	0.00	12	0.00	_	15	0.00	_	7	0.00	_
`bell3a`	14	39.03	9	52.48	Y	10	52.91	Y	18	55.69	Y
`bell5`	17	14.53	8	14.92	Y	11	19.25	Y	13	19.32	Y
`blend2`	5	16.04	3	16.90	Y	11	19.81	Y	10	20.00	Y
`cap6000`	2	39.91	6	40.07	Y	2	39.91	_	2	39.91	_
`danoint`	31	0.26	39	1.74	_	82	1.74	_	58	1.74	_
`dcmulti`	36	47.80	44	49.92	Y	70	52.59	Y	104	58.05	Y
`egout`	8	31.94	8	60.54	Y	15	62.24	Y	16	62.24	Y
`fiber`	22	69.30	26	71.08	Y	41	71.72	Y	262	82.39	Y
`fixnet6`	11	22.26	18	30.64	Y	34	35.35	_	54	36.54	_
`flugpl`	7	10.75	7	13.02	Y	8	13.04	_	6	43.38	Y
`gen`	6	1.27	13	9.53	_	12	9.92	Y	9	37.81	_
`gesa2_o`	70	30.68	136	32.87	Y	224	33.18	_	227	36.08	Y
`gesa3`	37	20.48	142	47.06	Y	210	47.71	Y	178	56.11	_
`gesa3_o`	64	50.54	146	69.36	Y	122	53.66	_	224	77.74	_
`gt2`	11	47.22	8	47.22	Y	12	51.02	Y	58	58.37	_
`harp2`	22	22.81	20	25.20	_	27	23.05	_	20	22.81	_
`khb05250`	19	73.16	15	73.16	Y	140	91.43	Y	132	91.43	Y
`l152lav`	7	2.01	14	2.04	_	18	2.57	_	5	3.06	_
`lseu`	5	20.48	8	21.01	Y	8	21.04	Y	18	67.62	Y
`mas74`	11	6.67	7	6.67	Y	9	6.72	Y	22	11.53	_
`mas76`	10	6.42	6	6.42	Y	6	6.43	Y	6	6.42	_
`misc03`	4	8.62	7	8.62	Y	7	8.62	Y	9	8.62	Y
`misc06`	16	28.48	17	48.27	_	20	48.27	_	22	48.81	Y
`misc07`	5	0.72	9	0.72	Y	9	0.72	Y	13	0.72	Y
`mitre`	101	50.71	476	81.71	_	246	50.71	_	666	73.61	_
`mkc`	34	0.96	89	2.64	Y	143	0.99	_	53	24.39	_
`mod008`	5	21.62	1	21.62	Y	3	21.64	Y	1	21.62	_
`mod011`	21	30.69	41	36.06	Y	17	30.69	_	19	30.69	_
`modglob`	28	17.28	29	26.13	Y	33	26.13	_	42	30.99	_
`nw04`	2	29.75	2	29.75	Y	2	29.75	_	4	29.90	_
`p0033`	4	34.42	5	34.81	Y	10	38.02	Y	32	71.45	Y
`p0201`	14	0.39	6	6.52	Y	9	6.52	Y	24	15.73	Y
`p0282`	23	3.19	22	4.63	Y	95	14.97	_	194	41.51	Y
`p0548`	31	60.67	33	60.72	Y	77	87.87	_	96	99.88	_
`p2756`	80	56.96	80	58.33	Y	123	69.95	Y	323	77.21	Y
`pp08a`	53	51.44	78	75.18	Y	61	60.57	_	84	75.18	Y
`pp08acuts`	41	31.52	40	38.78	Y	39	35.60	_	44	44.13	_
`qiu`	23	1.70	22	1.87	_	17	1.80	_	29	1.71	_
`qnet1`	17	15.76	18	14.28	Y	10	7.18	_	6	15.76	_
`qnet1_o`	10	26.43	15	31.49	Y	21	29.47	_	12	26.97	_
`rentacar`	13	4.97	11	5.45	_	10	6.45	_	12	5.25	_
`rgn`	12	5.02	8	5.02	Y	13	5.02	Y	30	25.10	Y
`rout`	30	1.40	8	1.51	Y	8	1.41	_	8	1.40	_
`set1ch`	121	28.20	126	60.63	Y	142	60.65	Y	143	60.75	Y
`seymour`	60	4.98	11	8.43	Y	55	5.06	_	24	9.12	_
`swath`	13	6.37	40	6.65	Y	39	6.61	_	17	12.21	_
`vpm1`	15	6.18	18	14.56	Y	26	15.35	Y	26	15.95	Y
`vpm2`	21	7.31	22	17.15	Y	40	22.99	Y	39	29.38	Y
average	29.231	22.686	37.673	28.153	81%	46.019	28.440	48%	66.308	35.705	42%

Table 5.15: Strengthenings of P_I, part 2

			P_I		$P_{S\text{-free}}$		P_{lifting}		P_{IU}		P_{full}	
name	GMIs	GMI %gc	cuts (t)	%gc	cuts (t)	%gc	cuts (t)	%gc	cuts (t)	%gc	cuts (t)	%gc
bell5	17	14.53	8	14.92	10	14.93	10	15.11	11	19.25	13	19.32
blend2	5	16.04	3	16.90	5	17.03	4	17.22	11	19.81	10	20.00
egout	8	31.94	8	60.54	9	61.67	8	60.54	15	62.24	16	62.24
khb05250	19	73.16	15	73.16	15	73.16	15	73.16	140	91.43	132	91.43
misc03	4	8.62	7	8.62	7	8.62	7	8.62	7	8.62	9	8.62
misc07	5	0.72	9	0.72	12	0.72	9	0.72	9	0.72	13	0.72
set1ch	121	28.20	126	60.63	143	60.72	126	60.64	142	60.65	143	60.75
average	25.571	24.744	25.143	33.641	28.714	33.836	25.571	33.716	47.857	37.531	48.000	37.583

Table 5.16: Strengthings of P_I, exact separation

			P_I		$P_{S\text{-free}}$		P_{IU}		P_{full}	
name	GMIs	GMI %gc	cuts (t)	%gc	cuts (t)	%gc	cuts (t)	%gc	cuts (t)	%gc
bell3a	14	39.03	9	52.48	38	54.90	10	52.91	18	55.69
bell5	17	14.53	8	14.92	10	14.93	11	19.25	13	19.32
blend2	5	16.04	3	16.90	5	17.03	11	19.81	10	20.00
dcmulti	36	47.80	44	49.92	50	54.83	70	52.59	104	58.05
egout	8	31.94	8	60.54	9	61.67	15	62.24	16	62.24
fiber	22	69.30	26	71.08	27	71.09	41	71.72	262	82.39
khb05250	19	73.16	15	73.16	15	73.16	140	91.43	132	91.43
lseu	5	20.48	8	21.01	8	21.04	8	21.04	18	67.62
misc03	4	8.62	7	8.62	7	8.62	7	8.62	9	8.62
misc07	5	0.72	9	0.72	12	0.72	9	0.72	13	0.72
p0033	4	34.42	5	34.81	7	44.28	10	38.02	32	71.45
p0201	14	0.39	6	6.52	4	8.49	9	6.52	24	15.73
set1ch	121	28.20	126	60.63	143	60.72	142	60.65	143	60.75
vpm1	15	6.18	18	14.56	19	14.95	26	15.35	26	15.95
vpm2	21	7.31	22	17.15	27	22.00	40	22.99	39	29.38
average	20.667	26.541	20.933	33.535	25.400	35.229	36.600	36.257	57.267	43.956

Table 5.17: Strengthings of P_I (except lifting), exact separation

very useful to methods like the branch-and-bound to reduce the enumeration space. As a consequence, the average percentage of gap closed is even lower for the P_{lifting} models than for the weaker P_I models, greatly impairing the meaningfulness of the results.

We can avoid this issue by looking only at instances for which the separation is exact for all of the tests, at the cost of vastly reducing the testset. On the seven instances of Table 5.16, two-row intersection cuts close nine more percents of gap (totalling 33.641%) on top of GMIs alone (24.744%), and fully-strengthened two-row cuts add four more percents (37.583%). Individual strengthenings do not seem to bring a lot of improvement over intersection cuts. Indeed, P_{IU} seems to close almost as much gap as P_{full} in Table 5.16, but enlarging the testset (by omitting the column P_{lifting}), we observe that P_{IU} is only promising for a few instances (see Table 5.17). The more representative testset in Table 5.15 confirms that trend, with P_{IU} closing only 28.440% of gap, up from 28.153% for P_I. Contrary to P_{lifting}, P_{IU} is not unfairly disadvantaged by our separator, as is confirmed by the gap closed by P_{full} jumping to 35.705% despite featuring fewer instances with exact separation than P_{IU}.

While the situation for P_{lifting} is less obvious, Table 5.14 and Table 5.15 show that on average on MIPLIB 3, separating cuts from an optimal tableau, GMIs close 22.686% of gap, two-row intersection cuts close 28.153%, partially strengthened two-row intersection cuts close around 30%, and fully strengthened two-row cuts close 35.705%. We concluded in Chapter 3 that two-row intersection cuts were not strong enough to consistently beat GMIs if considering GMIs from linear combinations of rows of the simplex tableau. The results in this chapter tend to show that only a fully-strengthened intersection cut model could be significantly stronger than P_I, and thus represent a possibility for generating useful cuts beyond GMIs.

5.10 Cuts from several tableaux

In all the computations performed so far, we limited ourselves to rank-1 valid inequalities from *one* (optimal) simplex tableau. We mentioned in the previous section that most of the two-row cuts we generated in Chapter 3 could be obtained as GMIs from linear combinations of rows of the tableau. And the various simplex tableaux describing a linear problem are precisely linear combinations of the rows of each other. Moreover, we noted in Chapter 2 that the most-violated inequality for a simple disjunction on one variable could be obtained through the lift-and-project method as a GMI from a specific simplex tableau. This motivates the need to reproduce our results for more than one basis of the LP relaxation, i.e. for cuts from more than one tableau.

In particular, the small but consistent advantage that two-row intersection cuts enjoy over GMIs read directly from the tableau in terms of gap closure could vanish in the presence of GMIs from other bases. This could be the case, for example, if two-row cuts close more gap only because so many more of them can be generated. Or, to the contrary, the two-row cuts could become significantly stronger when read from several bases.

In order to answer these questions, we need a method to explore different bases of the LP relaxation. We use the relax-and-cut method proposed by Fischetti and Salvagnin [43] in the context of the generation of GMI cuts. It is a simple way to find feasible bases of the LP relaxation, and it has been successfully used as one of several main components to approximate the first split closure [42].

The method consists in computing a round of GMIs from the optimal tableau, then adding them as Lagrangean penalties to the objective function instead of as additional constraints to the feasible region. When re-optimizing over the LP relaxation with this modified objective function, one may obtain a different basis. The process can be iterated in order to find more bases. We refer to [43] for a more detailed exposition of the method.

Table 5.18 shows the results for two-row intersection cuts. From one basis, GMIs close 23.084% of gap on average, and two-row cuts close 29.589%. When using relax-and-cut, we generate more GMIs and two-row cuts, as we compute them from up to 21 different bases. Then, GMIs close 36.189% and two-row cuts close 42.680%. Similar results hold for cuts from the full two-row model (see Table 5.19). From one basis, GMIs close 23.282% of gap on average, and strengthened two-row cuts close 36.080%. From up to 21 different bases, GMIs close 34.970% and strengthened two-row cuts close 47.277%.

Contrary to our earlier suggestions, the strengthening provided by two-row cuts on top of GMIs seems to consistently carry over to when generating cuts from several bases, both in the case of intersection cuts and in the case of cuts from the full model.

	2-row intersection cuts from 1 basis						2-row intersection cuts from up to 21 basis					
name	GMIs	GMI %gc	bases	cuts (t)	%gc	ex.	GMIs	GMI %gc	bases	cuts (t)	%gc	ex.
bell3a	14	39.03	1	9	52.48	Y	28	54.17	21	24	59.92	Y
bell5	17	14.53	1	8	14.92	Y	23	34.66	21	40	35.63	Y
blend2	5	16.04	1	3	16.90	Y	14	20.70	21	7	23.48	Y
cap6000	2	39.91	1	6	40.07	Y	24	45.07	21	12	45.26	Y
danoint	31	0.26	1	39	1.74	-	70	0.63	21	22	1.74	-
dcmulti	36	47.80	1	44	49.92	Y	77	64.28	21	74	68.10	Y
egout	8	31.94	1	8	60.54	Y	20	50.27	21	16	95.28	Y
fiber	22	69.30	1	26	71.08	Y	43	77.66	21	40	78.00	Y
fixnet6	11	22.26	1	18	30.64	Y	29	32.98	21	42	47.46	Y
flugpl	7	10.75	1	7	13.02	Y	8	11.30	21	7	13.11	Y
gen	6	1.27	1	13	9.53	-	28	40.11	21	30	40.28	Y
gesa2	40	27.56	1	91	56.48	-	104	35.24	21	198	37.63	-
gesa2_o	70	30.68	1	136	32.87	Y	162	40.45	21	249	58.51	-
gesa3	37	20.48	1	142	47.06	Y	55	33.70	21	215	49.69	-
gesa3_o	64	50.54	1	146	69.36	Y	118	54.70	21	422	69.81	-
gt2	11	47.22	1	8	47.22	Y	32	58.12	21	27	58.17	Y
harp2	22	22.81	1	20	25.20	-	101	31.30	21	36	32.72	-
khb05250	19	73.16	1	15	73.16	Y	40	86.93	21	40	90.94	Y
l152lav	7	2.01	1	14	2.04	-	38	16.65	21	17	16.97	Y
lseu	5	20.48	1	8	21.01	Y	39	40.92	21	20	40.92	Y
mas74	11	6.67	1	7	6.67	Y	28	6.98	21	8	6.98	Y
mas76	10	6.42	1	6	6.42	Y	53	6.67	21	6	6.67	Y
misc03	4	8.62	1	7	8.62	Y	17	17.59	21	25	17.59	Y
misc06	16	28.48	1	17	48.27	-	24	44.72	21	42	88.62	-
misc07	5	0.72	1	9	0.72	Y	14	0.72	21	13	0.72	Y
mitre	101	50.71	1	476	81.71	-	315	83.55	21	950	83.55	Y
mkc	34	0.96	1	89	2.64	Y	127	30.36	21	98	34.30	-
mod008	5	21.62	1	1	21.62	Y	33	35.89	21	7	35.93	Y
mod011	21	30.69	1	41	36.06	Y	158	40.70	21	130	46.76	-
modglob	28	17.28	1	29	26.13	Y	91	50.63	21	70	59.21	Y
nw04	2	29.75	1	2	29.75	Y	7	66.08	21	16	66.08	Y
p0033	4	34.42	1	5	34.81	Y	16	53.76	21	9	53.76	Y
p0201	14	0.39	1	6	6.52	Y	52	13.88	21	175	18.83	Y
p0282	23	3.19	1	22	4.63	Y	77	9.93	21	31	12.42	Y
p0548	31	60.67	1	33	60.72	Y	79	82.92	21	87	83.46	Y
p2756	80	56.96	1	80	58.33	Y	147	95.62	21	144	95.85	Y
pp08a	53	51.44	1	78	75.18	Y	72	59.67	2	96	85.19	Y
pp08acuts	41	31.52	1	40	38.78	Y	41	31.52	2	42	39.57	Y
qiu	25	1.69	1	22	1.87	-	126	6.12	21	67	6.42	Y
qnet1	19	7.18	1	18	14.28	Y	112	31.78	21	36	32.46	Y
qnet1_o	10	26.21	1	15	31.49	Y	46	45.76	21	53	54.87	Y
rentacar	13	4.97	1	11	5.45	-	16	5.09	21	16	5.51	-
rgn	12	5.02	1	8	5.02	Y	88	35.32	21	30	38.30	Y
rout	30	1.40	1	8	1.51	Y	60	5.97	21	20	7.82	Y
set1ch	121	28.20	1	126	60.63	Y	155	29.02	21	154	60.16	Y
seymour	57	8.05	1	11	8.43	Y	159	9.43	21	55	9.43	Y
swath	13	6.37	1	40	6.65	Y	77	20.88	13	59	20.88	-
vpm1	15	6.18	1	18	14.56	Y	32	9.06	21	42	23.22	Y
vpm2	21	7.31	1	22	17.15	Y	58	13.82	21	55	33.15	-
average	25.571	23.084	1.000	40.980	29.589	82%	68.020	36.189	20.061	83.143	42.680	76%

Table 5.18: Relax and cut: two-row intersection cuts

5.11 More than two rows

We have focused on two-row cuts so far because we have comparison points for them that are better understood than in the general multi-row case. Our separator however is not limited to two-row cuts, and we present in this section results with more than two rows.

To simplify the presentation, we only cover the k-row extension of the P_{full} model. Tables 5.20, 5.21, 5.22 and 5.23 show all the results for 1-, 2-, 3-, 4-, 5-, 6-, 7-, 10- and 15-row cuts.

We first focus on the speed of our separator. The computations were performed on 55 instances, as we further removed from MIPLIB 3 the instances for which one round of GMIs closes 100% gap, and the instances for which two-row cuts from all the pairs of rows do not close any more gap than GMIs.

	P_{full} from 1 basis						P_{full} from up to 21 basis					
name	GMIs	GMI %gc	bases	cuts (t)	%gc	ex.	GMIs	GMI %gc	bases	cuts (t)	%gc	ex.
`bell3a`	14	39.03	1	18	55.69	Y	28	54.17	21	24	59.92	Y
`bell5`	17	14.53	1	13	19.32	Y	23	34.66	21	19	50.33	Y
`blend2`	5	16.04	1	10	20.00	Y	14	20.70	21	22	26.63	Y
`cap6000`	2	39.91	1	2	39.91	_	24	45.07	21	8	45.07	_
`danoint`	31	0.26	1	58	1.74	_	62	0.64	21	46	1.74	_
`dcmulti`	36	47.80	1	104	58.05	Y	77	64.28	21	124	78.26	_
`egout`	8	31.94	1	16	62.24	Y	20	50.27	21	38	98.04	Y
`fiber`	22	69.30	1	262	82.39	Y	43	77.66	21	74	79.12	_
`fixnet6`	11	22.26	1	54	36.54	_	29	32.98	21	45	42.55	_
`flugpl`	7	10.75	1	6	43.38	Y	8	11.30	21	8	71.68	Y
`gesa2_o`	70	30.68	1	227	36.08	Y	162	40.45	21	285	47.60	_
`gesa3`	37	20.48	1	178	56.11	_	55	33.70	21	239	70.23	_
`gesa3_o`	64	50.54	1	224	77.74	_	118	54.70	21	472	84.01	_
`gt2`	11	47.22	1	58	58.37	_	28	58.12	21	46	60.14	_
`harp2`	22	22.81	1	20	22.81	_	101	31.30	21	30	31.30	_
`khb05250`	19	73.16	1	132	91.43	Y	40	86.93	21	96	95.52	_
`l152lav`	7	2.01	1	5	3.06	_	38	16.65	21	20	16.98	_
`mas74`	11	6.67	1	22	11.53	_	28	6.98	21	8	6.98	_
`mas76`	10	6.42	1	6	6.42	_	53	6.67	21	6	6.67	_
`misc03`	4	8.62	1	9	8.62	Y	17	17.59	21	56	17.59	Y
`misc06`	16	28.48	1	22	48.81	Y	24	44.72	21	63	91.67	_
`misc07`	5	0.72	1	13	0.72	Y	14	0.72	21	28	0.72	_
`mkc`	34	0.96	1	53	24.39	_	127	30.36	21	84	42.60	_
`mod008`	5	21.62	1	1	21.62	_	33	35.89	21	10	36.21	_
`modglob`	28	17.28	1	42	30.99	_	91	50.63	21	143	70.39	_
`nw04`	2	29.75	1	4	29.90	_	20	66.08	21	13	68.26	_
`p0033`	4	34.42	1	32	71.45	Y	16	53.76	21	86	100.00	Y
`p0201`	14	0.39	1	24	15.73	Y	52	13.88	21	53	15.18	_
`p0282`	23	3.19	1	194	41.51	Y	77	9.93	21	37	15.32	_
`p0548`	31	60.67	1	96	99.88	_	79	82.92	21	205	100.00	Y
`p2756`	80	56.96	1	323	77.21	Y	145	95.62	21	164	96.90	_
`pp08a`	53	51.44	1	84	75.18	Y	71	58.13	2	104	82.72	Y
`pp08acuts`	41	31.52	1	44	44.13	_	41	31.52	2	52	43.49	_
`qiu`	23	1.70	1	29	1.71	_	132	7.94	21	40	8.01	_
`qnet1`	17	15.76	1	6	15.76	_	112	31.78	21	32	31.84	_
`qnet1_o`	10	26.43	1	12	26.97	_	46	45.76	21	27	45.77	_
`rentacar`	13	4.97	1	12	5.25	_	16	5.09	21	14	5.09	_
`rout`	30	1.40	1	8	1.40	_	83	9.39	21	21	9.42	_
`set1ch`	121	28.20	1	143	60.75	Y	155	29.02	21	168	60.43	_
`seymour`	60	4.98	1	24	9.12	_	128	12.01	21	30	12.18	_
`swath`	13	6.37	1	17	12.21	_	77	20.88	13	51	32.07	_
`vpm1`	15	6.18	1	26	15.95	Y	31	9.06	21	94	25.61	Y
`vpm2`	21	7.31	1	39	29.38	Y	58	13.82	21	116	48.68	Y
average	24.814	23.282	1.000	62.140	36.080	47%	60.372	34.970	19.930	76.767	47.277	26%

Table 5.19: Relax and cut: full two-row model

Figure 5.3 shows the number of instances for which a result is reported (i.e. no memory exhaustion or dangerous numerical instability occurs), and the number of instances for which the separation is exact. The latter number quickly drops when going from 1- to 5-row cuts, but then stays around 10 from 5- through 15-row cuts. Figure 5.4 shows the geometric mean of the running times over the 42 instances for which a result was reported in every test (i.e. in the 1-row through to the 15-row test). Computing times indeed increase with the number of rows, but up to $k = 15$, we do not see yet the exponential growth of the complexity that is bound to occur for larger values of k (since the problem of separating over mixed-integer sets is $\mathcal{NP}$-hard).

On the same 42 instances, the average of the percentage of gap closed by the k-row cuts is plotted in Figure 5.5. Beyond $k = 5$, that value does not seem to reach above 38%, indicating that there would be limited interest in separating k-row cuts with $5 \leq k \leq 15$. Of course, this could be due to our separator being unable to separate as many cuts for the models with more rows, but Figure 5.6, considering only the few instances with exact separation across all tests, provides clues that this may not be the case.

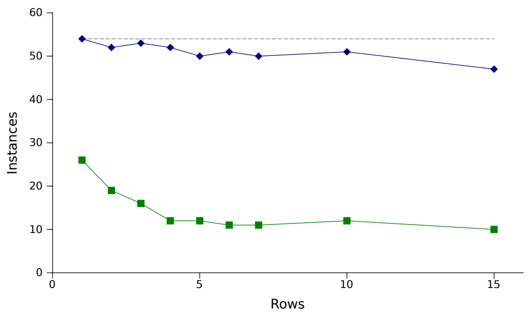

Figure 5.3: Number of instances ($\Diamond$) with a result and ($\Box$) with exact separation

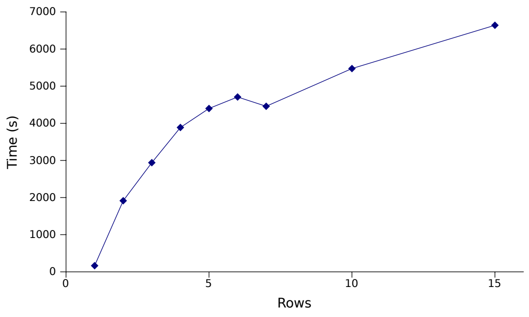

Figure 5.4: Geometric mean of the running time on 42 instances

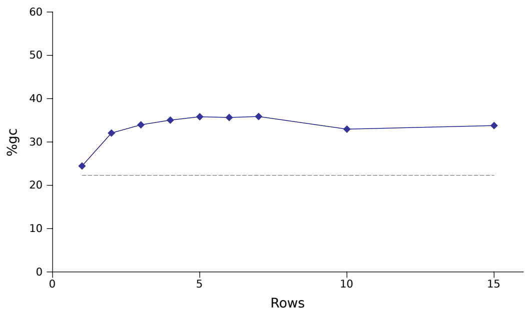

Figure 5.5: Average percentage of gap closed on 42 instances

Figure 5.7 on the other hand underlines a phenomenon that has important implications in practice. It displays, for each value of k, the average number of cuts generated by our separator and the number among them that are tight at x^* at the end of Algorithm 6. Recall that the number of k-row models considered is at most $\frac{m}{2}$ independently of k. The number of cuts that are tight at the end stays around 50 in average and varies very little for the different values of k. Meanwhile, the number of cuts that had to be generated raises steadily with increasing k. This means that when generating multi-row cuts with more rows, one needs to compute many more cuts, indicating that the complexity of the facial structure of multi-row models may raise one more hurdle for the use of multi-row cuts with many rows.

			1-row cuts				2-row cuts				3-row cuts			
name	GMIs	%gc	time	cuts(t)	%gc	ex.	time	cuts(t)	%gc	ex.	time	cuts (t)	%gc	ex.
10teams	101	57.14	14450.74	4	57.14	_	14425.15	4	57.14	_	14417.16	4	57.14	_
air04	100	7.90	1064.99	13	7.90	_	1270.79	13	7.90	_	2918.35	13	7.90	_
air05	101	4.63	3538.77	8	4.63	_	14400.01	8	4.63	_	14401.00	8	4.63	_
arki001	7	0.00	14400.54	10	0.99	_	14400.23	11	0.00	_	14402.71	11	0.00	_
bell3a	14	39.03	0.19	8	39.35	Y	38.09	17	55.69	Y	100.28	21	54.90	_
bell5	17	14.53	24.73	30	18.71	Y	87.47	15	19.32	Y	68.42	17	20.50	Y
blend2	5	16.04	7.61	3	16.47	_	2213.82	9	20.00	Y	10416.10	21	20.34	Y
cap6000	2	39.91	14400.30	2	39.91	_	14400.45	2	39.91	_	14400.02	2	39.91	_
danoint	32	0.26	35.00	5	0.26	_	14400.39	5	0.26	_	14400.16	5	0.26	_
dcmulti	36	47.80	7107.89	43	50.90	Y	14400.12	55	53.22	_	14400.02	47	52.22	_
egout	8	31.94	0.05	9	36.39	Y	0.40	14	62.24	Y	0.64	16	84.97	Y
fast0507	101	1.64	14401.08	8	1.64	_	14400.09	8	1.64	_	14400.42	8	1.64	_
fiber	22	69.20	43.22	62	73.24	Y	617.58	198	81.34	Y	1418.19	194	81.34	Y
fixnet6	11	22.26	9798.18	11	22.78	_	14400.08	14	22.40	_	14400.89	10	22.33	_
flugpl	7	10.75	0.05	4	10.75	Y	1.65	6	43.38	Y	5.66	7	87.91	Y
gen	6	1.27	0.33	6	1.27	Y	2868.59	40	50.81	_	14400.00	95	82.58	_
gesa2_o	70	30.68	54.39	94	33.07	Y	997.61	195	36.08	Y	2239.72	222	50.61	_
gesa3	37	20.48	81.42	91	21.90	Y	14400.02	107	42.79	_	14400.01	172	35.91	_
gesa3_o	64	50.54	438.35	123	50.73	_	14400.03	246	69.68	_	14400.15	185	63.68	_
gt2	11	47.22	3163.57	20	56.78	_	14400.16	39	58.05	_	14400.29	39	57.61	_
harp2	22	22.81	14400.41	20	22.81	_	14403.60	20	22.81	_	14401.06	20	22.81	_
khb05250	19	73.16	14400.07	34	82.12	_	14400.28	26	80.28	_	14400.76	27	80.54	_
l152lav	7	2.01	14400.21	10	5.88	_	14400.03	6	3.14	_	14400.55	5	3.06	_
lseu	5	20.48	67.26	52	23.37	_	416.03	18	67.62	Y	881.00	24	69.80	_
mas74	11	6.67	835.86	24	8.85	_	3055.31	8	7.04	_	1271.02	7	6.67	_
mas76	10	6.42	561.06	26	8.30	_	1184.28	6	6.42	_	1119.62	6	6.42	_
misc03	4	8.62	0.06	7	8.62	Y	58.21	9	8.62	Y	84.07	11	8.62	Y
misc06	16	28.48	0.17	13	28.48	Y	14401.48	17	48.81	_	14400.11	19	48.81	_
misc07	5	0.72	0.17	9	0.72	Y	4.31	13	0.72	Y	29.56	12	0.72	Y
mitre	101	50.71	14407.23	567	76.47	_	14410.88	657	73.61	_	14423.54	602	67.33	_
mkc	34	0.96	14400.59	48	2.22	_	14400.99	44	0.97	_	14400.38	51	1.04	_
mod008	5	21.62	206.97	5	21.85	Y	87.25	1	21.62	_	1203.54	1	21.62	_
mod011	21	30.69	14432.43	16	30.69	_	14425.86	16	30.69	_	14413.75	16	30.69	_
modglob	28	17.28	0.86	18	18.06	Y	2844.76	36	30.98	_	13484.47	35	33.47	_
nw04	2	29.75	14400.45	3	30.66	_	14400.67	4	29.90	_	14400.25	3	29.81	_
p0033	4	34.42	2.98	19	60.03	Y	3.33	34	71.45	Y	3.50	40	100.00	Y
p0201	14	0.39	0.54	18	0.39	Y	154.75	30	15.73	Y	224.28	21	16.00	Y
p0282	23	3.19	121.07	54	14.67	Y	2824.43	225	41.51	Y	1108.30	118	45.70	_
p0548	31	60.67	4.60	81	97.17	Y	9.83	96	99.88	Y	21.01	91	99.88	Y
pp08a	53	51.44	0.20	46	51.44	Y	3.11	83	75.18	Y	9.96	106	80.42	Y
pp08acuts	41	31.52	44.59	30	31.94	_	2531.02	42	44.12	_	2793.51	59	52.97	_
qiu	23	1.70	14402.38	15	1.70	_	14400.07	15	1.70	_	14400.77	15	1.70	_
qnet1	17	15.76	14409.04	10	16.00	_	14400.69	10	15.90	_	14401.61	9	15.90	_
qnet1_o	10	26.43	2257.79	91	31.53	_	14402.04	24	28.53	_	14400.76	20	27.07	_
rentacar	13	4.97	14401.84	8	4.97	_	14400.84	7	11.52	_	14404.37	7	14.98	_
rgn	12	5.02	16.49	11	10.00	Y	25.70	34	25.10	Y	569.58	38	25.10	Y
rout	30	1.40	14404.44	13	1.82	_	14400.06	7	1.40	_	14400.56	7	1.40	_
set1ch	121	28.20	1.11	120	28.20	Y	25.18	143	60.75	Y	94.96	196	82.52	Y
seymour	60	4.98	3011.23	16	5.27	Y	14401.19	18	4.98	_	14401.94	18	4.98	_
swath	13	6.37	383.46	24	11.39	_	14401.00	17	12.21	_	14400.86	17	12.21	_
vpm1	15	6.18	0.15	10	6.54	Y	5.00	24	15.95	Y	18.89	32	21.59	Y
vpm2	21	7.31	0.92	20	8.11	Y	13.76	39	29.38	Y	69.56	53	32.54	Y
avg.	29.673	21.607	4788.231	38.308	24.905	46%	7611.898	52.596	32.981	37%	8250.544	53.519	36.399	29%
geom.	17.908	0.000	124.481	18.024	12.747	-	1184.855	21.008	0.000	-	1852.224	22.065	0.000	-

Table 5.20: Cuts from full multi-row models, part 1

			4-row cuts				5-row cuts			
name	GMIs	GMI %gc	time	cuts (t)	%gc	proof	time	cuts (t)	%gc	proof
`10teams`	101	57.14	14419.40	4	57.14	_	14403.86	4	57.14	_
`air04`	100	7.90	14400.98	13	7.90	_	14400.98	13	7.90	_
`air05`	101	4.63	14400.99	8	4.63	_	14400.32	8	4.63	_
`arki001`	7	0.00	14401.09	9	0.00	_	14407.15	11	0.00	_
`bell3a`	14	39.03	133.09	22	54.90	_	130.74	24	56.82	_
`bell5`	17	14.53	101.52	15	20.50	Y	147.44	15	20.50	Y
`blend2`	5	16.04	14400.00	8	16.96	_	14400.23	23	20.78	_
`cap6000`	2	39.91	14400.89	2	39.91	_	14400.90	2	39.91	_
`danoint`	32	0.26	14400.07	5	0.26	_	14400.04	5	0.26	_
`dcmulti`	36	47.80	14400.01	33	50.87	_	14400.24	34	49.71	_
`egout`	8	31.94	1.22	18	98.36	Y	0.93	23	100.00	Y
`fast0507`	101	1.64	14401.08	8	1.64	_	14401.03	8	1.64	_
`fiber`	22	69.20	14400.02	153	81.25	_	14400.13	189	81.30	_
`fixnet6`	11	22.26	14400.34	10	22.31	_	14400.17	9	22.31	_
`flugpl`	7	10.75	5.72	10	99.16	Y	5.33	11	100.00	Y
`gen`	6	1.27	14400.02	80	85.16	_	14400.00	59	85.16	_
`gesa3`	37	20.48	14400.01	164	36.13	_	14400.01	131	36.01	_
`gesa3_o`	64	50.54	14400.01	156	61.16	_	14400.08	106	60.76	_
`gt2`	11	47.22	14400.28	58	56.45	_	14400.26	38	58.34	_
`harp2`	22	22.81	14400.72	20	22.81	_	14400.05	20	22.81	_
`khb05250`	19	73.16	14400.27	27	79.43	_	14400.32	23	77.04	_
`l152lav`	7	2.01	14400.05	8	2.96	_	14400.64	8	2.57	_
`lseu`	5	20.48	3962.76	50	82.13	Y	1850.72	103	82.13	Y
`mas74`	11	6.67	1672.66	7	6.67	_	1971.52	7	6.67	_
`mas76`	10	6.42	2241.36	6	6.42	_	1932.36	6	6.42	_
`misc03`	4	8.62	176.69	18	8.62	Y	56.63	8	8.62	Y
`misc06`	16	28.48	14401.57	17	47.64	_	14400.69	16	48.81	_
`misc07`	5	0.72	31.64	13	0.72	Y	298.76	21	0.72	Y
`mitre`	101	50.71	14404.67	652	83.11	_	14419.11	580	76.64	_
`mkc`	34	0.96	14400.13	53	1.04	_	14400.58	37	0.96	_
`mod008`	5	21.62	1965.00	1	21.62	_	5906.68	2	21.70	_
`mod011`	21	30.69	14412.28	16	30.69	_	14417.21	16	30.69	_
`modglob`	28	17.28	14449.54	65	32.97	_	14400.04	66	31.50	_
`nw04`	2	29.75	14400.60	2	29.75	_	14400.03	2	29.75	_
`p0033`	4	34.42	6.34	31	100.00	Y	6.39	21	100.00	Y
`p0201`	14	0.39	2072.63	49	17.83	Y	2232.62	35	17.83	Y
`p2756`	80	56.96	120.55	144	76.99	_	601.95	231	77.21	Y
`pp08a`	53	51.44	9.46	114	83.18	Y	20.04	113	83.47	Y
`pp08acuts`	41	31.52	5770.35	60	53.82	_	12352.69	93	54.15	_
`qiu`	23	1.70	14401.77	15	1.70	_	14400.65	16	1.70	_
`qnet1`	17	15.76	14409.23	7	15.76	_	14400.50	7	15.76	_
`qnet1_o`	10	26.43	14400.01	18	26.97	_	14400.42	14	26.96	_
`rentacar`	13	4.97	14400.06	7	4.97	_	15884.17	7	4.97	_
`rgn`	12	5.02	861.30	27	26.71	_	3465.41	34	27.06	_
`rout`	30	1.40	14400.28	6	1.40	_	14400.09	7	1.40	_
`set1ch`	121	28.20	181.18	200	89.11	Y	420.74	198	90.46	Y
`seymour`	60	4.98	14400.71	18	4.98	_	14400.56	18	4.98	_
`swath`	13	6.37	14400.13	16	6.37	_	14400.08	22	23.82	_
`vpm1`	15	6.18	15.96	34	26.79	Y	41.07	43	41.59	Y
`vpm2`	21	7.31	127.00	88	39.13	_	566.90	103	41.01	_
average	29.980	21.719	9319.273	51.300	36.540	22%	9598.989	51.800	37.251	24%
geometric	17.671	0.000	2574.734	20.977	0.000	-	3110.291	21.815	0.000	-

Table 5.21: Cuts from full multi-row models, part 2

			6-row cuts				7-row cuts			
name	GMIs	GMI %gc	time	cuts (t)	%gc	proof	time	cuts (t)	%gc	proof
`10teams`	101	57.14	14430.76	4	57.14	–	14444.12	4	57.14	–
`air04`	100	7.90	14400.92	13	7.90	–	14400.94	13	7.90	–
`air05`	101	4.63	14400.98	8	4.63	–	14400.99	8	4.63	–
`arki001`	7	0.00	14402.49	11	0.00	–	14402.08	10	0.00	–
`bell3a`	14	39.03	125.19	24	56.80	–	108.43	26	56.82	–
`bell5`	17	14.53	133.38	15	20.50	Y	176.68	16	20.50	Y
`blend2`	5	16.04	14400.01	18	21.33	–	14400.25	60	21.19	–
`cap6000`	2	39.91	14400.99	2	39.91	–	14400.07	2	39.91	–
`danoint`	32	0.26	14400.17	5	0.26	–	14401.17	5	0.26	–
`dcmulti`	36	47.80	14400.29	28	50.91	–	14400.09	35	52.87	–
`egout`	8	31.94	1.06	26	100.00	Y	1.07	40	100.00	Y
`fast0507`	101	1.64	14401.05	8	1.64	–	14401.07	8	1.64	–
`fiber`	22	69.20	14400.53	361	82.69	–	14400.06	256	82.11	–
`fixnet6`	11	22.26	14400.39	9	22.31	–	14400.25	9	22.29	–
`flugpl`	7	10.75	49.48	14	100.00	Y	5.85	11	100.00	Y
`gen`	6	1.27	14400.02	13	70.86	–	14400.05	15	62.64	–
`gesa2_o`	70	30.68	14400.01	208	67.97	–	14400.16	243	64.91	–
`gesa3`	37	20.48	14400.03	126	36.13	–	14400.02	89	36.01	–
`gesa3_o`	64	50.54	14400.03	98	56.27	–	14400.01	98	56.24	–
`gt2`	11	47.22	14400.13	16	52.21	–	14400.14	18	52.21	–
`harp2`	22	22.81	14404.23	20	22.81	–	14400.66	20	22.81	–
`khb05250`	19	73.16	14400.60	20	74.53	–	14400.94	18	74.53	–
`l152lav`	7	2.01	14400.94	7	2.76	–	14400.06	6	2.37	–
`lseu`	5	20.48	7822.41	91	85.81	Y	5507.89	84	85.81	–
`mas74`	11	6.67	1618.89	7	6.67	–	1691.01	7	6.67	–
`mas76`	10	6.42	4692.80	6	6.42	–	6304.31	6	6.42	–
`misc03`	4	8.62	59.45	11	8.62	Y	79.55	12	8.62	Y
`misc06`	16	28.48	14400.19	17	47.54	–	14401.30	17	47.62	–
`misc07`	5	0.72	530.71	21	0.72	–	209.12	50	0.72	Y
`mitre`	101	50.71	14407.72	565	84.26	–	2311.63	547	100.00	Y
`mkc`	34	0.96	14400.48	47	1.04	–	14400.20	49	1.65	–
`mod011`	21	30.69	14420.41	16	30.69	–	14437.35	16	30.69	–
`modglob`	28	17.28	14400.08	58	30.54	–	14400.01	62	31.65	–
`nw04`	2	29.75	14401.02	2	29.75	–	14400.32	2	29.75	–
`p0033`	4	34.42	4.59	27	100.00	Y	5.21	23	100.00	Y
`p0201`	14	0.39	1538.32	53	17.83	Y	4333.38	40	17.83	Y
`p2756`	80	56.96	628.48	275	77.21	Y	670.78	267	77.21	–
`pp08a`	53	51.44	22.16	117	83.18	Y	26.62	119	83.47	Y
`pp08acuts`	41	31.52	6794.67	75	54.62	–	5601.99	95	55.20	–
`qiu`	23	1.70	14400.75	15	1.70	–	14402.36	14	1.70	–
`qnet1`	17	15.76	14405.61	7	15.76	–	14400.75	7	15.76	–
`qnet1_o`	10	26.43	14400.23	12	26.91	–	14400.73	9	26.88	–
`rentacar`	13	4.97	14402.10	7	4.97	–	14404.95	7	4.97	–
`rgn`	12	5.02	2486.13	65	30.45	Y	5048.14	34	30.76	–
`rout`	30	1.40	14400.12	7	1.40	–	14402.88	7	1.40	–
`set1ch`	121	28.20	218.59	188	92.19	Y	355.31	184	92.19	Y
`seymour`	60	4.98	14400.09	18	4.98	–	14402.19	18	4.98	–
`swath`	13	6.37	14401.38	21	14.01	–	14401.64	17	12.21	–
`vpm1`	15	6.18	586.93	56	54.05	–	92.93	56	54.05	Y
average	31.490	22.198	9963.224	57.918	37.977	22%	9776.280	56.306	38.106	22%
geometric	18.583	0.000	3660.620	22.952	0.000	-	3380.613	23.248	0.000	-

Table 5.22: Cuts from full multi-row models, part 3

			10-row cuts				15-row cuts			
name	GMIs	GMI %gc	time	cuts (t)	%gc	proof	time	cuts (t)	%gc	proof
10teams	101	57.14	14432.84	4	57.14	–	14434.87	4	57.14	–
air04	100	7.90	14401.00	13	7.90	–	14400.95	13	7.90	–
air05	101	4.63	14400.92	8	4.63	–	14400.86	8	4.63	–
arki001	7	0.00	14403.64	10	0.00	–	14401.99	7	0.00	–
bell3a	14	39.03	268.31	26	56.89	–	154.26	23	58.67	Y
bell5	17	14.53	219.30	16	20.50	Y	585.25	16	21.68	Y
blend2	5	16.04	14400.03	27	21.24	–	14400.04	41	21.43	–
cap6000	2	39.91	14401.01	2	39.91	–	14402.61	2	39.91	–
danoint	32	0.26	14400.60	5	0.26	–	14400.38	5	0.26	–
dcmulti	36	47.80	14400.02	32	49.77	–	14400.15	33	50.79	–
egout	8	31.94	1.96	32	100.00	Y	4.04	29	100.00	Y
fast0507	101	1.64	14401.07	8	1.64	–	14401.07	8	1.64	–
fiber	22	69.20	14400.05	108	82.25	–	14400.98	115	81.58	–
fixnet6	11	22.26	14400.42	9	22.31	–	14400.18	9	22.31	–
gen	6	1.27	14400.07	19	10.33	–	14400.01	25	73.17	–
gesa2_o	70	30.68	14400.02	221	67.62	–	14400.04	164	62.50	–
gesa3	37	20.48	14400.03	72	36.02	–	14400.04	63	33.90	–
gesa3_o	64	50.54	14400.09	87	55.58	–	14400.05	65	52.05	–
gt2	11	47.22	14400.33	14	47.68	–	3262.71	7	47.22	–
harp2	22	22.81	14400.95	20	22.81	–	14400.22	20	22.81	–
khb05250	19	73.16	14406.43	15	73.16	–	14401.18	15	73.16	–
l152lav	7	2.01	14400.02	6	2.31	–	14400.04	6	2.46	–
lseu	5	20.48	3999.64	9	20.71	–	2034.85	6	20.48	–
misc03	4	8.62	1625.14	25	9.08	Y	6581.97	142	10.49	Y
misc06	16	28.48	14400.06	17	39.55	–	14401.17	14	28.48	–
misc07	5	0.72	527.93	49	0.72	Y	3548.89	109	0.80	Y
mitre	101	50.71	2681.02	508	100.00	Y	14400.91	444	79.03	–
mkc	34	0.96	14400.15	49	1.65	–	14400.05	52	1.65	–
mod011	21	30.69	14443.40	16	30.69	–	14416.87	16	30.69	–
modglob	28	17.28	14400.60	65	24.81	–	14400.02	25	18.66	–
nw04	2	29.75	14400.99	2	29.75	–	14400.95	2	29.75	–
p0201	14	0.39	9024.90	37	17.83	Y	14400.13	34	13.48	–
p0282	23	3.19	14400.11	122	97.57	–	14400.03	120	97.08	–
p0548	31	60.67	586.90	116	99.95	Y	4732.70	102	100.00	Y
p2756	80	56.96	689.73	324	77.21	Y	964.46	258	77.21	Y
pp08a	53	51.44	49.08	118	83.48	Y	550.81	152	84.56	–
pp08acuts	41	31.52	11938.10	140	56.50	–	14400.08	134	61.64	–
qiu	23	1.70	14404.50	14	1.70	–	14400.83	14	1.70	–
qnet1	17	15.76	14401.00	7	15.76	–	14401.52	7	15.76	–
qnet1_o	10	26.43	14400.83	9	26.89	–	14400.01	9	26.86	–
rgn	12	5.02	861.45	82	33.57	Y	4183.80	71	42.01	Y
rout	30	1.40	14400.99	6	1.40	–	14400.61	6	1.40	–
set1ch	121	28.20	785.66	189	92.19	Y	1075.44	186	92.19	Y
seymour	60	4.98	14400.36	22	7.09	–	14400.24	18	4.98	–
swath	13	6.37	14401.36	20	23.82	–	14400.01	20	23.82	–
vpm1	15	6.18	684.69	80	54.94	–	297.65	69	57.94	Y
average	33.739	23.660	10444.515	60.435	37.539	24%	10940.129	58.435	38.171	22%
geometric	20.600	0.000	5199.588	26.259	0.000	-	6537.140	25.662	0.000	-

Table 5.23: Cuts from full multi-row models, part 4

name	GMI %gc	1-row %gc	2-row %gc	3-row %gc	4-row %gc	5-row %gc	6-row %gc	7-row %gc	10-row %gc	15-row %gc
bell5	14.53	18.71	19.32	20.50	20.50	20.50	20.50	20.50	20.50	21.68
egout	31.94	36.39	62.24	84.97	98.36	100.00	100.00	100.00	100.00	100.00
misc03	8.62	8.62	8.62	8.62	8.62	8.62	8.62	8.62	9.08	10.49
set1ch	28.20	28.20	60.75	82.52	89.11	90.46	92.19	92.19	92.19	92.19
average	20.822	22.980	37.733	49.153	54.148	54.895	55.328	55.328	55.443	56.090

Table 5.24: Gap closed, 4 instances with exact separation

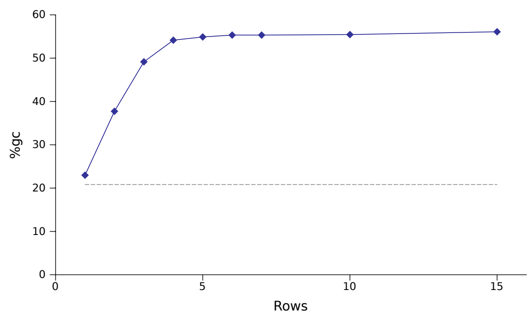

Figure 5.6: Average percentage of gap closed on 4 instances with exact separation

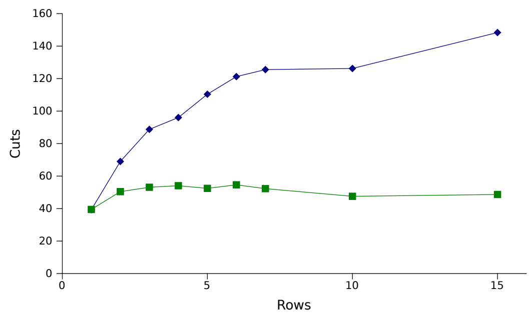

Figure 5.7: Number of cuts generated (◊) and tight at the end (□) on 42 instances

5.12 Summary

In this chapter, we implemented a separator for arbitrary mixed-integer sets. Computationally, the task is inherently costly, and a separator with such a generic scope is bound to be slow in practice. But our first naive implementation was unable to provide meaningful results, even for some of the smallest instances in MIPLIB 3. To partially mitigate the issue we developed a series of tricks, that are mainly based on the concept of lifting inequalities that are valid for projections of the feasible region. As a result, we obtain an improved version of our code that is over 14 times faster in geometric mean on 13 small benchmark instances (Table 5.8). More importantly, given a time limit of four hours per instance, we show that our improved code yields an exact value of the gap closure in 9 of the 13 instances, up from 5 in our initial implementation.

Using our separator, we show that on average over our testset, two-row intersection cuts close around 5% more gap than GMIs. Further, cuts from fully-strengthened two-row models close an additional 8% gap. We remark however that strengthening only partially the two-row models yields almost no improvement over intersection cuts. We then try generating GMIs and two-row cuts from several feasible bases of the LP relaxation, with surprisingly similar results. This leads us to conclude that the usefulness of two-row cuts, although limited, is not canceled by the effect of GMIs when considering cuts from several tableaux.

We then use our implementation to separate multi-row cuts with more than two rows. The running times show that the separator scales acceptably to up to 15 rows. On the other hand, the percentage of gap closure we obtain seems to tail off after 4- or 5-row cuts, indicating that there would be little interest in generating multi-row cuts with more than 5 rows (at least if still less than 15).

Chapter 6

Conclusions

6.1 Context of this thesis

The feasible region of mixed-integer linear programming problems (MIPs) is described as the intersection of a polyhedron (called the linear relaxation, and defined by a finite number of linear constraints), with integrality constraints on some of the variables. These problems are typically best solved using enumeration techniques such as the branch-and-bound method.

But three additional components have become essential to mixed-integer linear programming, and have driven the tremendous improvements that MIP solvers have known in the last 20 years:

1. preprocessing, that mainly reduces the size of the formulations and enhances their numerical stability,
2. primal heuristics, that provide fast methods for finding good feasible solutions, and
3. cutting planes, the subject of this thesis.

Cutting planes are simply linear constraints that are added to the formulation of a MIP, without affecting its feasible region. Their name comes from the fact that they cut part of the linear relaxation polyhedron.

The first cutting planes methods were developed by Gomory in 1960 [46, 47, 48]. He saw them as a stand-alone way of solving MIPs, by iteratively strengthening the linear relaxation until it describes the convex hull of the mixed-integer solutions. Computationally, these methods were plagued with numerical instabilities, and were widely considered impractical in the MIP research community. But in 1993, Balas, Ceria and Cornuéjols [9] discovered that adding cuts to strengthen a linear relaxation prior to performing branch-and-bound made the latter much faster in many cases.

Since then, many different types of cutting planes have been studied, and a dozen of them have made their way into MIP resolution codes. Among the most successful are one of Gomory's initial proposals, known as Gomory's mixed integer cut (GMI [47]), and a variant due to Nemhauser and Wolsey, called the mixed-integer rounding cut (MIR [67]). Most of these cuts share one property: they can be derived from one linear constraint that is valid for the linear relaxation.

In 2007, a paper by Andersen, Louveaux, Weismantel and Wolsey [4] sparked a wide interest among researchers looking for new ways of generating cuts. The paper exploits the concept of intersection cut developed by Balas in 1970 [7] and the concept of corner relaxation introduced by Gomory and Johnson [50, 51, 52]. It presents a simple setting where a single cut derived from two linear constraints of the linear relaxation is more effective than any number of cuts derived from a single linear constraint. These specific types of intersection cuts are since commonly referred to as multi-row cuts.

This thesis centers on the computational aspects of multi-row cuts.

6.2 Content of this thesis

Throughout this thesis, we have adopted the intersection cut point-of-view on multi-row cuts, and we studied ways to exploit and evaluate computationally this framework.

In Chapter 3, we devise a method for the separation of two-row intersection cuts. In the process, we propose a compact formulation for the polar set of the integer hull of the intersection cut model. We also describe a row-generation algorithm for optimizing over that polar set, including a closed-form oracle for finding violated rows of the description of the polar. We show that our implementation of this method is fast and works in practice. Moreover, our two-row cuts separator is exact, in the sense that it is able to find a violated facet-defining inequality whenever one exists. Therefore, it permits an accurate estimation of the strength of the two-row intersection cut model itself. In particular, our results indicate that this model may be too weak to generate useful cuts. Hence, in Chapter 4, we review the various ways that have been proposed to strengthen the intersection cut model, and perform a first rough experiment to evaluate their respective value. In Chapter 5, in need for a more precise assessment, we set upon the task of developing a generic, exact separator for arbitrary mixed-integer sets. We use this new tool to refine our computational understanding of the various models involved.

All our results converge towards a few conclusions. Gomory's mixed-integer cuts are a formidable asset in the arsenal of mixed-integer linear programming solvers. Besides GMIs, our computational data indicates that there is a room for the use of two-row cuts in practice, if strengthened to the fullest extent possible. However, a lot of work remains to be done before the effective separation of such strengthened two-row cuts becomes tractable (we can of course separate them with our generic separator, but not fast enough for practical purposes). Mainly, the works in Chapter 3 should be extended to the S-free case, and more explorations are necessary towards a generic implementation for solving the two-row lifting problem. Even then, while two-row cuts may have one day their place among the few other families of cutting that are exploited in general-purpose MIP solvers, we show that they will probably not constitute the revolution that the incorporation of GMIs represented for the resolution of MIPs. Nor would, according to our data, intersection cuts from few simplex tableau rows.

6.3 Further research

Two important questions arise from the conclusions of this thesis. The first is how to best select models to separate cuts from. We proposed various heuristics for selecting rows from a simplex tableau in Chapter 3 and Chapter 5. We also considered tableaux corresponding to different feasible bases in Chapter 5. But one can imagine other heuristics, and more importantly more bases. In particular, cuts from infeasible bases have proved very successful in the context of lift-and-project [11]. Furthermore, the models do not necessarily need to come from tableaux. Even taking linear combinations of tableau rows may already yield useful results, as MIR cuts have shown [67].

The second question is how to discriminate between the MIP problems where multi-row cuts may be useful and the ones where they may not. Our results (e.g. Tables 3.4, 3.5, 5.11 and 5.19) show a huge variance of the impact of multi-row cuts across different instances. The structure of the instances probably plays a role in this phenomenon, but we are not yet able to characterize this structure so as to understand the links with the efficacy of multi-row cuts.

In the longer term, there are a number of promising alternative approaches for cutting planes generation. While we mentioned the notion of infinite relaxation in a few places, we never took that point-of-view. We always worked on models with few rows and required exact separation from the models. Instead, in the context of the infinite relaxation introduced by Gomory and Johnson [51], this requirement is dropped in favour of a weaker one: the generation of facet-defining inequalities for an infinite relaxation of the model. This trade-off enables the computation of the cuts in closed-form. We know that multi-row cuts from few rows only offer limited prospects, but it is conceivable that, adopting that point-of-view, we could generate cuts from very large models, maybe up to the size of the original MIP.

The recent work of Balas and Margot [12] generalizes the concept of intersection cuts to models that are more complex. An interesting feature of the technique developed is that the cuts are represented internally by their intersection points with the LP relaxation [13]. Such a representation may present very enticing properties from a computational perspective, as it may reduce the impact of numerical errors.

Further departing from classical approaches, Dey and Pokutta [36] lay the theoretical foundations of a new framework for cut generation. Their idea is to first design inequalities that have desirable properties but may be invalid, and then check their validity. A lot of practical questions remain open about how to generate such cuts computationally.

While expectations should not be overblown, we hope that this thesis brings indications that cutting planes have not yet reached their fullest potential, and provides a few pointers to promising directions.

Bibliography

[1] Tobias Achterberg, Thorsten Koch, and Alexander Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):361–372, 2006.

[2] Kent Andersen, Quentin Louveaux, and Robert Weismantel. Mixed-integer sets from two rows of two adjacent simplex bases. *Mathematical Programming*, 124:455–480, 2010.

[3] Kent Andersen, Quentin Louveaux, Robert Weismantel, and Laurence Wolsey. Cutting planes from two rows of a simplex tableau (extended version), 2006. Working Paper available on `http://orbi.ulg.ac.be/handle/2268/82794`.

[4] Kent Andersen, Quentin Louveaux, Robert Weismantel, and Laurence Wolsey. Inequalities from two rows of a simplex tableau. In Matteo Fischetti and David Williamson, editors, *Integer Programming and Combinatorial Optimization*, volume 4513 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin / Heidelberg, 2007.

[5] Krzysztof R. Apt. *Principles of constraint programming*. Cambridge University Press, 2003.

[6] Alper Atamtürk. `http://ieor.berkeley.edu/~atamturk/data/`.

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

[8] Egon Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(1–3):3 – 44, 1998.

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

[10] Egon Balas, S. Ceria, G. Cornuéjols, and N. Natraj. Gomory cuts revisited. *Operations Research Letters*, (19):1–9, 1996.

[11] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Program.*, 58(3):295–324, February 1993.

[12] Egon Balas and François Margot. Generalized intersection cuts and a new cut generating paradigm. *Mathematical Programming*, pages 1–17, 2011.

[13] Egon Balas, François Margot, and Selvaprabu Nadarajah. A polynomial procedure for generating generalized intersection cuts. MIP 2012 poster session, Davis, 2012.

[14] Egon Balas and Michael Perregaard. Lift-and-project for mixed 0–1 programming: recent progress. *Discrete Applied Mathematics*, 123(1–3):129 – 154, 2002.

[15] Egon Balas and Michael Perregaard. A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts for 0-1 programming. *Mathematical Programming*, 94(2-3):221–245, 2003.

[16] Alexander Barvinok. A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. *Mathematics of Operations Research*, 19(4):769–779, 1994.

[17] Alexander Barvinok. *Integer Points in Polyhedra.* Zurich Lectures in Advanced Mathematics. European Mathematical Society, 2008.

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

[19] Amitabh Basu, Pierre Bonami, Gérard Cornuéjols, and François Margot. On the relative strength of split, triangle and quadrilateral cuts. In *SODA*, pages 1220–1229. SIAM, 2009.

[20] Amitabh Basu, Michele Conforti, Gérard Cornuéjols, and Zambelli Giacomo. Minimal inequalities for an infinite relaxation of integer programs. *SIAM Journal on Discrete Mathematics 24*, pages 158–168, 2010.

[21] Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific Series in Optimization and Neural Computation. Athena Scientific, 1997.

[22] Dimitris Bertsimas and Robert Weismantel. *Optimization Over Integers.* Dynamic Ideas, 2005.

[23] Robert E. Bixby, Sebastián Ceria, Cassandra M. McZeal, and Martin W. P Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, (58):12–15, June 1998.

[24] Robert E. Bixby, Mary Fenelon, and Zonghao Gu. MIP: theory and practice – closing the gap. In *System Modelling and Optimization: Methods, Theory, and Applications.* Kluwer Academic Publishers, 2000.

[25] Valentin Borozan and Gérard Cornuéjols. Minimal valid inequalities for integer constraints. *Mathematics of Operations Research*, 34(3):538–546, 2009.

[26] Vašek Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4):305 – 337, 1973.

[27] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. A geometric perspective on lifting. *Oper. Res.*, 59:569–577, May 2011.

[28] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Corner polyhedron and intersection cuts. *Surveys in Operations Research and Management Science*, 16(2):105–120, 2011.

[29] Gérard Cornuéjols and François Margot. On the facets of mixed integer programs with two integer variables and two constraints. *Mathematical Programming*, 120(2):429–456, 2009.

[30] Gérard Cornuéjols. Revival of the gomory cuts in the 1990's. *Annals of Operations Research*, 149:63–66, 2007.

[31] Gérard Cornuéjols. Valid inequalities for mixed integer linear programs. *Mathematical Programming B*, 112:3–44, 2008.

[32] George B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In T.C. Koopmans, editor, *Activity Analysis of Production and Allocation*, page 339–347. Wiley, New York, 1951.

[33] Santanu Dey and Laurence Wolsey. Two row mixed-integer cuts via lifting. *Mathematical Programming*, 124:143–174, 2010.

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

[35] Santanu S. Dey and Quentin Louveaux. Split rank of triangle and quadrilateral inequalities. *Mathematics of Operations Research*, 36(3):432–461, 2011.

[36] Santanu S. Dey and Sebastian Pokutta. Design and verify: a new scheme for generating cutting-planes. In *Proceedings of the 15th international conference on Integer programming and combinatoral optimization*, IPCO'11, pages 143–155, Berlin, Heidelberg, 2011. Springer-Verlag.

[37] Santanu S. Dey and Laurence A. Wolsey. Lifting integer variables in minimal inequalities corresponding to lattice-free triangles. In Andrea Lodi, Alessandro Panconesi, and Giovanni Rinaldi, editors, *Integer Programming and Combinatorial Optimization, 13th International Conference, IPCO 2008, Bertinoro, Italy, May 26-28, 2008, Proceedings*, volume 5035 of *Lecture Notes in Computer Science*, pages 463–475. Springer, 2008.

[38] Santanu S. Dey and Laurence A. Wolsey. Constrained infinite group relaxations of MIPs. CORE Discussion Papers 2009033, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE), May 2009.

[39] Daniel G. Espinoza. Computing with multi-row Gomory cuts. *Operations Research Letters*, 38(2):115 – 120, 2010.

[40] Gyula Farkas. Theorie der einfachen Ungleichungen. *Journal für die Reine und Angewandte Mathematik*, (124):1–27, 1902.

[41] Matteo Fischetti and Michele Monaci. How tight is the corner relaxation? *Discrete Optimization*, 5(2):262 – 269, 2008. In Memory of George B. Dantzig.

[42] Matteo Fischetti and Domenico Salvagnin. Approximating the split closure. *INFORMS Journal on Computing*. To appear.

[43] Matteo Fischetti and Domenico Salvagnin. A relax-and-cut framework for gomory's mixed-integer cuts. In Andrea Lodi, Michela Milano, and Paolo Toth, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 6140 of *Lecture Notes in Computer Science*, pages 123–135. Springer Berlin Heidelberg, 2010.

[44] Ricardo Fukasawa and Marcos Goycoolea. On the exact separation of mixed integer knapsack cuts. *Mathematical Programming*, 128:19–41, 2011.

[45] Ricardo Fukasawa and Oktay Günlük. Strengthening lattice-free cuts using non-negativity. *Discrete Optimization*, 8(2):229 – 245, 2011.

[46] Ralph E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[47] Ralph E. Gomory. An algorithm for the mixed integer problem. Technical Report RM-2597, The Rand Corporation, 1960.

[48] Ralph E. Gomory. An algorithm for integer solutions to linear programs. In R.L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, New York, 1963.

[49] Ralph E. Gomory. On the relation between integer and noninteger solutions to linear programs. *Proceedings of the National Academy of Sciences*, 53:260–265, 1965.

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

[51] Ralph E. Gomory and Ellis L. Johnson. Some continuous functions related to corner polyhedra, part I. *Mathematical Programming*, 3:23–85, 1972.

[52] Ralph E. Gomory and Ellis L. Johnson. Some continuous functions related to corner polyhedra, part II. *Mathematical Programming*, 3:359–389, 1972.

[53] Warwick Harvey. Computing two-dimensional integer hulls. *SIAM J. Comput*, 28(6):2285–2299, 1999.

[54] Leonid V. Kantorovich. Mathematical methods in the organization and planning of production (in Russian). *Publication House of the Leningrad State University*, 1939. English Translation: Management Science, Volume 6 (1960), pp. 363–422.

[55] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, December 1984.

[56] Leonid G. Khachiyan. A polynomial algorithm in linear programming (in Russian). *Doklady Akademiia Nauk SSSR*, (224):1093–1096, 1979. English Translation: Soviet Mathematics Doklady, Volume 20, 191–194.

[57] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[58] Quentin Louveaux and Laurent Poirrier. A computational survey of best-case gap closure for various relaxations. MIP 2010 poster session, Atlanta, 2010.

[59] Quentin Louveaux and Laurent Poirrier. An algorithm for the separation of two-row cuts. *Mathematical Programming*, 143(1-2):111–146, 2014.

[60] László Lovász. Geometry of numbers and integer programming. *Proc. Math. Appl. Jap. Ser. 6*, pages 177–201, 1989.

[61] François Margot. MIPLIB3 C V2. `http://wpweb2.tepper.cmu.edu/fmargot/`, 2009.

[62] Kim Marriott and Peter J. Stuckey. *Programming with Constraints: An Introduction.* MIT Press, Cambridge, Massatchusetts, London, England, 1998.

[63] S. Thomas McCormick, Scott R. S Smallwood., and Frits C. R. Spieksma. A polynomial algorithm for multiprocessor scheduling with two job lengths. *Math. Oper. Res.*, 26(1):31–49, February 2001.

[64] Robert R. Meyer. On the existence of optimal solutions to integer and mixed-integer programming problems. *Mathematical Programming*, 7:223–235, 1974.

[65] Hermann Minkowski. *Geometrie des Zahlen.* Teubner, Leipzig, Germany, 1896.

[66] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, 1988.

[67] George L. Nemhauser and Laurence A. Wolsey. A recursive procedure to generate all cuts for 0–1 mixed integer programs. *Mathematical Programming*, 46:379–390, 1990.

[68] Yurii Nesterov. *Introductory Lectures on Convex Optimization. A Basic Course.* Kluwer, Boston, 2004.

[69] Michael Perregaard and Egon Balas. Generating cuts from multiple-term disjunctions. In Karen Aardal and Bert Gerards, editors, *Integer Programming and Combinatorial Optimization*, volume 2081 of *Lecture Notes in Computer Science*, pages 348–360. Springer Berlin / Heidelberg, 2001.

[70] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence).* Elsevier Science Inc., New York, NY, USA, 2006.

[71] Alexander Schrijver. *Theory of linear and integer programming.* Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1998.

[72] Michael J. Todd. The many facets of linear programming. *Mathematical Programming*, 91(3):417–436, February 2002.

[73] Franz Wesselmann. *Generating General-Purpose Cutting Planes for Mixed-Integer Programs.* PhD thesis, Universität Paderborn, 2010.

[74] Laurence Wolsey. *Integer Programming.* John Wiley & Sons, 1998.