Investigating Polyhedra by Oracles and Analyzing Simple Extensions of Polytopes

Dissertation
zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)

von Dipl.-Comp.-Math. Matthias Walter
geb. am 24. September 1986 in Wolmirstedt

genehmigt durch die Fakultät für Mathematik
der Otto-von-Guericke-Universität Magdeburg

Gutachter: Prof. Dr. Volker Kaibel
Otto-von-Guericke-Universität Magdeburg

Prof. Dr. Marc Pfetsch
Technische Universität Darmstadt

eingereicht am: 16. Dezember 2015
Verteidigung am: 29. März 2016

Dedicated to Sarah.

Zusammenfassung

Ein wichtiges Hilfsmittel der *Polyedrischen Kombinatorik* ist die computergestützte Analyse von Polyedern, um deren strukturelle Eigenschaften zu untersuchen. Aufgrund des enumerativen Charakters der klassischen Algorithmen sind diese Ansätze bereits bei verhältnismäßig kleinen Dimensionen nur noch mit enormen Ressourcen zu bewältigen.

Da oft die konvexen Hüllen der Lösungsmengen von Optimierungsproblemen von Interesse sind und es sehr effiziente Software (zum Beispiel im Falle gemischt-ganzzahliger linearer Optimierungsprobleme (MIPs)) zur Lösung dieser Probleme gibt, stellt sich die Frage, inwieweit man eine Analyse basierend auf solcher Software, oder formal einem *Optimierungsorakel*, durchführen kann.

Dafür wurde im Rahmen der Arbeit die Softwarebibliothek IPO (**I**nvestigating **P**olyhedra by **O**racles) entwickelt. Der erste Schritt einer Analyse ist in dem meisten Fällen die Bestimmung der affinen Hülle eines Polyeders $P \subseteq \mathbb{R}^n$, insbesondere der Dimension. Dazu wurde ein Algorithmus entwickelt, der mit $2n$ Orakel-Aufrufen auskommt und in der Praxis sehr effizient ist. Zudem wurde die Optimalität dieser Anzahl für den allgemeinen Fall bewiesen. Unter Nutzung einer entsprechenden Implementation und des MIP-Lösers SCIP wurden Instanzen der `MIPLIB 2.0` betrachtet und Dimensionen verschiedener Polyeder bestimmt:

- Lineare Relaxierung und gemischt-ganzzahlige Hüllen vor und nach Anwendung der Presolve-Routinen von SCIP.
- Optimale Seiten bezüglich der Zielfunktion der Instanz.
- Durch gegebene Ungleichungen induzierte Seiten.

Die Hauptanwendung von IPO ist die Bestimmung von Facetten von P, da diese zu nicht dominierten Ungleichungen in einer Ungleichungsbeschreibung korrespondieren und daher ein großes Potential besitzen, bei der Lösung (mit Hilfe von MIP-Lösern) von Nutzen zu sein. Facettendefinierende Ungleichungen, welche von einem gegebenen Punkt maximal verletzt werden, können mit Hilfe von *Target-Cuts* effektiv bestimmt werden. Dies wird zum Beispiel für Linearisierungen von Matching-Polytopen mit einem quadratischen Term demonstriert, indem mit Hilfe von IPO eine bisher unbekannte Klasse von Facetten entdeckt wurde.

Zusätzlich zur Facetten- und Affine-Hülle-Bestimmung hat IPO eine dritte Funktionalität, die die Überprüfung der Adjazenz zweier Ecken von P betrifft. Diese ist äquivalent dazu, dass der Normalenkegel an ihren Mittelpunkt $(n-1)$-dimensional ist. Da man mit Hilfe eines Optimierungsorakels für P – ähnlich zu Target Cuts – ein Optimierungsorakel für den betrachteten Normalenkegel konstruieren kann, reduziert sich das Adjazenzproblem ebenfalls auf das Affine-Hülle-Problem. Zur Demonstration wurden eine große Anzahl von Adjazenzen zufälliger Ecken von TSP-Polytopen überprüft, wobei als Orakel die TSP-Software concorde genutzt wurde.

Damit IPO grundsätzlich korrekt arbeitet, die Probleme jedoch möglichst effizient löst, wird an vielen Stellen sowohl rationale als auch Gleitkomma-Arithmetik verwendet. Dies ist besonders bei Dimensionen größer als 500 enorm relevant, da hier gelegentlich exakte Zahlen mit Kodierungslängen von über 10^6 Bits auftreten. Um weiterhin die meist laufzeitintensiven Orakelaufrufe zu sparen, erlaubt IPO die Nutzung von Heuristiken, von denen außer der Zulässigkeit der angegebenen Lösungen nichts verlangt wird. So können beispielsweise die gerundeten Lösungen von SCIP als heuristisch betrachtet und die exakte Version von SCIP zur "Verifikation" genutzt werden.

Der zweite Teil der Arbeit fällt in das Forschungsgebiet der *Erweiterten Formulierungen* und basiert auf einem gemeinsam mit Volker Kaibel veröffentlichten Artikel. In diesem Gebiet geht es im Wesentlichen darum, ein gegebenes Polytop P als affine Projektion eines anderen Polytops Q mit möglichst wenigen Facetten darzustellen. Die größte Motivation hierfür ist die (in gewissem Sinne äquivalente) Frage, ob man ein lineares Optimierungsproblem über P mit Hilfe von weiteren Variablen als ein lineares Optimierungsproblem darstellen kann, welches mit sehr wenigen Ungleichungen auskommt.

In der Arbeit wurde untersucht, inwieweit solche kompakten Darstellungen möglich sind, wenn man noch zusätzlich fordert, dass das *Erweiterungspolytop* Q ein *einfaches* Polytop ist, das gesuchte lineare Optimierungsproblem also nicht (primal) degeneriert ist. Einerseits haben bereits bekannte Klassen von Erweiterungspolytopen diese Eigenschaft, andererseits kann man für jedes Polytop zunächst ein einfaches Erweiterungspolytop konstruieren. Die charakteristische Größe ist die sogenannte *simple extension complexity*, die die minimale Facettenzahl

eines solchen einfachen Erweiterungspolytops angibt.

Hierzu wurde eine Technik entwickelt, die es ermöglicht, aufgrund von Eigenschaften des Graphen von P untere Schranken an diese Größe zu bestimmen. Trotz der genannten positiven Beispiele stellt sich heraus, dass die Eigenschaft sehr selten ist: Bereits wenig komplizierte Polytope, wie Hypersimplizes, die zudem nur schwach degeneriert sind, haben eine simple extension complexity in der Größenordnung der Anzahl der Ecken des Polytops, einer trivialen oberen Schranke. Wenig überraschend ist da, dass dies auch für zahlreiche kombinatorische Polytope der Fall ist, unter ihnen perfekte Matching-Polytope vollständiger und vollständiger bipartiter Graphen, Flusspolytope unzerlegbarer azyklischer Netzwerke, Spannbaumpolytope vollständiger Graphen, sowie zufällige 0/1-Polytope mit gewissen Eckenanzahlen. Um die untere Schranke im Falle perfekter Matchings auf vollständigen Graphen zu beweisen, wurde ein Resultat von Padberg & Rao über Adjazenzbeziehungen entsprechender Polytope verbessert.

In einem kurzen Abschnitt dieses Teils wird charakterisiert, wann zwei der (wenigen) bekannten Methoden zur Konstruktion von Erweiterten Formulierungen Einfachheit herstellen.

Summary

An important tool of *Polyhedral Combinatorics* is the computer-aided analysis of polyhedra for understanding their structural properties, which are often related to an underlying optimization problem. Due to the enumerative nature of the classical algorithms for polyhedral analysis this approach can often only be pursued for relatively small dimensions or using an enormous amount of resources.

Since many polyhedra of interest are convex hulls of feasible sets of optimization problems and since there exists efficient software (e.g., in case of mixed-integer linear optimization problems (MIPs)) for solving these problems even in higher dimensions, it is a reasonable question how to carry out an analysis based on such software, or more formally, on an *optimization oracle*.

This thesis is accompanied by a software library IPO (**I**nvestigating **P**olyhedra by **O**racles) specifically designed for this approach. A first step in such an analysis is typically the computation of the affine hull of a polyhedron $P \subseteq \mathbb{R}^n$, in particular its dimension. For this we developed an algorithm that needs at most $2n$ oracle calls and is very efficient in practice. We also prove that this number is the minimum in the worst case. Using our implementation and the MIP solver SCIP we investigated instances of the `MIPLIB 2.0` and determined dimensions of several polyhedra:

- Linear relaxations and mixed-integer hulls before and after applying presolve routines of SCIP.
- Optimal faces with respect to the objective of the instance.
- Faces induced by given inequalities.

The main application of IPO is the detection of some of P's facets since those correspond to undominated inequalities in an inequality description, and hence usually have great potential of being useful during the solving process (using a MIP solver). Facet-defining inequalities that are maximally violated by a given point can be found efficiently using *Target Cuts*. In particular, this is demonstrated for linearizations of matching polytopes with one quadratic term, for which we determined a new class of facets using IPO.

In addition to facet- and affine-hull computations, IPO has a third component that can check adjacency of two vertices of P. The latter is equivalent to the property that the dimension of the normal cone at their barycenter is equal to $n - 1$. Since we can – similar to Target Cuts – construct an optimization oracle for this normal cone, this problem reduces to the affine-hull problem. Again for demonstration purposes we tested a huge number of vertex pairs of TSP polytopes for adjacency, using the software concorde as an optimization oracle.

Since we strive for efficiency, but still want to ensure correct behavior of IPO, we often use floating-point- and rational arithmetic. This is particularly important for dimensions greater than 500 as then sometimes the occurring exact numbers have encoding lengths of more than 10^6 bits. In order to save costly oracle calls, IPO allows to use heuristics from which we do not require optimality of returned solutions. This way we can use, for instance, the rounded (potentially incorrect) solutions of SCIP and use the exact version of SCIP for verification purposes only.

The second part of the thesis belongs to the field of *extended formulations* and is based on a joint publication [41] with Volker Kaibel. The field basically captures how to describe a given polytope P as an affine projection of another polytope Q with only few facets. The main motivation for this is the (in some sense equivalent) question of constructing a linear optimization problem for P with additional variables, but only few inequalities.

In this work we consider the additional restriction of the *extension polytope* Q to be a *simple* polytope, that is, the linear optimization problem over Q has to be (primal) non-degenerate. On the one hand, some of the known classes of extension polytopes have this property, and on the other hand, we can construct a simple extension for every polytope. The characteristic quantity for this task is the so-called *simple extension complexity* that measures the minimum number of facets of such a simple extension polytope Q.

To get a hand on this quantity we developed lower bounds that depend only on the graph of P. Despite the mentioned positive examples, it turns out that this property is very rare: Even very basic polytopes like hypersimplices, which are only slightly degenerate, have a very high simple extension complexity in the order of P's number of vertices, a trivial upper bound. Having this in mind, it is not surprising

that this is also the case for other combinatorial polytopes, among them perfect-matching polytopes of complete and complete bipartite graphs, uncapacitated flow polytopes for nontrivially decomposable directed acyclic graphs, spanning-tree polytopes of complete graphs and random 0/1-polytopes with vertex numbers in a certain range. On our way to obtain the result on perfect-matching polytopes we improve on a result of Padberg and Rao's on the adjacency structures of those polytopes.

In a short section of this part we characterize when two of the (few) known methods for constructing extended formulations actually yield simple polytopes.

Acknowledgements

This thesis would not have been possible without the inspiring environment I found in the group of Volker Kaibel who guided me in the right directions by spreading good ideas, finding faults in the proofs or stopping me from wasting time on things that can't work at all. He left me a lot of room for my own research, but also spent a lot of time for sharing knowledge as well as writing- and presentation skills. I want to thank my colleagues, especially Stefan Weltge, Ferdinand Thein, Jan Krümpelmann and Julia Lange for making IMO such a great place to suspend research for a coffee break. It was a particular pleasure to work in an office with Stefan who is the counterpart in a dream team in which making math is true fun. Together with Volker at the blackboard we often were completely absorbed, discussing with so much emotion that other people on the floor surely suffered from our intensity. I also want to thank Stefan and Volker for improving the quality of the manuscript by thorough proofreading.

Traveling so much allowed me to get in touch with many remarkable and bright people in the field of integer programming and combinatorial optimization. It was a pleasure to collaborate with Michele Conforti, Jon Lee and Klaus Truemper, and to have inspiring discussions with many more. I particularly benefited from advice by Marc Pfetsch who told me about an effective generation scheme for Target Cuts, previous work on stabilized column generation and suggested the consideration of presolved MIP instances for my computational studies. Attending many great talks at several Aussois meetings, IPCOs and ISMPs clearly coined my perspective on our field.

The computational aspect of this thesis builds on the work of many people. I thank Roland Wunderling for creating SoPlex, Tobias Achterberg for creating SCIP and Ambros Gleixner for turning SoPlex into an exact arithmetic LP solver to which I had access in early development stages. Although I bothered Ambros with several bugs I could usually find a corresponding fix at 5:33[1] the next day. General thanks go to the whole development team of the SCIPOptSuite. I am grateful to Bill Cook who provided me with code I could turn into an exact arithmetic solver for the traveling salesman problem, and to him and his coau-

[1] See commit `8efd6801ef46039f85b75ad04b09d26b4b15b989` of the SoPlex-repository.

thors for their work on the concorde TSP solver. For helping me with statistical issues I want to thank Martin Radloff.

Finally I want to say *thank you* to all my family. To my parents Gabriele and Karl-Heinz who always encouraged me to give way to my mathematical interests, to my brother Tobias who taught me programming when I was quite young, and to my wife Sarah and our son Lukas. The two probably have the smallest mathematical contribution among the people I listed here, but are nevertheless the most important ones in my life, giving me nothing more than a home full of love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

INTRODUCTION:

1.1 Polyhedral Combinatorics

Many combinatorial optimization problems can be stated as the problem to find a subset $F \subseteq E$ of a finite ground set having certain properties such that a given cost function is minimized. If the cost function is linear, i.e., there exists a vector $c \in \mathbb{R}^E$ such that the costs are $\sum_{e \in F} c_e$, then we can formulate this as a *linear optimization problem* (LP)

$$\min c^\intercal x \text{ subject to } x \in \text{conv.hull}\left\{\chi(F) \in \{0,1\}^E \mid F \subseteq E \text{ is feasible}\right\},$$

where $\chi(F)$ is F's *characteristic vector*, i.e, $\chi(F)_e = 1$ holds if and only if $e \in F$ holds and conv.hull $(\cdot)$ denotes the convex hull.

Such linear programming formulations are a standard tool to gain structural insight, derive algorithms and to analyze complexity. In order to solve such an LP we need to describe it by means of linear inequalities, i.e., $Ax \leq b$, which is a nontrivial task in general: For many polyhedra for which the associated optimization problem is solvable in polynomial time such a description is known, but of exponential size (in the dimension). Even worse, for NP-hard problems we should not expect that we can characterize such a description in a certain nice way, since this would imply NP = coNP, that is, every propositional tautology would have a polynomial-size proof. Nevertheless, for practically *solving* these problems, good *linear relaxations* are important. Understanding these polytopes is (part of) the branch of discrete mathematics called *Polyhedral Combinatorics*. This thesis contributes with new techniques, results and software to this area.

1.1.1 Investigating Polyhedra by Oracles

In the first part we address questions regarding the computer-aided analysis of polyhedra that are only implicitly defined by means of a black-box algorithm (a so-called *oracle*) solving the optimization problem. At first glance, this approach may seem rather complicated, but it is quite effective in practice if we have access to a solver for *mixed-integer linear optimization problems* (MIPs). In this situation we can simply give a mathematical model as the input.

Fundamental work on this topic was done by Grötschel, Lovász and Schrijver in 1981, when they established the famous theorem on "equivalence of optimization and separation" [33]. It essentially states that solving the linear optimization problems over a rational polytope P is equivalent (with respect to polynomial-time solvability) to finding, for a given rational point $\hat{x}$, some hyperplane that separates $\hat{x}$ from P. From this result one can derive that one can carry out certain tasks like finding facets, determining the dimension, or optimizing over faces in a theoretically efficient way (see Chapter 14 in Schrijver's book [62]). Unfortunately, the result and its applications heavily depend on the Ellipsoid method, a theoretically efficient, but practically almost useless algorithm.

In Chapter 2, we will present *practically efficient* algorithms for some of these problems, describe the underlying ideas, provide the reader with proofs of correctness, and introduce a new software library IPO that implements them. Finally, we carry out small computational studies on different problems showing the software's capabilities and limitations.

1.1.2 Analyzing Simple Extensions of Polytopes

Chapter 3, the second part of this thesis, deals with our contributions to the theory of *extended formulations*, a recently very active area of Polyhedral Combinatorics. Consider once again some polytope P associated to a class of certain combinatorial sets. Even if a description of P by a system of linear inequalities is known, it may still be impractically large.

Sometimes it is possible to write P as a projection of some other polytope Q that needs fewer inequalities for the price of a higher dimension, i.e., more variables. Then we call Q together with the projection map an *extension* of P. The theory of extended formulations mainly deals with

the question of the *minimum* number of inequalities actually required to describe P, its so-called *extension complexity*.

Inspired by this phenomenon, one may even dare to ask for an extension polytope Q that is simple in the sense that every vertex of Q lies in exactly d facets, where d denotes the dimension of Q. Unfortunately, we should not be too optimistic when asking for few facets *and* non-degeneracy at the same time. It turns out that several well-understood combinatorial classes of polytopes do not have such an extension.

To prove such results we develop a technique for establishing lower bounds on the extension complexity when only simple extensions are permitted. In subsequent sections we apply it to different classes of combinatorial polytopes. In one particular case, namely for perfect matching polytopes of complete graphs, we have to improve a result of Padberg and Rao's on adjacencies of perfect matchings [52] (see Theorem 3.7.6). All results of this chapter, except for Theorem 3.7.1, are already published in the following article:

- Volker Kaibel and Matthias Walter: *Simple extensions of polytopes*. Mathematical Programming Series B, 154(1-2): 381–406, 2015.

INTRODUCTION:

1.2 Preliminaries

In this section we introduce our basic notation without going into details. If appropriate, we point the reader to the relevant literature. Whenever we focus on a combinatorial problem, we will introduce the necessary concepts locally.

1.2.1 Basics

By $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{Z}$ and $\mathbb{N}$ we denote the real numbers, the rationals, integers and the natural[1] numbers, respectively (resp.). Using a "+" or "−" in the subscript (e.g., $\mathbb{R}_+$ or $\mathbb{R}_-$), we denote the corresponding restrictions to non-negative (resp. non-positive) numbers. We write $[n] := \{1, 2, \ldots, n\}$. The greatest common divisor of two integers a and b is denoted by $\gcd(a, b)$. For two sets A, B we denote by $A \Delta B := (A \setminus B) \cup (B \setminus A)$ the symmetric difference, and write $A \uplus B$ for the disjoint union, i.e., we implicitly require $A \cap B = \emptyset$. As usual, the boundary, relative boundary, the interior and the relative interior of a set A are denoted by $\operatorname{bd}(A)$, $\operatorname{relbd}(A)$, $\operatorname{int}(A)$ and $\operatorname{relint}(A)$, respectively. By $\operatorname{proj}_x(A)$ we denote the orthogonal projection of a set A on the subspace indexed by the x-variables (which are clear from the context).

To state asymptotic behavior, we use the usual Bachmann-Landau notation $O(\cdot)$, $\Omega(\cdot)$, $\Theta(\cdot)$ and $o(\cdot)$.

[1] Ignoring DIN norm 5473, we consider 0 not to be a natural number.

1.2.2 Vectors, Matrices and Linear Algebra

We make use of the standard scalar product $\langle \cdot, \cdot \rangle$ in the Euclidean space. The zero vector in $\mathbb{R}^n$ and the $m \times n$ all-zeros matrix are denoted by $\mathbb{0}_n$ and $\mathbb{0}_{m,n}$, respectively, where we omit the subscript whenever the dimension is clear. Similarly, the all-ones vector is $\mathbb{1} := \mathbb{1}_n$, and the i-th unit vector is denoted by $e^{(i)}$. As already stated in the introduction, the characteristic vector $\chi(F)$ of a subset $F \subseteq E$ is the binary vector with $\chi(F)_e = 1$ if and only if $e \in F$. For the product $\langle v, \chi(F) \rangle$ we use the notation $v(F)$. The $n \times n$ unit matrix is denoted by $\mathbb{I}_n$ and transposition of vectors and matrices by $(\cdot)^\intercal$. For a matrix $A \in \mathbb{R}^{m \times n}$, a row subset $I \subseteq [m]$ and a column subset $J \subseteq [n]$, we write $A_{I,J}$ for the submatrix indexed by the corresponding rows and columns. Instead of $A_{I,[n]}$ and $A_{[m],J}$ we also write $A_{I,*}$ and $A_{*,J}$, respectively, and for the single entries $A_{\{i\},\{j\}}$ we write $A_{i,j}$.

For the set of rows of a matrix A, considered as vectors, we write $\operatorname{rows}(A)$ and for the *nullspace* of A we write $\ker(A)$. The *dimension* $\dim(X)$ of a set $X \subseteq \mathbb{R}^n$ is defined as the dimension of its *affine hull* (denoted by $\operatorname{aff.hull}(X)$), i.e., the dimension of the smallest affine space that contains X. We say that X is *full-dimensional* if $\dim(X) = n$ holds. As usual, the *linear hull* (denoted by $\operatorname{lin.hull}(X)$) is the set of all linear combinations of elements of X.

1.2.3 Polyhedra

For precise definitions and basic properties of convex sets and polyhedra we refer to [61] and [62], respectively. As already done in the introduction, we write $\operatorname{conv.hull}(X)$ for the convex hull and $\operatorname{conic.hull}(X)$ for the convex conic hull of X, respectively. A *polyhedron* $P \subseteq \mathbb{R}^n$ is the intersection of finitely many halfspaces, that is, $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ for some matrix $A \in \mathbb{R}^{m \times n}$ and some vector $b \in \mathbb{R}^m$. *Polytopes* are bounded polyhedra or, equivalently, convex hulls of finitely many points. We say that a direction $v \in \mathbb{R}^n$ is *unbounded* with respect to (w.r.t.) P if P is non-empty and $x + \lambda v \in P$ holds for some $x \in P$ and for all $\lambda \geq 0$. The set $\operatorname{recc}(P) := \{v \in \mathbb{R}^n \mid v \text{ is unbounded w.r.t. } P\}$ is the so-called *recession cone* of P (which is a polyhedral cone). By Minkowski-Weyl's Theorem (see Corollary 7.1b in [62]), every polyhedron P is the sum of a polytope

and P's recession cone, that is,

$$P = \text{conv.hull}(S) + \text{conic.hull}(R) \text{ for finite } S \subseteq P \text{ and finite } R \subseteq \text{recc}(P).$$

We call such a pair (S, R) an *inner description* of P, whereas we refer to $Ax \leq b$ as an *outer description*. If S and R are sets of rational vectors, we say that P is a *rational polyhedron*. The *lineality space*, defined as $\text{lineal}(P) := \text{recc}(P) \cap \text{recc}(-P)$, is the set of all directions corresponding to lines contained in P. We say that P is *pointed* if $\text{lineal}(P) = \{0\}$ holds.

Farkas' Lemma (see Corollary 7.1e in [62]) states that either the system $Ax \leq b$ (for $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$) has a solution or there exists a nonnegative vector y with $y^\intercal A = 0$ and $y^\intercal b < 0$.

We say that an inequality $\langle a, x \rangle \leq \beta$ (for $a \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$) is *valid* for a polyhedron P if it is satisfied by all points in P. A *face* of a polyhedron P is a subset $F \subseteq P$ that is *induced* by a valid inequality $\langle a, x \rangle \leq \beta$ in the sense that $F = \{x \in P \mid \langle a, x \rangle = \beta\}$ holds. For non-empty faces F we also say that F is the *a-maximum face* of P since then clearly $F = \arg\max \{\langle a, x \rangle \mid x \in P\}$ holds. The faces of a polyhedron P form a graded lattice $\mathcal{L}(P)$ (in the sense of a partially ordered set), ordered by inclusion (see [70]). The $(d-1)$-dimensional faces of a d-dimensional polyhedron are called *facets* and the 0-dimensional faces are called *vertices*. Although vertices are formally singleton sets $\{x\}$ for some $x \in P$, we usually just write x. The set of all vertices of a polyhedron P is denoted by $\text{vert}(P)$. Note that only pointed polyhedra actually have actually have vertices. As motivated in the introduction, 0/1-polytopes, that is, polytopes whose vertices have only 0/1-coordinates, are of special interest in Polyhedral Combinatorics. The bounded (resp. unbounded) 1-dimensional faces of pointed polyhedra are called *edges* (resp. *extreme rays*). The set of direction vectors corresponding to the extreme rays is denoted by $\text{ext.rays}(P)$. The *irredundant* (that is, inclusion-wise minimal) inner description of a pointed polyhedron P is the pair $(\text{vert}(P), \text{ext.rays}(P))$.

For a polyhedron $P \subseteq \mathbb{R}^n$ and a point $\hat{x} \in P$ we will use the *radial cone*, defined as $\text{rad.cone}_{\hat{x}}(P) := \text{conic.hull}(P - \hat{x})$, and its *polar* cone $\text{nml.cone}_{\hat{x}}(P) := \{y \in \mathbb{R}^n \mid \langle y, x - \hat{x} \rangle \leq 0 \text{ for all } x \in P\}$, called the *normal cone*. Intuitively, the radial cone contains all directions one can go from $\hat{x}$ without immediately leaving P, whereas the normal cone consists of all objective vectors whose maximum face contains $\hat{x}$.

In order to use (sets of) inequalities or equations as input or output

of algorithms, we will often consider system $Ax \leq b$ or $Cx = d$ as objects, where we formally mean the pairs (A, b) and (C, d), respectively.

1.2.4 Linear and Mixed-Integer Linear Optimization

A *linear optimization problem* (*LP*) is an optimization problem of the form

$$\max \langle c, x \rangle \text{ subject to (s.t.) } Ax \leq b.$$

for a matrix $A \in \mathbb{R}^{m \times n}$ and vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. Of course we can also minimize, allow equations, "$\geq$"-inequalities, or special inequalities like $x \geq \mathbb{0}$, so-called *variable bounds*. We often use the phrase of "optimizing over a polyhedron P" which means to solve linear optimization problems with P as the feasible region and arbitrary linear objective functions.

A *mixed-integer optimization problem* (*MIP*) is an LP with the additional restriction that a subset $I \subseteq [n]$ of variables must be integral, that is, $x \in \mathcal{I} := \{x \in \mathbb{R}^n \mid x_i \in \mathbb{Z} \text{ for all } i \in I\}$ holds. The feasible points are sometimes called *integer-feasible* to distinguish from *LP-feasibility* for which we ignore the integrality constraints. The special case of $I = [n]$ is called just an *integer program* (*IP*).

A polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ is called a *linear relaxation* of a set $X \subseteq \mathcal{I}$ (w.r.t. I) if $P \cap \mathcal{I} = X$ holds. Its corresponding *mixed-integer hull* is the set $P_I := \text{conv.hull}\,(P \cap \mathcal{I})$, that is, $P_I = \text{conv.hull}\,(X)$.

1.2.5 Graphs

Most of the combinatorial optimization problems we are concerned with are defined for graphs. We will only introduce notation here, and refer to the books of Schrijver [63] and Korte and Vygen [47]. They are not graph-theoretic, but introduce all concepts that are necessary to deal with our problems.

We mostly deal with undirected graphs $G = (V, E)$ consisting of a set V of *nodes*[2] and a set E of *edges*, where an edge is a 2-element set of nodes. In particular, all our graphs are simple (no edge exists twice) and loopless (edges connect distinct nodes). We often consider the complete graphs K_n with n nodes and all $\binom{n}{2}$ possible edges. We denote for a set

[2]Note that we only use the term "vertices" for the 0-dimensional faces of polyhedra.

$S \subseteq V$ of nodes by $E[S]$ the set of edges whose endpoints lie in S. By $\delta(S)$ we denote the *cut induced by* S, that is, the set of edges having precisely one endnode in S. We abbreviate $\delta(\{v\})$ by $\delta(v)$ for single nodes $v \in V$. For an edge set $F \subseteq E$ we denote by $V(F)$ the set of all nodes of edges in F.

In Section 3.6 we also consider directed graphs $G = (V, A)$ with node sets V and *arcs* A which are pairs of nodes. We will introduce suitable notation locally.

Another type of graphs we consider is the so-called 1-skeleton of a polytope whose nodes are the vertices and whose edges are the edges (1-dimensional faces) of the polytope.

1.2.6 Computational Complexity

Although we do not directly touch the field of computational complexity, we benefited a lot from it. As argued in the introduction we can use it to guide expectations on the possibility to characterize facial structures of polyhedra. We won't define the complexity classes, P, NP and coNP here, but refer to standard literature [29], [7].

Chapter 2

Investigating Polyhedra by Oracles

INVESTIGATING POLYHEDRA BY ORACLES:

2.1 Motivation: Mixed-Integer Hulls

As described in the introduction, one goal of Polyhedral Combinatorics is to study (facial structures of) polytopes of which one implicitly knows the extreme points. Since in most of the applications one also knows some linear relaxation, a MIP solver can be used to optimize over the polytopes. In general, it is advantageous for a MIP solver if the relaxation approximates the corresponding mixed-integer hull very well, where inequalities that are not dominated by others (e.g., within a bounding box) are particularly interesting. Geometrically, this means that the face induced by such an inequality is a facet. Hence, a typical task for researchers is to identify such facets for a given linear relaxation.

One way of doing this is to first compute all extreme points $X \subseteq \mathbb{R}^n$ of the mixed-integer hull, either by developing problem-specific software or by using tools that do this automatically for a given relaxation. For the latter task, there are many different types of algorithms and even more different implementations. For an overview and a comparison we refer to [8], where the authors compare all tools that are interfaced by the polymake system [56]. Note that there are binary sets $X \subseteq \{0,1\}^n$ for which every linear relaxation in the original space has exponentially many inequalities (see [42]), which is a problem for such tools since they get such a relaxation as an input.

As a second step, this set X is usually given as input to some convex hull tool which computes a set of linear equations defining aff.hull(X) and one inequality representing each facet of conv.hull(X). Again, there are many algorithms for this task, and we refer to [8] for a comparison.

Finally, the researcher has to analyze the output, understand why the inequalities are valid for X, identify structure, classify them, and prove validity. Establishing the facet-defining property of a class of inequalities is of course not necessary for a practical application, still it is of interest since then the inequalities are not dominated by others. Often, such classes contain exponentially (in the dimension) many facets, and hence, one is also interested in the investigation of the separation problem for a class. The work-flow described above is visualized in Figure 2.1.

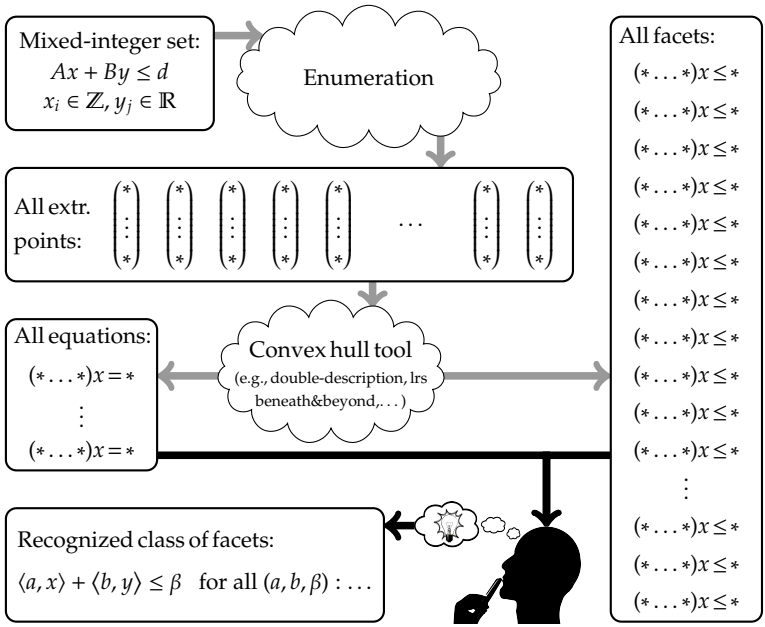


Figure 2.1: Traditional work-flow in polyhedral combinatorics.

Motivated by the fact that this approach is limited by the shear amount of data, namely extreme points and facets, we propose new ideas how to still find new facet classes. Although the work-flow described above usually fails for dimensions larger than 20, actually *solving* the corresponding optimization problem using a MIP solver is nowa-

days a matter of a second. The main goal of this part of the thesis is to develop algorithms that make use of this observation to overcome some of the issues mentioned before.

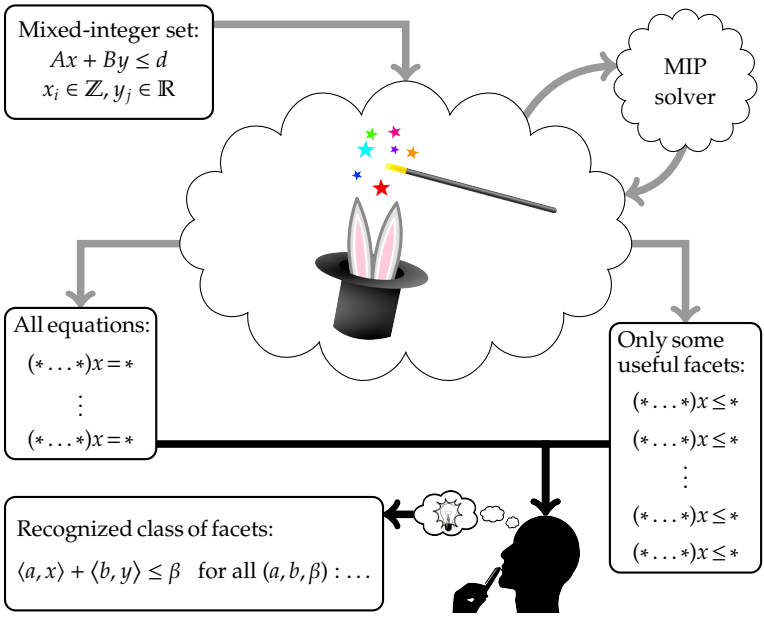


Figure 2.2: Proposed new work-flow in polyhedral combinatorics.

For instance, in order to still obtain all structural information with respect to an outer description consisting of facets, it offen suffices to compute a dozen or a hundred of them. First, a researcher will not be able to inspect all existing facets, anyway. Second, due to symmetry, often the facets are also very similar to each other, hence it may well be that all facet classes[1] actually have a representative in our small collection! This proposed new work-flow is visualized in Figure 2.2.

Sometimes one is interested in optimizing a specific class of objective functions, e.g., nonnegative costs, or metric distance weights. In such a

[1]Note that there is no formal definition of a facet class, but in practice it consists of facets that expose a similar structure.

case, there may exist many inequalities that define facets, but will never be relevant for objective functions from the respective class. The method for facet-detection we will present has the remarkable property that it may be controlled by an objective vector. Hence, by choosing objective vectors from the set of interesting ones we can at least slightly influence which facets will be detected.

INVESTIGATING POLYHEDRA BY ORACLES:

2.2 The Software Library IPO

Motivated by the observations from the previous section we developed the software library IPO (**I**nvestigating **P**olyhedra by **O**racles, [67]). We now briefly describe its functionality, leaving the details for Sections 2.3–2.8.

The user has to provide IPO with a so-called *optimization oracle*, which is a black-box procedure for maximizing any given linear objective function over a certain polyhedron. We will define this formally in the next section. In principle this can be any algorithm that solves a certain type of optimization problems, but in particular, it may be a MIP model together with a solver. For the latter case, IPO already interfaces the solver SCIP [1], which can solve an even broader problem class than just MIPs. It does not matter whether the SCIP optimization oracle was created for an explicit MIP or whether it uses so-called *constraint handlers* that dynamically enforce some of the constraints, e.g., by solving separation problems. Hence, one can also use oracles for problems that allow no small LP relaxation. Since most of the MIP solvers work with floating-point arithmetic, one may also use the exact MIP solver ExactSCIP (see [20], [64] and [19] for details) for numerically challenging problems, taking a considerable increase in running time into account.

We assume that the oracle implicitly defines a polyhedron $P := \text{conv.hull}(S) + \text{conic.hull}(R)$, where S is the set of all points that the oracle may return as optimal, and R is the set of direction vectors that the oracle may return as unbounded directions. Note that just by this

definition P could be any convex closed set, a generalization not considered here. A description of our requirements, including the one that an oracle must not contradict its previous answers, can be found in Section 2.3.

In addition to user-written optimization oracles and the predefined `SCIPOptimizationOracle` and `ExactSCIPOptimizationOracle` (interfacing SCIP and ExactSCIP, respectively), the user can easily derive new oracles from existing ones without much effort. Among those "wrapper" optimization oracles are the ones for the recession cone, for arbitrary faces and affine projections of the polyhedron defined by a given oracle.

2.2.1 Outline

Using an optimization oracle for a polyhedron $P \subseteq \mathbb{R}^n$, IPO can

- compute the affine hull of P in the sense that it returns $(d + 1)$ many affinely independent points in P and $(n - d)$ many linearly independent equations valid for P, where d is the dimension of P. The details are discussed in Section 2.4,

- compute a facet-defining inequality that is violated by a given point. The technique is based on *Target Cuts* developed by Buchheim, Liers and Oswald [15] in 2008, and presented in Section 2.5.

- compute facets that are "helpful" when maximizing a specified objective vector c. More precisely, it iteratively solves LPs whose inequalities correspond to some of P's facets, until the current optimum is in P. As long as this is not the case, the procedure returns violated facet-defining inequalities and adds them to the LP.

- check whether two given vertices of P are adjacent. In Section 2.6.2 we reduce this problem to the next one:

- compute the smallest face of P containing a given point by means of an inequality defining this face. Check whether a given point is a vertex of P. Section 2.6 is dedicated to this topic.

Section 2.7 deals with an auxiliary problem that needs to be solved for the facet-computation as well as for the computation of the smallest containing face. It is essentially the separation problem for the cone of inequalities valid for a polyhedron.

In Section 2.8 we develop methods that linearly combine two integer vectors in order to produce one of minimum Manhattan norm (without allowing the zero vector). This is useful to post-process IPO's output: first, equations may be combined to become more readable. Second, the equations may be combined with a computed inequality to improve the readability of the latter.

The capabilities of IPO are demonstrated in several small computational studies. For the affine hull computations we considered the linear relaxations and the mixed-integer hulls for a subset of the MIPLIB 2.0 instances [14]. We computed their dimensions, the dimensions of their optimal faces and for the mixed-integer hulls also the dimensions of the faces induced by the inequality constraints specified in the model instance. The results are presented in Section 2.9.

For the facet computation we investigated integer hulls for several integer programs. First we considered polytopes associated to matching problems with objective functions consisting of a linear term plus a single product term. Section 2.10 shows how we used IPO to compute several unclassified facets, and also contains proofs that the identified classes indeed consists of valid facet-defining inequalities. As a second example we considered inequality constraints that link node sets and edge sets in different ways. We deal with outer descriptions of the corresponding integer hulls and with computational complexity of the associated optimization problems. IPO requires a fractional solution to be cut off by a facet, and such a solution is sometimes hard to find. Hence, in a third example we present a strategy to come up with such solutions, in our case based on a polyhedral reduction from vertex cover to tree polytopes.

In our last experiment we used the famous solver concorde [4] as an oracle for the *traveling salesman polytope* to check many random vertex pairs for adjacency. We present some statistics, and describe a technique to produce adjacent pairs by node splitting.

2.2.2 IPO's User Interface

In order to make IPO user-friendly, the following programs also belong to the library:

- `ipo-facets`: Input is any instance that SCIP can read, in particular MIPs in `LP` or `MPS` format and models written in the ZIMPL modelling language [45]. It uses the objective from the MIP to compute "helpful" facets (see above).
- `ipo-dimensions`: Input is as for `ipo-facets`, together with a choice of whether it shall compute the dimension of the linear relaxation, of the mixed-integer hull, of the optimal face of either of them, or of all faces induced by the inequality constraints given in the model.
- `ipo-smallest-face`: Input is as for `ipo-facets`, together with a point $s \in P$. It then computes the dimension of the smallest face that contains s. This can be used to check adjacency of two vertices (e.g., s is their midpoint), as discussed in Section 2.6.3.

INVESTIGATING POLYHEDRA BY ORACLES:

2.3 Optimization Oracles

In this section we will specify what IPO requires from an optimization oracle in order to work correctly. Then we discuss how to derive from an optimization oracle for a polyhedron those for related polyhedra (e.g., faces and projections). In the last subsection we introduce a special optimization oracle that is helpful when analyzing mixed-integer hulls with the help of a floating-point based MIP solver.

A proper *optimization oracle* $\mathfrak{O}$ for a rational polyhedron $P \subseteq \mathbb{R}^n$ is an oracle that can be called for any direction $c \in \mathbb{Q}^n$ (we denote this by $\mathfrak{O}(c)$) and returns

- $\mathfrak{O}(c).\text{val} = -\infty$ if $P = \emptyset$ holds,
- $\mathfrak{O}(c).\text{val} = +\infty$ and a rational direction $r := \mathfrak{O}(c).\text{dir} \in \text{recc}(P)$ satisfying $\langle c, r\rangle > 0$ if $\sup\{\langle c, x\rangle \mid x \in P\} = \infty$,
- and $\delta := \mathfrak{O}(c).\text{val} \in \mathbb{Q}$ and a rational point $s := \mathfrak{O}(c).\text{point} \in P$ satisfying $\langle c, s\rangle = \delta = \max\{\langle c, x\rangle \mid x \in P\}$ otherwise.

In an implementation, such an oracle usually also has functionality to obtain P's ambient dimension n, and hence we will always assume that n is known.

We say that $\mathfrak{O}$ is *finite* if the set of all possible answers is finite, i.e., if

$$|\{\mathfrak{O}(c).\text{dir} \mid c \in \mathbb{Q}^n \text{ with } \mathfrak{O}(c).\text{val} = +\infty\}| < \infty \text{ and}$$
$$|\{\mathfrak{O}(c).\text{point} \mid c \in \mathbb{Q}^n \text{ with } \mathfrak{O}(c).\text{val} \neq \pm\infty\}| < \infty$$

hold. We will often use finiteness of oracles in order to prove termination of our algorithms. In several cases we will not be able to give proofs on the actual running time since we make heavy use of the Simplex method for which no polynomial-time version is known.

Many of the algorithms we present do not always require optimality of the solutions returned by an oracle. Hence it is useful to also consider oracles that still guarantee to return feasible solutions but not necessarily return optimal ones. We call such an oracle a *heuristic optimization oracle* (in contrast to *proper* optimization oracles). Often, our algorithms require optimality only in certain steps, so we can sometimes make progress using a heuristic optimization oracle first without losing correctness. A simple example is to use a standard MIP solver, which is prone to rounding errors triggered by float-point arithmetic, as a heuristic and a slower exact MIP solver as a proper optimization oracle.

2.3.1 Optimization Oracle for the Recession Cone

Let $\mathfrak{O}$ be an optimization oracle for a non-empty rational polyhedron $P \subseteq \mathbb{R}^n$ and let $C := \operatorname{recc}(P)$ be its recession cone. We can very easily derive an optimization oracle $\mathfrak{O}_C$ for C from $\mathfrak{O}$:

Given a direction $c \in \mathbb{Q}^m$, call $\mathfrak{O}(c)$. If $\mathfrak{O}(c).\text{val} = +\infty$ holds, then $\mathfrak{O}_C$ also returns $\mathfrak{O}(c).\text{dir}$ since the latter lies in C. Otherwise, i.e., if $\mathfrak{O}(c).\text{val} \neq \pm\infty$ holds, then $\mathfrak{O}_C$ returns $\mathbb{O}$ as the optimum point. The origin is optimal since any point $y \in C$ with a positive objective value would immediately make $\max\{\langle c, x\rangle \mid x \in P\}$ unbounded because we have $\lambda y \in \operatorname{recc}(P)$ for any $\lambda \geq 0$, i.e., $\mathfrak{O}$ would have returned $\mathfrak{O}(c).\text{val} = +\infty$.

2.3.2 Optimization Oracles for Projections of Polyhedra

Let $\mathfrak{O}$ be an optimization oracle for a rational polyhedron $P \subseteq \mathbb{R}^n$. Let furthermore $A \in \mathbb{Q}^{m\times n}$ be a matrix and $b \in \mathbb{Q}^m$ be a vector defining the projection map $\pi : \mathbb{R}^n \to \mathbb{R}^m$ via $\pi(x) := Ax + b$. Let $Q := \pi(P)$ be the image of the projection.

In this case an optimization oracle $\mathfrak{O}_Q$ for Q can be obtained in a straight-forward way: Given a direction $c \in \mathbb{Q}^m$, call $\mathfrak{O}(A^\intercal c)$. Clearly, $P = \emptyset$ holds if and only if $Q = \emptyset$ holds. If $\mathfrak{O}(A^\intercal c).\text{val} = +\infty$ holds, then $\mathfrak{O}_Q$ also returns an unbounded direction Ar with $r = \mathfrak{O}(A^\intercal c).\text{dir}$. To

see that this is correct, observe $\langle c, Ar\rangle = \langle A^\intercal c, r\rangle > 0$ and consider some $x \in P$. For all $\lambda \geq 0$ we have $x + \lambda r \in P$, and hence $(Ax + b) + \lambda Ar \in Q$, i.e., $Ar \in \operatorname{recc}(Q)$. In the last case, i.e., if $\mathfrak{O}(A^\intercal c).\text{val} \neq \pm\infty$ holds, then $\mathfrak{O}_Q$ also returns a point $\pi(s)$ with $s = \mathfrak{O}(A^\intercal c).\text{point}$. This is justified by the fact that for any $x, x' \in \mathbb{R}^n$ we have

$$\langle A^\intercal c, x\rangle \leq \langle A^\intercal c, x'\rangle \iff \langle c, Ax + b\rangle \leq \langle c, Ax' + b\rangle .$$

2.3.3 Optimization Oracles for Faces of Polyhedra

In contrast to the construction of an optimization oracle for a projection of a polyhedron P, it is harder to create one for a face F of $P \subseteq \mathbb{R}^n$ without relying on the equivalence of optimization and separation (see Chapter 14 in Schrijver's book [63]). In contrast, note that creating a separation oracle for a face from a separation oracle for the polyhedron is trivial.

One idea for a direct construction is to tilt a given objective vector such that an optimal point is contained in F or an unbounded direction is also in the recession cone of F. Unfortunately, the last property is too restrictive without further assumptions as shown in the following example.

Example 2.3.1. *Consider the polyhedron $P = \mathbb{R}\times\mathbb{R}_-$ with its face $F = \mathbb{R}\times\{0\}$, and the objective $c = (1, 0)^\intercal$. Clearly, maximizing c over F is an unbounded LP since $\operatorname{recc}(F) = \operatorname{lin.hull}\{c\}$ holds. Unfortunately, for arbitrary $M > 0$, when called with the tilted objective vector $\tilde{c} = \frac{1}{M}c + (0, 1)^\intercal$, an oracle for P may answer with the unbounded ray $(2, -\frac{1}{M})^\intercal \in \operatorname{recc}(P) \setminus \operatorname{recc}(F)$ since its scalar product with $\tilde{c}$ is positive.*

Schrijver's proofs (see Chapter 14 in [62]) avoid this phenomenon by requiring bounds on P's encoding size, which even allows to compute a sufficiently large constant M a-priori. We do not want to do this, in particular because such an M would be impractically large, but instead focus on finite oracles. Note that this is a generalization of the usual strategy since an oracle whose answers have bounded encoding lengths can only give finitely many different answers. Unfortunately, despite finiteness of oracles, M can still be arbitrarily large, and hence we have to determine it on the fly by the following algorithm:

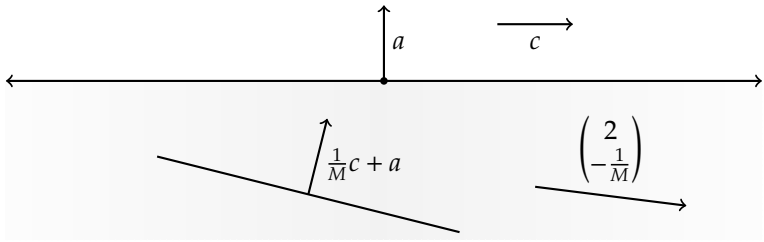


Figure 2.3: Illustration of Example 2.3.1.

Algorithm 2.3.1: Optimization Oracle for a Face

Input: Optimization oracle $\mathfrak{O}$ for a rational polyhedron $P \subseteq \mathbb{R}^n$, a rational inequality $\langle a, x \rangle \leq \beta$ defining a non-empty face F of P and an objective vector $c \in \mathbb{Q}^n$.

Output: A direction $r \in \operatorname{recc}(F)$ with $\langle c, r \rangle > 0$ or a point $s \in F$ with $\langle c, s \rangle = \max\{\langle c, x \rangle \mid x \in F\}$.

1 **for** $M := 1, 2, 4, 8, \ldots$ **do**
2 Compute $\tilde{c} := c + Ma$.
3 **if** $\mathfrak{O}(\tilde{c}).\text{val} = +\infty$ **and** $\langle a, \mathfrak{O}(\tilde{c}).\text{dir} \rangle = 0$ **then**
4 **return** $\mathfrak{O}(\tilde{c}).\text{dir}$.
5 **end**
6 **if** $\mathfrak{O}(\tilde{c}).\text{val} \neq \pm\infty$ **and** $\langle a, \mathfrak{O}(\tilde{c}).\text{point} \rangle = \beta$ **then**
7 **return** $\mathfrak{O}(\tilde{c}).\text{point}$.
8 **end**
9 **end**

Proposition 2.3.2. *When called for a finite optimization oracle $\mathfrak{O}$, Algorithm 2.3.1 works correctly and yields a finite optimization oracle.*

Proof. On termination, the algorithm clearly returns correct answers as ensured by the conditions in Steps 3 and 6. Furthermore, since it only returns points and directions that were themselves returned by $\mathfrak{O}$, the finiteness property is also clear. Hence we only have to prove that it terminates.

Let $\overline{x} \in F$ be some point in the non-empty face F and consider any $s \in P \setminus F$. Define $M(s) := \frac{\langle c, s-\overline{x}\rangle+1}{\beta-\langle a,s\rangle}$ (note that $\langle a, s\rangle < \beta$ holds). We now observe that

$$\langle c + M(s)a, \overline{x} - s\rangle = \langle c, \overline{x} - s\rangle + \frac{\langle c, s - \overline{x}\rangle + 1}{\beta - \langle a, s\rangle}(\beta - \langle a, s\rangle) > 0$$

holds, i.e., no M greater than $M(s)$ will yield s as a $(c + Ma)$-maximum point.

Also observe that for every direction $r \in \operatorname{recc}(P) \setminus \operatorname{recc}(F)$ we have $\langle a, r\rangle < 0$. Hence, there exists a number $\vec{M}(r) > 0$ with $\langle c + \vec{M}(r)a, r\rangle \geq 0$.

Now consider an M^* (from the algorithm) that is greater than $M(s)$ for all (finitely many) points s that the oracle may return and greater than $\vec{M}(r)$ for all (finitely many) directions r that the oracle may return. For M^*, the conditions $\langle a, \mathfrak{O}(\tilde{c}).\text{dir}\rangle = 0$ and $\langle a, \mathfrak{O}(\tilde{c}).\text{point}\rangle = \beta$ must be satisfied by the arguments above. □

2.3.4 Corrector Oracle for Mixed-Integer Programs

Since the polyhedra of our main interest are mixed-integer hulls, we often use MIP solvers as heuristics and oracles. It is not straight-forward to use a floating-point precision MIP solver for such a task as our oracle specification requires them to return results in exact arithmetic. Clearly, for pure integer programs this is not an issue as we can simply round the returned vectors component-wise[2].

In order to handle at least explicitly given mixed-integer programs, IPO implements a `MixedIntegerProgramCorrectorOracle` that does the following: Given an approximate solution $(\tilde{x}, \tilde{y}) \in \mathbb{R}^n \times \mathbb{R}^d$ to the MIP $\max\left\{\langle c, x\rangle + \langle d, y\rangle \mid Ax + By \leq g, y \in \mathbb{Z}^d\right\}$, it rounds $\tilde{y}$ component-wise to the nearest integer and obtains vector $\hat{y} \in \mathbb{Z}^d$. Then, the LP $\max\{\langle c, x\rangle \mid Ax \leq g - B\hat{y}\}$ (with fixed $\hat{y}$) is solved with SoPlex [68], which can be configured to return exact solutions. This rational solution $\hat{x}$ is augmented by $\hat{y}$ and returned. If the LP is unbounded, then the original problem is unbounded as well. If the LP is infeasible, an exception

[2] In general rounding may produce infeasible points, but in such a case we probably have numerical troubles, anyway.

is raised, since clearly the answer of the floating-point precision MIP solver is not even approximately feasible.

INVESTIGATING POLYHEDRA BY ORACLES:

2.4 Computing the Affine Hull

This section is dedicated to the following problem. Given an optimization oracle $\mathfrak{O}$ for a rational polyhedron $P \subseteq \mathbb{R}^n$, we want to determine P's affine hull. A very obvious (and useful) certificate consists of an inner description, i.e., an affine basis of aff.hull (P) and an outer description of aff.hull (P), i.e, an (irredundant) set of equations valid for P. We first present a simple algorithm that produces such a certificate using at most $2n+1$ oracle calls. In Section 2.4.2 we refine the algorithm to reduce this number to $2n$, also adding several other improvements. Section 2.4.3 is dedicated to the proof that that no oracle-based algorithm can do better. In order to exploit presence of heuristic optimization oracles we developed an algorithm that first runs our refined algorithm for a heuristic oracle, and then tries to verify the k returned potential equations with only $k+1$ calls to a proper optimization oracle. This algorithm, together with a proof that $k+1$ calls are again best possible, is presented in Section 2.4.4. In Section 2.4.5 we give some insight in details of the software implementation.

2.4.1 A Basic Scheme

We begin with a simple iterative scheme. It maintains a set $S \subseteq P$ of affinely independent points (initialized with a single point s_0) and a non-redundant system $Cx = d$ of equations valid for aff.hull (P) (initially empty), and either enlarges the set S by one point or extends the system of equations by an equation until aff.hull $(S) = \{x \in \mathbb{R}^n \mid Cx = d\}$ holds.

For dimension reasons this process needs n iterations.

Finding an equation $\langle c, x\rangle = \delta$ valid for aff.hull(P) means that c is orthogonal to aff.hull(P), which in turn means that

$$(\delta :=) \max_{x\in P} \langle c, x\rangle = \min_{x\in P} \langle c, x\rangle$$

holds. Hence, suppose we have a good guess for such a c, two calls of the oracle suffice to verify that $\langle c, x\rangle = \delta$ is valid. In order to make progress we require that c is linearly independent of already known equation normals.

The next idea is to restrict c to be orthogonal to the affine hull of the points $S \subseteq P$ known so far. Under this restriction, if

$$\max_{x\in P} \langle c, x\rangle > \min_{x\in P} \langle c, x\rangle$$

holds, i.e., we could not find an equation, then the maximizer or the minimizer is affinely independent of S. Hence, in both cases we make progress. This obviously leads to at most $2n$ oracle calls plus one initial call to find s_0.

Algorithm 2.4.1: Affine Hull Basic Scheme

Input: Optimization oracle $\mathfrak{O}$ for a polyhedron $P \subseteq \mathbb{R}^n$.
Output: An affine basis S of aff.hull(P), a non-redundant system $Cx = d$ with aff.hull$(P) = \{x \in \mathbb{R}^n \mid Cx = d\}$.

if $\mathfrak{O}(\mathbb{O})$.val $= -\infty$ **then return** $(\emptyset, \langle \mathbb{O}, x\rangle = 1)$.
else Initialize $s_0 := \mathfrak{O}(\mathbb{O})$.point, $S := \{s_0\}$, $Cx = d$ empty.
while $|S| + |\text{rows}(C)| < n + 1$ **do**
  Find $c \in \mathbb{Q}^n \setminus \{\mathbb{O}\}$, $c \perp (S - s_0)$ and lin. indep. of rows(C).
  if $\mathfrak{O}(c)$.val $= +\infty$ **then** $s^+ := s_0 + \mathfrak{O}(c)$.dir.
  else $s^+ := \mathfrak{O}(c)$.point.
  if $\mathfrak{O}(-c)$.val $= +\infty$ **then** $s^- := s_0 + \mathfrak{O}(-c)$.dir.
  else $s^- := \mathfrak{O}(-c)$.point.
  if $\langle c, s^+\rangle = \langle c, s^-\rangle$ **then** extend $Cx = d$ with $\langle c, x\rangle = \langle c, s^+\rangle$.
  else if $\langle c, s^+\rangle > \langle c, s_0\rangle$ **then** $S := S \cup \{s^+\}$.
  else $S := S \cup \{s^-\}$.
end
return $(S, Cx = d)$.

We will not prove correctness of Algorithm 2.4.1, but do so for a refined version in a moment. We will also present missing details like how to determine the vector c in Step 4. Several other extensions are also handled that add capabilities or improve the numerical behavior.

2.4.2 The Algorithm

Since we consider oracle calls as rather expensive operations we will often cache their results, i.e., have sets of known points and known unbounded directions available. For simplicity, Algorithm 2.4.1 hides the directions by just adding them on top of a known point and considering the result as the new point. In the more elaborate algorithm we want to handle unbounded directions explicitly. The orthogonal complement of the affine hull $\text{aff.hull}(S) + \text{lin.hull}(R)$ of $S + R$ for a non-empty set S of points and a set R of unbounded directions is clearly the intersection of $\text{aff.hull}(S)^\perp$ and $\text{lin.hull}(R)^\perp$, and hence corresponds to the set of all $c \in \mathbb{R}^n$ for which a $\delta \in \mathbb{R}$ with

$$\langle s, c\rangle - \delta = 0 \quad \text{for all } s \in S \qquad \text{and} \qquad \langle r, c\rangle = 0 \quad \text{for all } r \in R \tag{2.1}$$

exists. The $(|S| + |R|) \times (n+1)$-coefficient-matrix of (2.1) as a linear system of equations in variables (c, δ) is subsequently denoted by $A^{(S,R)}$.

The basic scheme needs $2n + 1$ oracle calls to determine the affine hull, and the following algorithm improves this to $2n$. Although this is only a tiny improvement, it closes the gap to a lower bound of $2n$ that we prove in Section 2.4.3. The improvement works as follows:

The set S is not initialized using an oracle call, but is empty at the beginning. When the oracle yields a feasible point for the first time by maximizing a direction c, the algorithm calls it a second time to maximize $-c$, ensuring that $c \neq \mathbb{O}$ holds. This second call yields a second piece of information: An unbounded direction, another point, or a valid equation. This is in contrast to the later iterations, which only result in one piece of information for the cost of two oracle calls in the worst case.

An aspect left open in the basic scheme is the computation of the vector c. The requirement that c is orthogonal to $\text{aff.hull}(P)$ is satisfied by choosing $\binom{c}{\delta} \in \ker(A^{(S,R)})$. For the second property one may be tempted to model c's linear independence of C's rows by orthogonality, that is,

to add the corresponding rows to $A^{(S,R)}$ and then search for an element in the null space. This would certainly yield a correct algorithm, but computational experiments suggest that this has undesirable numerical consequences: First, the underlying linear algebra has to handle rational numbers with much higher encoding lengths that can increase the running time dramatically. Second, the normals of the final set of equations then constitute an orthonormal basis. For several applications such a requirement is not natural, and it is hard to post-process the equations to make them easier to read (see Section 2.8). Instead of this direct approach, the following algorithm iterates over a set of basis elements of $A^{(S,R)}$'s null space and checks every candidate direction c for linear dependence of C's rows. For this it maintains a column basis $B \subseteq [n+1]$, an LU-factorization[3] of $A^{(S,R)}_{*,B}$ and a list $\mathbb{L} \subseteq [n+1] \setminus B$ of indices that induce candidate directions. Since adding an equation does not change $A^{(S,R)}$, the single loop of Algorithm 2.4.1 is split into two loops, where the inner loop iterates over the candidate directions, and the candidate list is reset in every iteration of the outer loop. This yields Algorithm 2.4.2.

Lemma 2.4.1. *Given an optimization oracle for an empty polyhedron, Algorithm 2.4.2 works correctly and needs one oracle call.*

Proof. Clearly, both loop conditions are satisfied for the first time, and in the first iteration of the inner loop the conditions of Step 7 are satisfied since $S = \emptyset$ and $\mathbb{L} = [n+1]$ hold. The solution of the system in Step 10 yields $c = e^{(j)}$ and $\delta = 0$ due to $j \neq n+1$. Since $Cx = d$ is empty, the oracle is called with objective c in Step 12 and returns the correct answer using precisely one oracle call. □

We will make use of the following well-known proposition in order to show regularity of the basis matrices involved [36].

Proposition 2.4.2 (Schur Complement)**.** *Let $A \in \mathbb{R}^{n\times n}$ be regular, $b, c \in \mathbb{R}^n$ and $\delta \in \mathbb{R}$. Then $\begin{pmatrix} A & b \\ c^\intercal & \delta \end{pmatrix}$ is regular if and only if $\delta - c^\intercal A^{-1} b \neq 0$ holds.*

[3] Writing $A = L \cdot U$ for lower-triangular matrix L and upper-triangular matrix U is an efficient alternative to computing A's inverse.

Algorithm 2.4.2: Affine Hull

Input: Optimization oracle $\mathfrak{O}$ for a polyhedron $P \subseteq \mathbb{R}^n$.

Output: A tuple $(S, R, B, Cx = d)$, where $S \subseteq P$ are points and $R \subseteq \operatorname{recc}(P)$ are directions such that $A^{(S,R)}$ has full row-rank, B is a column basis of $A^{(S,R)}$ and the system $Cx = d$ is irredundant with $\operatorname{aff.hull}(P) = \operatorname{aff.hull}(S) + \operatorname{lin.hull}(R) = \{x \in \mathbb{R}^n \mid Cx = d\}$.

1 Initialize $S := \emptyset$, $R := \emptyset$, $Cx = d$ empty, $B := \emptyset$.
2 **while** $|S| + |R| + |\operatorname{rows}(C)| < n + 1$ **do**
3 Update LU-factorization of $A^{(S,R)}_{*,B}$.
4 $\mathbb{L} := [n+1] \setminus B$.
5 **while** $\mathbb{L} \neq \emptyset$ **do**
6 **if** $S \neq \emptyset$ **then** let $j \in \mathbb{L}$.
7 **else if** $S = \emptyset$ **and** $\mathbb{L} \neq \{n+1\}$ **then** let $j \in \mathbb{L} \setminus \{n+1\}$.
8 **else return** $(\{\mathfrak{O}(\mathbb{O}).\text{point}\}, R, B \cup \{n+1\}, Cx = d)$.
9 Set $\mathbb{L} := \mathbb{L} \setminus \{j\}$.
10 Solve $A^{(S,R)}\binom{c}{\delta} = \mathbb{O}, (c^\intercal, \delta)_{[n+1]\setminus B} = \mathbb{e}^{(j)}$.
11 **if** $c \in \operatorname{lin.hull}(\operatorname{rows}(C))$ **then** go to Step 5.
12 **if** $\mathfrak{O}(c).\text{val} = -\infty$ **then return** $(\emptyset, \emptyset, \emptyset, \langle \mathbb{O}, x \rangle = 1)$.
13 **if** $\mathfrak{O}(c).\text{val} = +\infty$ **then**
14 $R := R \cup \{\mathfrak{O}(c).\text{dir}\}$, $B := B \cup \{j\}$, go to Step 2.
15 **else if** $S = \emptyset$ **then**
16 $S := S \cup \{\mathfrak{O}(c).\text{point}\}$, $B := B \cup \{n+1\}$, $\delta := \mathfrak{O}(c).\text{val}$.
17 **else if** $\langle c, \mathfrak{O}(c).\text{point} \rangle > \delta$ **then**
18 $S := S \cup \{\mathfrak{O}(c).\text{point}\}$, $B := B \cup \{j\}$, go to Step 2.
19 **if** $\mathfrak{O}(-c).\text{val} = +\infty$ **then**
20 $R := R \cup \{\mathfrak{O}(-c).\text{dir}\}$, $B := B \cup \{j\}$, go to Step 2.
21 **else if** $\langle c, \mathfrak{O}(-c).\text{point} \rangle < \delta$ **then**
22 $S := S \cup \{\mathfrak{O}(-c).\text{point}\}$, $B := B \cup \{j\}$, go to Step 2.
23 **else**
24 Extend $Cx = d$ with $\langle c, x \rangle = \delta$.
25 **if** Step 16 was just executed **then** go to Step 2.
26 **end**
27 **end**
28 **return** $(S, R, B, Cx = d)$.

Proof. The fact that the first matrix in the multiplication

$$\begin{pmatrix} A^{-1} & \mathbb{O} \\ c^\intercal A^{-1} & 1 \end{pmatrix} \cdot \begin{pmatrix} A & b \\ c^\intercal & \delta \end{pmatrix} = \begin{pmatrix} \mathbb{I} & A^{-1}b \\ \mathbb{O}^\intercal & \delta - c^\intercal A^{-1} b \end{pmatrix}$$

is regular immediately proves the result. □

Lemma 2.4.3. *Given an optimization oracle for a non-empty rational polyhedron $P \subseteq \mathbb{R}^n$, then during the run of Algorithm 2.4.2, the following properties are satisfied:*

(A) $S \subseteq P$ *and* $R \subseteq \operatorname{recc}(P)$.

(B) In Step 10, if $B \neq \emptyset$ *holds, the matrix* $A^{(S,R)}_{*,B}$ *is regular.*

(C) The vector c *computed in Step 10 satisfies* $c \neq \mathbb{O}$.

(D) R *consists of linearly independent vectors.*

(E) If $S \neq \emptyset$ *holds,* $|S| + |R| = \dim(\text{aff.hull}(S) + \text{lin.hull}(R))$ *is satisfied.*

Proof. Property (A) is easy to check: S is initially empty and only extended with results of oracle calls in bounded directions in Steps 16, 18 and 22. The set R is also initially empty and only extended with results of oracle calls in unbounded directions in Steps 14 and 20. In all cases, B is extended, hence $|B| = |S| + |R|$ also holds.

For the remaining properties, we will apply induction on the cardinality of B. For $B = \emptyset$, we also have $S = R = \emptyset$, proving Properties (B), (D) and (E). For Property (C), observe that $c = \mathbb{e}^{(j)}$ and $\delta = 0$ hold for some $j \in [n]$ since $j \neq n + 1$ by Steps 7 and 8.

So assume we are in an iteration of the inner loop, where B will be extended. By induction hypothesis, $A^{(S,R)}_{*,B}$ is regular, and hence the system in Step 10 has a solution (c, δ).

We now prove Property (C), i.e., that $c \neq \mathbb{O}$ holds. Assuming the contrary implies $j = n + 1$, which by Steps 7 and 8 in turn implies that S is not empty. But then $A^{(S,R)}\binom{c}{\delta} = A^{(S,R)}_{*,n+1} = \mathbb{O}$, contradicting $S \neq \emptyset$ since the last column of $A^{(S,R)}$ contains $|S|$ many -1's.

Property (D) is now easily verified: Since $\langle r, c \rangle = 0$ holds for every $r \in R$ by Step 10, a new unbounded direction $\hat{r}$ added in Step 14 or

Step 20 must be linearly independent of R because $\langle \hat{r}, c \rangle \neq 0$ holds as well.

If $S = \emptyset$ holds and the algorithm reaches Step 16, then after this step Property (E) is satisfied since $|R| = \dim(\text{lin.hull}(R))$ holds by Property (D).

We now prove Property (E) if $S \neq \emptyset$ holds, including the case that Step 16 was performed. In this case $\langle c, s \rangle = \gamma$ holds for all $s \in S$ and $\langle c, r \rangle = 0$ holds for all $r \in R$. A new point $\hat{s}$ added in Step 18 or Step 22 must be affinely independent of $\text{aff.hull}(S) + \text{lin.hull}(R)$.

It remains to show Property (B). Iterations in which B is extended twice are handled one by one. Let $B' = B \cup \{j\}$ be the new basis and let S', R' be the new sets of points and unbounded directions, respectively. Note that we either have $S' = S$ and $R' = R \cup \{\hat{r}\}$ or $S' = S \cup \{\hat{s}\}$ and $R' = R$. By induction we can assume that $A^{(S,R)}_{*,B}$ is regular.

We first show regularity of $A^{(S',R')}_{*,B'}$ for the case when $S = \emptyset$ and a point is added in Step 16. $A^{(S',R')}_{*,B'}$ is regular since it arises from $A^{(S,R)}_{*,B}$ by adding a row and a (negative) unit column with the -1 in the new row.

We now show regularity of $A^{(S',R')}_{*,B}$ for the case when $S = \emptyset$ and $n + 1 \notin B$ hold and an unbounded directions $\hat{r}$ is added in Step 14 or Step 20. From $\mathbb{O} = A^{(S,R)}\binom{c}{\delta} = A^{(S,R)}_{*,B} c_B + A^{(S,R)}_{*,j}$ we derive that

$$\hat{r}_j - \hat{r}_B^\intercal A^{(S,R)^{-1}}_{*,B} A^{(S,R)}_{*,j} = \hat{r}_j + \langle \hat{r}_B, c_B \rangle = \langle \hat{r}, c \rangle \neq 0$$

holds, where the last non-equality comes from the fact that $\hat{r}$ is unbounded in direction c or $-c$. Thus, by Proposition 2.4.2, the matrix $A^{(S',R')}_{*,B'}$ is regular.

We continue with the regularity of $A^{(S',R')}_{*,B}$ for the case when $S \neq \emptyset$ and $n + 1 \in B$ hold and an unbounded direction $\hat{r}$ is added in Step 14 or Step 20. From $\mathbb{O} = A^{(S,R)}\binom{c}{\delta} = A^{(S,R)}_{*,B}\binom{c_B}{\delta} + A^{(S,R)}_{*,j}$ we derive that

$$\hat{r}_j - \begin{pmatrix} \hat{r}_B \\ 0 \end{pmatrix}^\intercal A^{(S,R)^{-1}}_{*,B} A^{(S,R)}_{*,j} = \hat{r}_j + \left\langle \begin{pmatrix} \hat{r}_B \\ 0 \end{pmatrix}, \begin{pmatrix} c_B \\ \delta \end{pmatrix} \right\rangle = \langle \hat{r}, c \rangle \neq 0$$

holds, where the last non-equality comes from the fact that $\hat{r}$ is unbounded in direction c or $-c$. Again, by Proposition 2.4.2, the matrix $A^{(S',R')}_{*,B'}$ is regular.

We finally show regularity of $A^{(S',R')}_{*,B}$ for the case when $S \neq \emptyset$ and $n+1 \in B$ hold and a point $\hat{s}$ is added in Step 18 or Step 22. From $\mathbb{O} = A^{(S,R)}\binom{c}{\delta} = A^{(S,R)}_{*,B}\binom{c_B}{\delta} + A^{(S,R)}_{*,j}$ we derive that

$$\hat{s}_j - \begin{pmatrix}\hat{s}_B\\-1\end{pmatrix}^{\intercal} A^{(S,R)\,-1}_{*,B} A^{(S,R)}_{*,j} = \hat{s}_j + \left\langle \begin{pmatrix}\hat{s}_B\\-1\end{pmatrix}, \begin{pmatrix}c_B\\\delta\end{pmatrix} \right\rangle = \langle \hat{s}, c\rangle - \delta \neq 0$$

holds, where the last non-equality comes from the fact that $\hat{s}$ has c-value distinct from δ. Again, by Proposition 2.4.2, the matrix $A^{(S',R')}_{*,B'}$ is regular. □

Lemma 2.4.4. *Given an optimization oracle for a non-empty rational polyhedron $P \subseteq \mathbb{R}^n$, then during the run of Algorithm 2.4.2 the system $Cx = d$ is valid for P, and the rows of C are linearly independent.*

Proof. We prove the lemma by induction on the number of equations in $Cx = d$, being trivially satisfied at the beginning.

Let $Cx = d$ be non-redundant and valid for P, and consider equation $\langle c, x\rangle = \delta$ added in Step 24. Due to Step 11 we have that c is linearly independent from $\operatorname{rows}(C)$, and from Steps 17 and 21 we derive that $\langle c, x\rangle \leq \delta$ and $\langle c, x\rangle \geq \delta$ are valid for P, which concludes the proof. □

Lemma 2.4.5. *Given an optimization oracle defining a non-empty rational polyhedron $P \subseteq \mathbb{R}^n$, then during the run of Algorithm 2.4.2 we have*

$$\operatorname{lin.hull}(\operatorname{rows}([C \mid d])) \subseteq \ker(A^{(S,R)}), \tag{2.2}$$

and equality holds if and only if we have $|S| + |R| + |\operatorname{rows}(C)| = n + 1$.

Proof. Since $Cx = d$ is valid for P by Lemma 2.4.4 and since we have $S \subseteq P$ and $R \subseteq \operatorname{recc}(P)$ by Property (A) of Lemma 2.4.3, the relation (2.2) holds. Then, by linear independence of C's rows (Lemma 2.4.4) and regularity of $A^{(S,R)}_{*,B}$ (Lemma 2.4.3, Property (B)), we obtain the inequality $|\operatorname{rows}(C)| \leq \dim\ker(A^{(S,R)}) = n + 1 - |S| - |R|$. Hence, equality is equivalent to the given condition, which concludes the proof. □

Lemma 2.4.6. *Given an optimization oracle defining a non-empty rational polyhedron $P \subseteq \mathbb{R}^n$, then whenever the inner loop of Algorithm 2.4.2 is left, then the number of oracle calls made so far is bounded from above by the term $2(|S| + |R| + |\operatorname{rows}(C)| - 1)$ if $S \neq \emptyset$ holds, by $|R| + 1$ if we are in Step 8 and by $|R|$ otherwise.*

Proof. We prove the lemma by induction on the number of times Step 13 has been reached at the point of consideration (not necessarily with the condition satisfied). It trivially holds after initialization.

Case 1: $S = \emptyset$.
If Step 14 is executed, then both $|R|$ and the number of oracle calls are increased by 1. If Step 8 is executed, the number of calls is $|R| + 1$ due to the additional call in that step. Otherwise, S is for the first time extended with a point. Let $k := |R|$ denote the number of unbounded directions at that time. In the same iteration of the inner loop another point or a direction or an equation is added to S, R or $Cx = d$, respectively. Hence, when this loop is left we have called the oracle $k + 2$ times and observe $|S| + |R| + |\operatorname{rows}(C)| \geq k + 2$, which proves the claimed bound (using $k \geq 0$).

Case 2: $S \neq \emptyset$.
In every iteration of the inner loop in which the oracle is called, it is called at most twice and S, R or $Cx = d$ are extended, which proves the claimed bound. □

Theorem 2.4.7. *Algorithm 2.4.2 is correct and needs at most* $2n$ *calls to the given optimization oracle for a rational polyhedron* $P \subseteq \mathbb{R}^n$.

Proof. By Lemma 2.4.1 we can assume $P \neq \emptyset$.

We first prove the upper bound on the number of oracle calls. After $2n$ oracle calls $|S| + |R| + |\operatorname{rows}(C)| - 1 \geq n$ holds by Lemma 2.4.6, and thus the condition of the outer loop in Step 2 is violated. Except for Steps 16 and 24 the condition is always checked before the next oracle call. Step 16 is only executed after at most n executions of Step 14 (since then R already contains a basis of $\mathbb{R}^n$), which takes only n oracle calls. Suppose the $(2n)$'th oracle call results in the execution of Step 24. By the above reasoning we have $|S| + |R| + |\operatorname{rows}(C)| = n + 1$ and Lemma 2.4.5 then implies that for every $\binom{c}{\delta} \in \ker(A^{(S,R)})$ the vector c is linearly dependent of lin.hull (C). Hence, Step 11 ensures that no oracle call is triggered before the condition in Step 2 is checked the next time. This concludes the proof of the bound on the number of oracle calls.

We now show that the algorithm terminates. Suppose the contrary, and observe that the inner loop does not run forever since $\mathbb{L}$ is finite and an element is removed in each iteration in Step 9. Furthermore,

Step 13 can only be reached finitely many times since the number of oracle calls is bounded. The only possibility to run forever is thus by having $|S| + |R| + |\operatorname{rows}(C)| < n + 1$ and $c \in \operatorname{lin.hull}(\operatorname{rows}(C))$ for every $\binom{c}{\delta} \in \ker\left(A^{(S,R)}\right)$ satisfied such that the algorithm leaves the inner loop via Step 11. But both conditions contradict each other by Lemma 2.4.5.

We finally show that the algorithm always returns the correct output for $P \neq \emptyset$. If $S = \emptyset$ holds when we return, then it is in Step 8. We must have $Cx = d$ empty since adding an equation in Step 24 happens only if $S \neq \emptyset$ holds. Hence, since $c \neq \mathbb{O}$ holds by Property (C) of Lemma 2.4.3, whenever j was removed from $\mathbb{L}$ in Step 9, an unbounded direction was added to R in Step 13. Thus we have $[n+1] \setminus B = \{n+1\}$, showing $|R| = n$, i.e. $\operatorname{recc}(P) = \mathbb{R}^n$. The last oracle call in Step 8 then yields a feasible point and the returned basis is $[n+1]$.

Otherwise we return due to the violation of the outer loop's condition $|S| + |R| + |\operatorname{rows}(C)| < n+1$, i.e., $(|S|-1) + |R| \leq \dim(P) \leq n - |\operatorname{rows}(C)|$ is satisfied with equality. Hence, the returned objects determine P's affine hull. Furthermore, by Property (B) of Lemma 2.4.3 the returned basis is correct. □

2.4.3 A Lower Bound on the Number of Oracle Calls

The following theorem states that the number of oracle calls in Algorithm 2.4.2 is best possible.

Theorem 2.4.8. *Every algorithm that computes the affine hull of any polyhedron $P \subseteq \mathbb{R}^n$ only by using an optimization oracle for P needs at least $2n$ oracle calls in the worst case, even if the polyhedra are restricted to cones.*

We will soon define the behavior of an oracle such that the algorithm cannot determine the affine hull after calling the oracle only $2n-1$ times. The oracle then implicitly defines a polyhedron P that will be a cone having the property in the following proposition.

Proposition 2.4.9. *Let $A \in \mathbb{R}^{m \times n}$ be such that $\langle A_{i,*}, A_{j,*} \rangle \geq 0$ holds for every $i, j \in [m]$. Then the cone $C = \{x \in \mathbb{R}^n \mid Ax \leq \mathbb{O}\}$ is full-dimensional.*

Proof of Proposition 2.4.9. Let $\langle a, x \rangle = 0$ be an equation valid for C, i.e., $\max\{\langle \pm a, x \rangle \mid x \in C\} = 0$ holds (note that $\mathbb{O} \in C$). We want to prove that $a = \mathbb{O}$ holds showing that $\langle \mathbb{O}, x \rangle = 0$ is the only equation valid for C.

From strong LP duality (Corollary 7.1g in [62]) we obtain that there exist dual multiplier vectors $\lambda, \mu \in \mathbb{R}^m_+$ such that $a^\intercal = \lambda^\intercal A$ and $-a^\intercal = \mu^\intercal A$ hold. Hence, for every $j \in [m]$ we have

$$\left\langle a, A_{j,*} \right\rangle = \sum_{i=1}^{m} \lambda_i \left\langle A_{i,*}, A_{j,*} \right\rangle \geq 0 \quad \text{and} \quad \left\langle -a, A_{j,*} \right\rangle = \sum_{i=1}^{m} \mu_i \left\langle A_{i,*}, A_{j,*} \right\rangle \geq 0,$$

which shows $a \perp \operatorname{rows}(A)$. But since a was combined from A's rows, this implies $a = \mathbb{O}$, which concludes the proof. □

We now turn to the proof of the theorem above.

Proof of Theorem 2.4.8. We define an optimization oracle whose behavior depends on the previous queries of an affine-hull algorithm. The oracle implicitly defines a set of *consistent* polyhedra, i.e., those polyhedra with which the answers are consistent. A correct affine-hull algorithm has to perform queries (at least) until all the consistent polyhedra have the same affine hull. Thus it suffices to prove that after less than $2n$ queries there exist two consistent cones of different dimensions.

Our oracle maintains a set $R \subseteq \mathbb{R}^n$ of directions, which is initially empty. It behaves as follows when called with direction $a \in \mathbb{R}^n$ to be maximized.

- If there is a direction $r \in R$ with $\langle r, a \rangle > 0$, the oracle returns r as an unbounded direction.
- Otherwise, the oracle returns the point $\mathbb{O}$ as the maximum and adds $-a$ to R.

Note that every direction $r' = -a$ that is added to R satisfies $\langle r, r' \rangle \geq 0$ for every previously added direction $r \in R$. Hence, from Proposition 2.4.9 we obtain that the cone $Y := \{y \in \mathbb{R}^n \mid \langle a, y \rangle \geq 0 \text{ for all } a \in R\}$ is full-dimensional.

We denote by $X \subseteq R$ the subset of directions that were returned to the algorithm. By construction of the oracle, a polyhedron $C \subseteq \mathbb{R}^n$ is consistent if and only if $X \subseteq C \subseteq Y$ holds. From this and from the fact that Y is a cone we obtain $\text{conic.hull}(X) \subseteq Y$, i.e., $\text{conic.hull}(X)$ and Y are both consistent.

First, from $\dim(\text{conic.hull}(X)) \leq |X|$ and from the fact that Y is full-dimensional we obtain that the two cones can only have the same affine

hull if $|X| = n$ holds. Second, during the algorithm, the number of oracle calls is at least $|X| + |R| \geq 2|X|$ since for at most one of the two involved sets the cardinality is increased by one in each oracle call. This implies that the number of oracle calls must be at least $2n$ when the algorithm correctly determined the affine hull, which concludes the proof. □

2.4.4 Heuristic Optimization Oracles

A major speed-up can be obtained by using heuristic oracles when computing the affine hull. Clearly, in order for the sets S and R to grow, we only need affine independence, which does not depend on the oracle's ability to produce optimal solutions. Hence, it is a good strategy to run Algorithm 2.4.2 with a heuristic optimization oracle first, and then verify the returned equations $Cx = d$ using the (proper) optimization oracle.

The simplest way to verify k equations $Cx = d$ is to check whether

$$\mathfrak{O}(C_{*,i}).\text{val} = d_i \text{ and } \mathfrak{O}(-C_{*,i}).\text{val} = -d_i$$

holds for every $i \in [k]$ for the cost of $2k$ oracle calls. Since we are optimistic that the heuristic determined the correct equations, we prefer an algorithm for the verification that needs only $k + 1$ oracle calls, but is not able to determine *which* equation is actually invalid.

Algorithm 2.4.3 does precisely this and calls Algorithm 2.4.2 again if the verification fails. The next theorem states that this is correct.

Theorem 2.4.10. *Algorithm 2.4.3 is correct and needs $k + 1$ calls to the given optimization oracle for a rational polyhedron $P \subseteq \mathbb{R}^n$ to verify the validity of k equations for P.*

Proof. If Step 7 is reached, then the output of the algorithm is correct by Theorem 2.4.7.

Otherwise, observe that S, R, and B obtained in Step 1 are valid, i.e., $S \subseteq P$ and $R \subseteq \text{recc}(P)$ hold and $A_B^{(S,R)}$ is regular. It remains to show that $Cx = d$ is valid for P. From Step 4 we obtain that $C_{i,*}x \leq d_i$ is valid for P

Algorithm 2.4.3: Affine Hull with Additional Heuristic Oracle

Input: Optimization Oracle $\mathfrak{O}$ and heuristic optimization oracle $\mathfrak{H}$ for a polyhedron $P \subseteq \mathbb{R}^n$.

Output: A tuple $(S, R, B, Cx = d)$, where S are points and R are unbounded directions such that $A^{(S,R)}$ has full row-rank, B is a column basis of $A^{(S,R)}$, $Cx = d$ is irredundant with $\text{aff.hull}(P) = \text{aff.hull}(S) + \text{lin.hull}(R) = \{x \in \mathbb{R}^n \mid Cx = d\}$.

1 Call Algorithm 2.4.2 with $\mathfrak{H}$ and obtain $(S, R, B, Cx = d)$.
2 Let k be the number of C's rows.
3 **for** $i = 1, 2, \ldots, k$ **do**
4 | **if** $\mathfrak{O}(C_{i,*}).\text{val} \neq d_i$ **then** go to Step 7.
5 **end**
6 **if** $\mathfrak{O}(-\sum_{i=1}^{k} C_{i,*}).\text{val} = -\sum_{i=1}^{k} d_i$ **then return** $(S, R, B, Cx = d)$
7 Call Algorithm 2.4.2 with $\mathfrak{O}$ and obtain $(\hat{S}, \hat{R}, \hat{B}, \hat{C}x = \hat{d})$.
8 **return** $(\hat{S}, \hat{R}, \hat{B}, \hat{C}x = \hat{d})$

for every $i \in [n]$. From Step 7 and Steps 4 for $i \in [n] \setminus \{j\}$ we obtain

$$-C_{j,*}x = \left(-\sum_{i=1}^{n} C_{i,*}\right)x + \sum_{\substack{i=1 \\ i \neq j}}^{n} C_{i,*}x \leq -\sum_{i=1}^{n} d_i + \sum_{\substack{i=1 \\ i \neq j}}^{n} d_i = -d_j,$$

i.e., $C_{j,*}x \geq d_j$ is valid for P for every $j \in [n]$, which concludes the proof. □

The last theorem settles that $k + 1$ calls to the optimization oracle are actually best possible for verification.

Theorem 2.4.11. *Let $\mathfrak{O}$ be an optimization oracle for a non-empty polyhedron $P \subseteq \mathbb{R}^n$, let $C \in \mathbb{Q}^{k \times n}$ and $d \in \mathbb{Q}^k$.*

Every algorithm that verifies the validity of k equations $Cx = d$ (or determines that at least one equation is invalid) for a polyhedron $P \subseteq \mathbb{R}^n$ only by using an optimization oracle for P needs at least $k + 1$ oracle calls.

Proof. We define an optimization oracle whose behavior depends on the previous queries of a verification algorithm, which has the task to

verify the $k := n$ equations $x_i = 0$ for $i \in [n]$. Similar to the proof of Theorem 2.4.8, the oracle implicitly defines *consistent* polyhedra, i.e., those with which the oracle's answers are consistent. We show that after n queries there exist consistent polyhedra P and Q such that P does not satisfy all n equations, but Q does.

For the first n query directions $c_1, \ldots, c_n$, the oracle answers with the origin as the optimal point, and thus reveals to the algorithm that $\langle c_i, x \rangle \leq 0$ is a valid inequality for all $i \in [n]$.

If the c_i ($i \in [n]$) are linearly dependent, then there exists a nonzero vector v with $v \perp c_i$ for every $i \in [n]$. We can now define the polyhedron as $P := \text{conic.hull}(\{v\})$ without contradicting previous answers.

Otherwise, let $T \in \mathbb{Q}^{n \times n}$ be the matrix whose rows are the vectors $c_i^\intercal$. We claim that $v := -T^{-1}\mathbb{1}_n$ satisfies all inequalities $\langle c_i, x \rangle \leq 0$. To see this, observe that

$$\langle c_i, v \rangle = -\left\langle c_i, T^{-1}\mathbb{1}_n \right\rangle = -\left\langle e^{(i)}, \mathbb{1}_n \right\rangle \leq 0$$

holds for all $i \in [n]$. Thus we can again define the polyhedron as $P := \text{conic.hull}(\{v\})$ without contradicting previous answers.

In both cases the polyhedra P and $Q := \{0\}$ are consistent. Since P does not satisfy all the given equations (note that $\dim P = 1$ holds), but Q does, the verification algorithm can not correctly terminate after the first n queries. □

2.4.5 Implementation Details

We implemented Algorithms 2.4.2 and 2.4.3 and enhanced them in several ways in order to improve the performance.

Reusing Known Data. Having in mind MIP solvers as optimization oracles, the first improvement idea is certainly to use more of its output: Usually, a MIP solver returns more than one (not necessarily optimal) solution. We thus added the capability to the algorithm to check a set of cached points or unbounded directions before actually calling the oracle.

If the underlying MIP is known explicitly we can also extract the given equations and pass them to the algorithm, which clearly saves iterations and hence oracle calls.

Similarly, if we computed the affine hull of a polyhedron P we may afterwards compute the affine hull of one of its faces. In this case we reuse all valid equations and fill the cache of points and directions with those feasible for the face.

Precomputing With Floating Point Arithmetic. Computational evidence shows that for larger dimensions the linear algebra part of the algorithm can be quite time-consuming due to exact arithmetic. There are several steps in Algorithm 2.4.2 in which we could improve the speed by precomputing certain data with floating-point arithmetic followed by an exact computation of the interesting parts.

A first application of this technique is the mentioned cache of points and unbounded directions. We are given a set $S' \subseteq \mathbb{Q}^n$ of points, a direction vector $c \in \mathbb{Q}^n$ and a number $\delta \in \mathbb{Q}$, defining a hyperplane $H = \{x \in \mathbb{R}^n \mid \langle c, x\rangle = \delta\}$. The task is to find out whether there exists a point $s' \in S' \setminus H$. For this we first sort S' by increasing sparsity and compute the approximate scalar products in that order using the floating-point representations $\tilde{c}$ of c and $\tilde{s'}$ of s' for all $s' \in S'$. If an approximate scalar product satisfies $|\langle \tilde{c}, \tilde{s'}\rangle - \delta| \geq \tau$ for some threshold parameter $\tau > 0$, then we recompute the scalar product exactly and, if different from δ, return the current point. Otherwise we increase τ by a factor of 10, that is, we do not trust floating-point results in this order of magnitude anymore. Sorting S' prior to searching allows us to stop the search as soon as we find a suitable point since all points that we would inspect later have no better sparsity. If we do not succeed with this strategy, we also try the point with the largest approximate distance to H. Of course it may happen that there exists a point $s' \in S' \setminus H$, but our algorithm is unable to find it. The algorithm for unbounded directions is very similar and we refer to the source code.

A second application is the selection of the index j in Steps 6 and 7 of Algorithm 2.4.2. Different indices j yield different directions c and, once again, we prefer sparse vectors for different reasons: On the one hand, the oracle may be faster to maximize a sparse objective vector, e.g., if c is a unit vector then the task is to check whether a variable bound is tight. On the other hand, some of the directions may end up as equation normals, and a user is probably more interested in a sparse set of equations valid for P.

Since the computation of a single direction can be already be very

expensive (see Figure 2.9 in Section 2.9.1) we again compute *all* directions using floating-point arithmetic and (approximately) determine their sparsity. For this we declare a number with absolute value less than 10^{-7} as zero[4].

[4]This value was determined in order to get a good approximation of the true sparsity.

INVESTIGATING POLYHEDRA BY ORACLES:

2.5 Computing Facets

This section is dedicated to the problem of finding facets of a polyhedron P, again specified by means of an optimization oracle $\mathfrak{O}$. We first describe the main idea from a geometric point of view and present an equivalent model that is more attractive from a computational standpoint. We only sketch how the oracle is actually used to solve the model, since we need to solve a similar one for a related problem, and hence discuss the details in Section 2.7, dedicated to this topic.

After establishing the correctness of the model we discuss a technical problem that arises, suggest how to resolve it and finally consider details concerning the actual derivation of the facets.

2.5.1 Polarity and Target Cuts

Since we want to start with a geometric intuition, we consider a full-dimensional polyhedron $P \subseteq \mathbb{R}^n$ that contains the origin $\mathbb{O}$ in its interior. The *polar* P^* of P is defined as the set

$$P^* := \{y \in \mathbb{R}^n \mid \langle y, x\rangle \leq 1 \text{ for all } x \in P\}.$$

From basic convex geometry we know several basic properties that will turn out to be very useful for us.

Proposition 2.5.1 (Theorem 9.1 and Corollary 9.1a in [62])**.** *Let $P \subseteq \mathbb{R}^n$ be a full-dimensional pointed polytope with $\mathbb{O} \in \operatorname{int}(P)$. Then*

(i) P^* is a again a full-dimensional pointed polytope,

(ii) $\langle x, y \rangle \leq 1$ *is valid for all* $y \in P^*$ *if and only if* $x \in P$ *holds, and*

(iii) $\langle y, x \rangle \leq 1$ *induces a facet of* P *if and only if* y *is a vertex of* P^*.

Since we are interested in facets of P, Proposition 2.5.1 (iii) tells us that we want to find vertices of P^*. In order to find a specific facet, namely one that separates a point $\hat{x} \in \mathbb{R}^n$ from P, we can solve the following problem.

$$\max \langle \hat{x}, y \rangle \tag{2.3}$$
$$\text{s.t.} \quad \langle x, y \rangle \leq 1 \qquad \text{for all } x \in P \tag{2.4}$$
$$y \in \mathbb{R}^n \tag{2.5}$$

Note that formally this is not an LP since in general it contains infinitely many constraints. It is not too hard to see that we can restrict Constraint (2.4) to those $x \in P$ that are vertices. By Proposition 2.5.1 (iii) a *vertex solution* $\hat{y}$ of Problem (2.3)–(2.5) corresponds to a facet of P. Furthermore, if the maximum objective is larger than 1, i.e., $\langle \hat{x}, \hat{y} \rangle > 1$ holds, then the facet-defining inequality $\langle \hat{y}, x \rangle \leq 1$ of P is violated by $\hat{x}$. On the other hand, if the maximum objective is at most 1, that is, $\langle \hat{x}, y \rangle \leq 1$ holds for all $y \in P^*$, then again by Proposition 2.5.1 (iii) there is no facet-defining inequality (and hence no valid inequality at all) that separates $\hat{x}$ from P.

Using Proposition 2.5.1 (ii) we can show that the separation problem for Inequalities (2.4) is essentially the optimization problem over P, which we can solve by means of our optimization oracle. More precisely, if we have a partial outer description of LP (2.3)–(2.5), then we can decide by one call to the optimization oracle for P whether the current solution $\hat{y}$ is feasible, obtaining a violated inequality if this is not the case. The details of this procedure including the exact usage of our optimization oracle are discussed in the dedicated Section 2.7.

Note that we do not require the oracle to return *vertex solutions*, since our (partial) outer description of P^* is allowed to contain non-facet-defining inequalities, too. On the other hand, it is necessary that we compute a *vertex solution* $\hat{y}$ as only those correspond to facets of P.

If P does not contain $\mathbb{O}$ in its interior we can simply shift it by some

vector $o \in \operatorname{int}(P)$, which yields the following LP

$$\max \langle \hat{x} - o, y \rangle \tag{2.6}$$
$$\text{s.t.} \quad \langle s - o, y \rangle \leq 1 \qquad \text{for all } s \in S \tag{2.7}$$
$$y \in \mathbb{R}^n, \tag{2.8}$$

where $S \subseteq P$ is a (finite) superset of P's vertices.

The ideas we just introduced are not new. To the best of our knowledge, Applegate, Bixby, Chvátal and Cook [3] were the first to develop cutting plane methods based on optimization oracles. Their *Local Cut* procedure uses a different objective function, which does not necessarily yield facet-defining inequalities. Hence, they describe a tilting procedure that allows them to generate facets for small subproblems of the traveling salesman problem (TSP), which are in turn lifted to inequalities valid for the TSP-polytope. Later, Buchheim, Liers and Oswald [15] revisited this idea and came up with the *Target Cuts* for which they used a variant of LP (2.6)–(2.8).

2.5.2 An Equivalent Model

The LP (2.6)–(2.8) in principle works for our purposes, except that we restricted ourself to full-dimensional pointed polytopes. Unfortunately, interior points of (combinatorial) polyhedra are typically very dense vectors, which implies that the matrix of the LP's inequality system is a very dense matrix in general. Since the running time of state-of-the-art LP solvers critically depends on the number of nonzeros of the problem matrix (see, e.g., [66] and [65]), it is obvious that this LP is numerically not very attractive.

Luckily, there is a transformation, communicated by Marc Pfetsch, that yields an equivalent LP in which the interior point o appears in only one inequality.

$$\max \quad \langle \hat{x}, a \rangle - \beta \tag{2.9}$$
$$\text{s.t.} \quad \langle s, a \rangle - \beta \leq 0 \qquad \text{for all } s \in S \tag{2.10}$$
$$\langle r, a \rangle \leq 0 \qquad \text{for all } r \in R \tag{2.11}$$
$$\langle \hat{x} - o, a \rangle \leq 1 \tag{2.12}$$
$$a \in \mathbb{R}^n, \quad \beta \in \mathbb{R} \tag{2.13}$$

We will eventually prove that this model can be used for facet-separation, also in case of lower-dimensional (potentially unbounded) polyhedra. Our proof is independent from the results on polar polytopes discussed above. Since both models might be used for facet-separation one may ask whether they compute the same facets. The next theorem states that both LPs are indeed equivalent in the sense that the facets corresponding to optimal solutions of the respective LPs are the same.

Theorem 2.5.2. *Let $S \subseteq \mathbb{R}^n$ be finite sets such that $P = \text{conv.hull}(S)$ is full-dimensional. Let $\hat{x} \in \mathbb{R}^n \setminus P$ and $o \in \text{int}(P)$. Consider an inequality $\langle a, x\rangle \leq \beta$ that is valid for P but violated by $\hat{x}$, and the unique vector y such that $\langle y, x - o\rangle \leq 1$ is equal to this inequality up to multiplication with a positive scalar. Then y is an optimal solution of LP* (2.6)–(2.8) *if and only if (a, β) is an optimal solution of LP* (2.9)–(2.13).

Proof. For the first LP we know that it is always bounded (see Proposition 2.5.1 (i)). For the second LP we write $o = \sum_{s\in S} \lambda_s s$ with $\sum_{s\in S} \lambda_s = 1$ and $\lambda \in \mathbb{R}^S_+$, and derive an upper bound on the objective value

$$\langle \hat{x}, a\rangle - \beta \leq 1 + \langle o, a\rangle - \beta = \sum_{s\in S} \lambda_s \langle s, a\rangle + 1 - \beta = \sum_{s\in S} \lambda_s \underbrace{(\langle s, a\rangle - \beta)}_{\leq 0} + 1 \leq 1,$$

where the first inequality holds due to (2.12), and the second due to (2.10).

We now relate the inequalities $\langle a, x\rangle \leq \beta$ valid for P to the vectors $y \in (P-o)^*$. Given a and β, we can set $y = \pi(a, \beta) := a/(\beta - \langle o, a\rangle$. Then the normal vectors are positive scalar multiples of each other since $\langle a, o\rangle < \beta$ holds (due to $o \in \text{int}(P)$). This scaled version of $\langle a, x\rangle \leq \beta$ has right-hand side $\beta/(\beta - \langle o, a\rangle) = 1 + \langle o, a\rangle / (\beta - \langle o, a\rangle) = 1 + \langle o, y\rangle$, which agrees with that of $\langle x - o, y\rangle \leq 1$.

Although it is rather clear from the definitions of the LPs and from the observation above that $y = \pi(a, \beta)$ is feasible for LP (2.6)–(2.8), we still prove it formally. Assume (a, β) is an optimal solution of LP (2.9)–(2.13). Then (a, β) satisfies (2.12) with equality, since otherwise we could scale it, increasing its positive objective value $\langle \hat{x}, a\rangle - \beta$. From (2.10) we obtain that for all $s \in S$,

$$\langle s - o, y\rangle = \frac{\langle s - o, a\rangle}{\beta - \langle o, a\rangle} = \frac{\langle s, a\rangle - \langle o, a\rangle}{\beta - \langle o, a\rangle} \leq 1$$

holds (Note that $\beta - \langle a, o\rangle$ is positive). We conclude that our constructed $y = \pi(a, \beta)$ is feasible for LP (2.6)–(2.8). The objective value of y is equal to

$$\langle \hat{x} - o, y\rangle = \frac{\langle \hat{x} - o, a\rangle}{\beta - \langle o, a\rangle} = \frac{\langle \hat{x} - o, a\rangle}{\langle \hat{x} - o, a\rangle - \langle \hat{x}, a\rangle + \beta} = \frac{1}{1 - (\langle \hat{x}, a\rangle - \beta)},$$

which is strictly monotone in (a, β)'s objective (using the fact that the denominator is positive).

This already proves that π maps non-optimal solutions of LP (2.9)–(2.13) that satisfy (2.12) with equality to non-optimal solutions of the LP (2.6)–(2.8). It could still be that there exist solutions to the latter LP that are better, but are no images (w.r.t. π) of feasible vectors (a, β).

To rule out this case, consider a vector y that satisfies (2.7)–(2.8) and also has a positive objective $\langle \hat{x} - o, y\rangle$. It suffices to prove that the vector (a, β) with $a = \frac{y}{\langle \hat{x}-o, y\rangle}$ and $\beta = \frac{\langle o,y\rangle+1}{\langle \hat{x}-o,y\rangle}$ is feasible for LP (2.9)–(2.13), since it satisfies

$$\pi(a, \beta) = \frac{a}{\beta - \langle o, a\rangle} = \frac{\frac{y}{\langle \hat{x}-o,y\rangle}}{\frac{\langle o,y\rangle+1}{\langle \hat{x}-o,y\rangle} - \frac{\langle o,y\rangle}{\langle \hat{x}-o,y\rangle}} = \frac{y}{\langle o, y\rangle + 1 - \langle o, y\rangle} = y$$

From (2.7) we obtain that for all $s \in S$,

$$\langle s, a\rangle - \beta = \left\langle s, \frac{y}{\langle \hat{x} - o, y\rangle}\right\rangle - \frac{\langle o, y\rangle + 1}{\langle \hat{x} - o, y\rangle} = \frac{\langle s - o, y\rangle - 1}{\langle \hat{x} - o, y\rangle} \leq 0$$

holds. Last but not least,

$$\langle \hat{x} - o, a\rangle = \frac{\langle \hat{x} - o, y\rangle}{\langle \hat{x} - o, y\rangle} = 1$$

shows that (2.12) is satisfied with equality, which establishes that π defines a bijection between the optimal faces of the two LPs, and concludes the proof. □

Theorem 2.5.2 shows that it does not matter which of the two problems we actually solve, that is, none of the formulations prefers certain inequalities more than the other. Since the Inequalities (2.10) and (2.11)

occur in another problem as well (see Section 2.6.1), we describe how to actually separate this class of (typically exponentially many) inequalities in a dedicated section, namely Section 2.7. Before we turn to the proof that vertex solutions of LP (2.9)–(2.13) yield facets (or equations) in Section 2.5.4 we briefly discuss technical details for actually obtaining vertex solutions.

2.5.3 Extended Basic Solutions

Students in optimization courses are usually taught LPs in some standard form, e.g., $\max \langle c, x \rangle$ s.t. $Ax = b$, $x \in \mathbb{R}^n_+$ with $A \in \mathbb{R}^{m \times n}$ of full row rank, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$. In this form, a *basis* is a set $B \subseteq [n]$ for which $A_{*,B}$ is a regular submatrix. The corresponding *basic solution* $x \in \mathbb{R}^n_+$ is defined by setting $x_i := 0$ for all $i \in [n] \setminus B$ (*nonbasic variables*) and $x_B := A^{-1}_{*,B} b$. If it is feasible, then it defines a vertex of the polyhedron of feasible solutions.

In practice though, LPs are not brought into standard form, but instead every variable has a lower bound $\ell_i \in \mathbb{R} \cup \{-\infty\}$ and an upper bound $u_i \in \mathbb{R} \cup \{\infty\}$. Hence, for nonbasic variables we must distinguish at which bound the variable is fixed. Additionally, if $\ell_i = -\infty$ and $u_i = \infty$ holds, the common implementations[5] implicitly split x_i into two nonnegative variables $x_i = x_i^+ - x_i^-$. This lifting allows both new variables to be fixed at their lower bound 0, which in general does not correspond to a basic solution in the original space. We call these solutions *extended basic solutions* and denote the basic information by a partitioning of $[n]$ into L, U, Z, B with the following meaning for variable x_i:

- $i \in Z$: $x_i = 0$, i.e., x_i^+ and x_i^- are nonbasic. ("*fixed at zero*")
- $i \in L$: $x_i = \ell_i$ ("*fixed at lower bound*")
- $i \in U$: $x_i = u_i$ ("*fixed at upper bound*")
- $i \in B$: x_i is uniquely defined by requiring $Ax = b$. ("*basic*")

Due to the artificial case of variables that are fixed at zero, an extended basic solution need not be a vertex, a property that is essential for the

[5] For instance IBM ILOG CPLEX Optimization Studio, Gurobi Optimizer, FICO Xpress-Optimizer and SoPlex.

derivation of facet-defining inequalities. Fortunately, SoPlex, the LP solver we use, has the following property[6]:

Assumption 2.5.3. *During the run of the LP solver, the number of variables that are fixed at zero never increases.*

Thus, if we initialize the solver with a basis having sufficiently many basic variables, then the returned optimal extended basic solution will be a vertex. Having this behavior in mind we can continue with our actual usage of LP (2.9)–(2.13) for facet-generation.

2.5.4 Extracting Inequalities and Equations

In this section we consider an arbitrary polyhedron $P \subseteq \mathbb{R}^n$, which may be lower-dimensional and need not be pointed. Let d be its dimension. We assume that we know sets $\hat{S} \subseteq P$ of points and $\hat{R} \subseteq \operatorname{recc}(P)$ directions with $|\hat{S}| + |\hat{R}| = d + 1$ such that $\operatorname{conv.hull}(\hat{S}) + \operatorname{conic.hull}(\hat{R})$ is already a d-dimensional polyhedron. Such sets can be computed by Algorithm 2.4.2, together with a set $I \subseteq [n+1]$ of $d+1$ variables such that the submatrix of our LP indexed by columns I and by rows corresponding either to (2.10) for all $s \in \hat{S}$ or to (2.11) for all $r \in \hat{R}$ is a regular $(d+1) \times (d+1)$ matrix.

We will warm-start our LP solver with this as a basis matrix, that is, only $(n+1)-(d+1) = n-d$ variables are fixed at zero. The corresponding extended basic solution corresponds to $a = \mathbf{0}$ and $\beta = 0$, and is thus primal feasible.

Assumption 2.5.3 guarantees that all subsequent bases will have at most $n - d$ many variables fixed at zero. Our next theorem then ensures that we can either extract a facet-defining inequality or an equation from the optimal basis.

Theorem 2.5.4. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron and let d be its dimension. Let $(a, \beta) \in \mathbb{R}^n \times \mathbb{R}$ be an extended basic feasible solution of LP* (2.9)–(2.13) *whose extended basis has at most $n - d$ variables fixed at zero. If $\langle a, o \rangle < \beta$ holds, then $\langle a, x \rangle \leq \beta$ induces a facet, and otherwise $\langle a, x \rangle = \beta$ is valid for P.*

Proof. We consider the LP in standard form, i.e., after adding slack variables for all inequalities. The associated constraint matrix has precisely

[6] Personal communication with Ambros Gleixner.

$m' := |S| + |R| + 1$ rows, $n' := n + 1 + |S| + |R| + 1$ columns and has full row rank due to the identity submatrix of size m'.

Let $S' \subseteq S$ and $R' \subseteq R$ index the Inequalities 2.10 and (2.11) whose corresponding slack variables are nonbasic. Since at most $n - d$ of the $n + 1$ original variables are fixed at zero and since these variables have no lower or upper bounds, there must be at least $d + 1$ basic variables among the original ones. Thus at most $m' - (d+1)$ of the remaining (slack) variables can be basic, i.e., at least $d + 1$ of them are nonbasic. Except the slack variable for Inequality (2.12), all slack variables correspond to Inequalities (2.10) and (2.11). Hence, $|S'| + |R'| \geq d$ holds. These points and directions induce a polyhedron conv.hull(S') + conic.hull$(R') \subseteq P$ of dimension k with $d - 1 \leq k \leq d = \dim P$. Since by definition of S' and R', all $s \in S'$ satisfy $\langle a, s \rangle = \beta$ and all $r \in R'$ satisfy $\langle a, r \rangle = 0$, this k-dimensional polyhedron is contained in the face F induced by $\langle a, x \rangle \leq \beta$, and hence F is either a facet of P or equal to P.

Finally, we can distinguish between the two cases by checking whether $o \in F$ holds, since a point in the relative interior is not contained in any facet. □

For lower-dimensional polyhedra P, the set of feasible (a, β) is not a pointed polyhedron. We clearly could have made it pointed, e.g., by requiring $a \in$ lin.hull $(P - o)$, also making inequality representations unique up to scaling. This approach though has several disadvantages. First, there exist points[7] in $\mathbb{R}^n \setminus$ aff.hull (P) that are not cut off by such inequalities, leading to an infeasible LP. Second, requiring $a \in$ lin.hull $(P - o)$ may be a canonical representative from a geometric point of view, but the typical inequalities known for combinatorial polyhedra are usually not of this type. Third, having a certain amount of coefficients fixed to zero increases sparsity of the inequality, a property that is interesting for the user anyway.

It is interesting to analyze which facets are actually selected by the outlined procedure. Consider an inequality $\langle c, x \rangle \leq \gamma$ valid for P with $\langle c, \hat{x} - o \rangle > 0$. It is clearly represented in LP (2.9)–(2.13) as (a, β) with $a = \lambda c$ and $\beta = \lambda\gamma$ for $\lambda > 0$ such that $\langle a, \hat{x} - o \rangle = 1$ holds, i.e., by setting

[7] For instance the point o, slightly moved outside of aff.hull (P).

$\lambda = 1/\langle c, \hat{x} - o\rangle$. Hence, its objective value is equal to

$$\lambda \cdot (\langle \hat{x}, c\rangle - \gamma) = \frac{\langle c, \hat{x}\rangle - \gamma}{\langle c, \hat{x} - o\rangle} = \frac{\langle c, \hat{x}\rangle - \langle c, \overline{x}\rangle}{\langle c, \hat{x}\rangle - \langle c, o\rangle} = \frac{\|\hat{x} - \overline{x}\|}{\|\hat{x} - o\|}, \tag{2.14}$$

where $\overline{x}$ is the unique point on the line connecting $\hat{x}$ and o that also satisfies $\langle c, \overline{x}\rangle = \gamma$. Hence, LP (2.9)–(2.13) yields only facets of P that are hit first when going straight from o to $\hat{x}$.

We can make use of this observation when we are given a point $\tilde{x} \in \text{relbd}(P)$ and would like to find a facet $\tilde{F}$ that contains $\tilde{x}$ (see Section 2.6.5 for an application). Solving the LP with $\hat{x} := \tilde{x}$ may result in an equation (with zero objective, since it is not violated). If we instead use $\hat{x} := 2\tilde{x} - o$, then the mentioned unique point on the line connecting $\hat{x}$ and o is clearly $\overline{x} = \tilde{x}$ because of $\tilde{x} \in \text{relbd}(P)$ and $o \in \text{relint}(P)$. Hence, the same facets correspond to optimal solutions as before, but now have objective 1/2. On the other hand, since from $\tilde{x}, o \in \text{aff.hull}(P)$ we obtain $\hat{x} \in \text{aff.hull}(P)$, all equations valid for P again have violation zero, and are thus not optimal anymore.

2.5.5 Computing Multiple Facets

In the previous subsections we considered the problem to separate a given point $\hat{x}$ from our polyhedron $P \subseteq \mathbb{R}^n$ by a facet-defining inequality. We now discuss an obvious way to compute such points based on an objective vector $c \in \mathbb{R}^n$. For this we consider a relaxation polyhedron $Q \supseteq P$. Note that we do not care for integrality here, that is, it need not be an LP relaxation for a certain integral polyhedron.

We start by optimizing $\langle c, x\rangle$ over $x \in Q$. If the optimum is finite, we obtain an optimum solution $\hat{x}$, which we can either cut off by a facet-defining inequality or observe that $\hat{x} \in P$ holds. In the former case we add the inequality to the description of Q and repeat, and in the latter case we stop our procedure. If the optimum is not bounded, e.g., if we start with $Q = \mathbb{R}^n$, then we obtain an unbounded direction $\hat{r} \in \text{recc}(Q)$. To separate a direction $\hat{r}$ from P we also solve LP (2.9)–(2.13), except that we set β's coefficient in the objective to zero and use for o a point in $\text{recc}(P)$'s relative interior. The proofs that the optimum is always finite and that it is positive if and only if a separating inequality exists are very similar to the ones for separating a point, and thus left to the reader.

Using this strategy we typically obtain several facets until we eventually find a point $\hat{x} \in P$. This iteration number clearly depends on the tightness of the relaxation Q: If Q does not approximate P in a reasonable way, then many iterations are needed, which is not bad by itself, but it happens frequently that the procedure yields many trivial facets before actually computing an interesting one. On the other hand, if the optimum objective value (w.r.t. c) over Q is equal to that of P, then we might not learn anything new. To cope with this problem there are two ways, which can also be combined. One simple trick is to weaken the relaxation by simply relaxing right-hand side vectors. Although in this case IPO may just return the original inequalities, it sometimes happens that a different inequality is obtained. The other approach is to try different objective vectors. In fact it may even require some mathematical work in order to come up with such bad objective vectors, or at least useful parameters to randomly generate them. For instance, in Section 2.10.3 we consider a polyhedral reduction of vertex cover polytopes to tree polytopes that can be used to come up with objective vectors that yield fractional solutions, and hence allow IPO to detect new facets of the latter polytopes.

INVESTIGATING POLYHEDRA BY ORACLES:

2.6 Identifying Vertices, Edges and Other Faces

The goal of this section is to devise practical algorithms that solve the following problem. Given an optimization oracle $\mathfrak{O}$ for a rational polyhedron $P \subseteq \mathbb{R}^n$, as well as finite sets $S \subseteq P$ and $R \subseteq \operatorname{recc}(P)$, decide whether $F := \operatorname{conv.hull}(S) + \operatorname{conic.hull}(R)$ is a face of P. A related task is to ask for the smallest face $\hat{F}$ containing F, and particular cases are to check whether a given point $s \in P$ defines a vertex and to check whether two vertices s_1, s_2 of P are adjacent (i.e., $\operatorname{conv.hull}(s_1, s_2)$ defines an edge).

All these problems can be solved in time polynomial in n, the encoding length of P and the running-time of $\mathfrak{O}$ using the ellipsoid method [62]. Schrijver presents algorithms for related problems, but – although they run in polynomial-time – they make use of the equivalence of optimization and separation multiple times in a nested manner, which is impractical even if one replaces the ellipsoid method by some practically efficient method.

Let, throughout this section, $S, R \subseteq \mathbb{Q}^n$ be given finite sets, and $\mathfrak{O}$ be given. We will always assume that $\emptyset \neq S \subseteq P$ and $R \subseteq \operatorname{recc}(P)$ hold.

2.6.1 The Smallest Containing Face

We begin with an algorithm that determines a vector $\hat{c}$ whose associated optimal face $\hat{F} := \arg\max\{\langle \hat{c}, x\rangle \mid x \in P\}$ is the smallest face containing F.

A first useful observation, stated in the next lemma, is that it suffices

to find the smallest face containing some point from F's relative interior, e.g.,

$$\hat{x} := \frac{1}{|S|} \sum_{s \in S} s + \sum_{r \in R} r. \tag{2.15}$$

Lemma 2.6.1. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron, and $\emptyset \neq S \subseteq P$ and $R \subseteq \operatorname{recc}(P)$ be finite sets. Define $F := \operatorname{conv.hull}(S) + \operatorname{conic.hull}(R)$ and $\hat{x}$ by* (2.15)*. Then a face of P contains F if and only if it contains $\hat{x}$. In particular, the smallest face of P containing F is the smallest face of P containing $\hat{x}$.*

Proof. Every face of P that contains F clearly also contains $\hat{x} \in F$. It remains to prove that if $\hat{F}$ is a face of P containing $\hat{x}$, then it contains F.

Because $\hat{F}$ is a face of P, there exist $a \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$ such that $\langle a, x \rangle \leq \beta$ is valid for P, and $\hat{F} = \{x \in P \mid \langle a, x \rangle = \beta\}$ holds. In particular, $\langle a, s \rangle \leq \beta$ holds for all $s \in S$ and $\langle a, r \rangle \leq 0$ holds for all $r \in R$. Due to $\hat{x} \in F$ we obtain

$$\beta = \langle a, \hat{x} \rangle = \frac{1}{|S|} \sum_{s \in S} \underbrace{\langle a, s \rangle}_{\leq \beta} + \sum_{r \in R} \underbrace{\langle a, r \rangle}_{\leq 0}$$

and conclude that all $s \in S$ satisfy $\langle a, s \rangle = \beta$ and all $r \in R$ satisfy $\langle a, r \rangle = 0$. It follows that $\langle a, x \rangle = \beta$ is valid for F, which proves $F \subseteq \hat{F}$ and concludes the proof. □

The next theorem states that our desired directions are precisely those in the relative interior of the normal cone of $\hat{x}$.

Theorem 2.6.2. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron and let $\hat{x} \in P$. Then the smallest face containing $\hat{x}$ is equal to the $\hat{c}$-maximum face for some vector $\hat{c}$ if and only if $\hat{c} \in \operatorname{relint}(\operatorname{nml.cone}_{\hat{x}}(P))$ holds.*

Proof. Denote by $\hat{F}$ the smallest face of P containing $\hat{x}$ and define the cone $C := \operatorname{nml.cone}_{\hat{x}}(P)$. Strong LP duality (Corollary 7.1g in [62]) states that $c \in C$ holds if and only if $\hat{x}$ is c-maximum.

For sufficiency, let $\hat{c} \in C$ and suppose the $\hat{c}$-maximum face $\overline{F}$ of P is a strict superset of $\hat{F}$, that is, there exists a point $\overline{x} \in \overline{F} \setminus \hat{F}$. From $\overline{x} \in P$ we get that $\langle \overline{x} - \hat{x}, c \rangle \leq 0$ is valid for C and defines a face G. As $\overline{x}$ is also $\hat{c}$-maximum, we have $\langle \overline{x} - \hat{x}, \hat{c} \rangle = 0$ and hence $\hat{c} \in G$. On the other

hand, since $\overline{F} \neq \hat{F}$ holds, there exists a facet-defining inequality $\langle a, x\rangle \leq \beta$ satisfied with equality by $\hat{F}$ (in particular by $\hat{x}$), but not by $\overline{x}$. This in turn implies $\langle \overline{x} - \hat{x}, a\rangle < 0$, i.e., $a \notin G$. Hence, G is a proper face of C and we obtain $\hat{c} \notin \operatorname{relint}(C)$.

For necessity, let $\hat{c} \in C \setminus \operatorname{relint}(C)$, that is, there exists a proper face G of C induced by the inequality $\langle v, c\rangle \leq 0$ valid for C with $\langle v, \hat{c}\rangle = 0$. This implies that $v \in \operatorname{rad.cone}_{\hat{x}}(P)$, i.e., we can assume v to be scaled such that $\overline{x} := \hat{x} + v \in P$ holds. Since G is a proper face, there exists a direction $a \in C \setminus G$ that is a normal vector of a facet of P containing $\hat{x}$, i.e., for some $\beta \in \mathbb{R}$, $\langle a, x\rangle \leq \beta$ is valid for P, $\langle v, a\rangle < 0$, and $\langle a, \hat{x}\rangle = \beta$ hold. From $\langle v, a\rangle < 0$ we obtain $\langle a, \overline{x}\rangle < \beta$, which proves $\overline{x} \in P \setminus \hat{F}$. Since $\langle \hat{c}, \overline{x}\rangle = \langle \hat{c}, \hat{x}\rangle$ holds, the $\hat{c}$-maximum face of P must be strictly larger than $\hat{F}$. This concludes the proof. □

Luckily, we can compute a relative interior point of a polyhedron if we can optimize over it by invoking Algorithm 2.4.2 from Section 2.4. Note that in the case of a cone, "optimizing" essentially means to compute for a given objective vector an unbounded direction with positive scalar product or return the zero vector as the optimal point. In order to optimize a linear function $\langle v, \cdot\rangle$ over the normal cone of a given point $\hat{x}$ we will use the following LP that is based on the definition of a normal cone.

$$\begin{aligned}
\max \quad & \langle v, c\rangle && && (2.16)\\
\text{s.t.} \quad & \langle s, c\rangle - \delta \leq 0 && \text{for all } s \in \operatorname{vert}(P) && (2.17)\\
& \langle r, c\rangle \leq 0 && \text{for all } r \in \operatorname{ext.rays}(P) && (2.18)\\
& \langle \hat{x}, c\rangle - \delta = 0 && && (2.19)\\
& c \in \mathbb{R}^n, \quad \delta \in \mathbb{R} && && (2.20)
\end{aligned}$$

Note that the projection of the set of feasible (c, δ) onto the c-space is an isomorphism due to Equation (2.19), and hence the relative interior of the feasible region is projected onto the relative interior of the normal cone. Techniques to solve this LP efficiently are described in Section 2.7.

Suppose we know the inequality $\langle \hat{c}, x\rangle \leq \hat{\delta}$ that induces our desired face and we want to determine its dimension (e.g., in order to check whether a given point is a vertex). For this we could turn our optimization oracle for P into one for F (see Section 2.3.3) and compute F's affine

hull. The next proposition states that we can calculate the dimension from that of the normal cone we used to compute $\hat{c}$.

Proposition 2.6.3. *Let $P \subseteq \mathbb{R}^n$ be a polyhedron, let $\hat{x} \in P$, and let $\hat{F}$ be the smallest face of P containing $\hat{x}$. Then* $\operatorname{lin.hull}\left(\hat{F} - \hat{x}\right) = \operatorname{nml.cone}_{\hat{x}}(P)^{\perp}$ *holds and hence the dimension of $\hat{F}$ is equal to $n - \dim\left(\operatorname{nml.cone}_{\hat{x}}(P)\right)$.*

Proof. Let $v \in (\hat{F} - \hat{x})$. Since $\hat{F}$ is the *smallest* face of P containing $\hat{x}$, we have $\hat{x} \in \operatorname{relint}(\hat{F})$, and hence there exists $\lambda > 0$ such that $(\hat{x} - \lambda v) \in \hat{F}$ holds, proving $v, -v \in \operatorname{rad.cone}_{\hat{x}}(P)$, which implies $v \perp \operatorname{nml.cone}_{\hat{x}}(P)$.

Let the equation $\langle v, x\rangle = 0$ be valid for $\operatorname{nml.cone}_{\hat{x}}(P)$. Hence, we have $v \in \operatorname{lineal}\left(\operatorname{rad.cone}_{\hat{x}}(P)\right)$, which proves $(\hat{x} + \lambda v), (\hat{x} - \lambda v) \in P$ for sufficiently small $\lambda > 0$. Let $a \in \mathbb{R}^n$, $\beta \in \mathbb{R}$ be such that $\langle a, x\rangle \leq \beta$ is valid for P, and $\hat{F} = \arg\max\{\langle a, x\rangle \mid x \in P\}$ holds. From $\langle a, \hat{x}\rangle = \beta$, $\langle a, \hat{x} + \lambda v\rangle \leq \beta$ and $\langle a, \hat{x} - \lambda v\rangle \leq \beta$ it follows that $\langle a, v\rangle = 0$ holds, that is, $v \in \operatorname{lin.hull}(\hat{F} - \hat{x})$ holds, which concludes the proof. □

The results of this section, together with Theorem 2.4.7 directly yield the correctness of the following algorithm:

Algorithm 2.6.1: Smallest Containing Face

Input: Optimization oracle $\mathfrak{O}$ for a polyhedron $P \subseteq \mathbb{R}^n$ and $\hat{x} \in P$
Output: A vector $\hat{c}$ whose maximum face $\hat{F}$ of P is the smallest face containing $\hat{x}$, and the dimension $\dim \hat{F}$ of that face.

1 Using $\mathfrak{O}$ and LP (2.16)–(2.20), construct an optimization oracle $\mathfrak{O}_C$ for $C := \operatorname{nml.cone}_{\hat{x}}(P)$ (see Section 2.7).
2 Call Algorithm 2.4.2 with $\mathfrak{O}_C$ and obtain $(S, R, B, Ax = b)$.
3 Compute a relative interior point $\hat{c}$ of C by Equation (2.15).
4 **return** $(\hat{c}, |\operatorname{rows}(A)| - 1)$.

2.6.2 Detecting Vertices

With Proposition 2.6.3 at hand we can easily check if a given point $s \in P$ is a vertex of P. For this we simply compute the dimension of the normal cone at s, which is equal to the ambient dimension n if and only if s is a vertex. The next observation shows that we still have to be careful, though.

Observation 2.6.4. *Let $P \subseteq \mathbb{R}^n$ be a polytope and let $\hat{x} \notin P$. Then*

$$C := \{c \in \mathbb{R}^n \mid \exists \delta \text{ satisfying (2.17), (2.18) and (2.19) }\}$$

is full-dimensional.

Proof. From $\hat{x} \notin P$ we conclude that there exists a separating hyperplane, i.e., there exists $c \in \mathbb{R}^n$ with $\langle c, \hat{x}\rangle > \max\{\langle c, x\rangle \mid x \in P\}$. Since P is bounded, any sufficiently small perturbation $\tilde{c}$ of c has the same property. This concludes the proof since c and all $\tilde{c}$ are contained in C. □

Suppose we accidentally call Algorithm 2.6.1 for a point $\hat{x} \notin P$. Then we would conclude $\hat{x}$ to be a vertex, although it is not even contained in P. Hence we have to ensure $\hat{x} \in P$ beforehand.

2.6.3 Detecting Edges and Extreme Rays

Suppose we are given two vertices s_1 and s_2 of P and want to test whether $F := \text{conv.hull}(s_1, s_2)$ is an edge of P. By Lemma 2.6.1 we just have to determine the dimension of the smallest face containing $\hat{x} := \frac{1}{2}s_1 + \frac{1}{2}s_2$ since this smallest face is 1-dimensional if and only if F is an edge.

The situation is similar for extreme rays: For given vertex $s \in P$ and extreme ray $r \in \text{recc}(P)$ we only have to check whether the smallest face containing $s + r$ is 1-dimensional.

We now discuss how to obtain certificates for the two situations. Suppose the algorithm determines that F is indeed 1-dimensional, that is, the normal cone C of $\hat{x}$ is at least $(n-1)$-dimensional (in fact $\dim C = n-1$ holds since $\langle s_1 - s_2, c\rangle = 0$ is valid for C). The certificate for this fact is a set of $(n-1)$ linearly independent rays $r_1, \ldots, r_{n-1} \in C$. Note that Algorithm 2.4.2 actually computes these rays. In order to verify such a certificate, the user has to test for linear independence and to check that each of the r_i's is in the normal cone, which in turn can be done by maximizing $\langle r_i, x\rangle$ over P (and verifying that the maximum is equal to $\langle r_i, \hat{x}\rangle$).

Suppose the algorithm determines that F is higher-dimensional. A very practical certificate, often used in proofs for non-adjacency, is to write $\hat{x}$ as a nontrivial convex combination of points in P plus a conic combination of extreme rays of P. We now describe how to obtain such

a certificate while running Algorithm 2.6.1. Since F is not an edge, there exists an equation $\langle w, c\rangle = 0$ valid for all $c \in C$ such that w is not parallel to $(s_2 - s_1)$. Algorithm 2.4.2 obtains this equation by maximizing and minimizing $\langle w, c\rangle$ over C. According to the construction of $\mathfrak{O}_C$ in Step 1 of Algorithm 2.6.1, this is done by solving two of the LPs (2.16)–(2.20). We now consider the dual of this LP:

$$\min \quad 0 \tag{2.21}$$

$$\text{s.t.} \quad \sum_{s \in \operatorname{vert}(P)} s\lambda_s + \sum_{r \in \operatorname{ext.rays}(P)} r\mu_r + \hat{x}\xi \;=\; v \tag{2.22}$$

$$\sum_{s \in \operatorname{vert}(P)} -\lambda_s \qquad\qquad - \;\; \xi \;=\; 0 \tag{2.23}$$

$$\lambda \in \mathbb{R}_+^{\operatorname{vert}(P)}, \;\; \mu \in \mathbb{R}_+^{\operatorname{ext.rays}(P)}, \quad \xi \in \mathbb{R} \tag{2.24}$$

The next proposition states that the two sets of dual values obtained for $v \in \{w, -w\}$ yield our desired convex combination.

Proposition 2.6.5. *Let $P \subseteq \mathbb{R}^n$ be a pointed polyhedron, and let $F \subseteq \mathbb{R}^n$ be a face of P of dimension at least 2. Let $s_1, s_2 \in F$ be two vertices such that their barycenter $\hat{x}$ is in F's relative interior, and let $w \in \operatorname{lin.hull}(F - \hat{x})$ be a direction valid for F. Let furthermore $(\lambda^+, \mu^+, \xi^+)$ and $(\lambda^-, \mu^-, \xi^-)$ be optimal solutions of LP (2.21)–(2.24) for $v = +w$ and $v = -w$, respectively. Then*

$$\hat{x} = \sum_{s \in \operatorname{vert}(P)} \frac{\lambda_s^+ + \lambda_s^-}{-\xi^+ - \xi^-} s + \sum_{r \in \operatorname{ext.rays}(P)} \frac{\mu_r^+ + \mu_r^-}{-\xi^+ - \xi^-} r \tag{2.25}$$

holds. Furthermore, if w is not parallel to $(s_2 - s_1)$, then the combination is not the trivial one $\hat{x} = \frac{1}{2}s_1 + \frac{1}{2}s_2$.

Proof. We will first prove that the LP (2.21)–(2.24) is feasible. Due to $\hat{x} \in \operatorname{relint}(F)$, we have that for $v \in \{+w, -w\}$, v lies in the radial cone of $\hat{x}$ w.r.t. P, that is, there exist $\lambda \in \mathbb{R}_+^{\operatorname{vert}(P)}$ and $\mu \in \mathbb{R}_+^{\operatorname{ext.rays}(P)}$ that satisfy $\sum_{s \in \operatorname{vert}(P)}(s - \hat{x})\lambda_s + \sum_{r \in \operatorname{ext.rays}(P)} r\mu_r = v$. By setting ξ according to (2.23), we obtain the two feasible solutions.

Let $(\lambda^+, \mu^+, \xi^+)$ and $(\lambda^-, \mu^-, \xi^-)$ be optimal solutions of LP (2.21)–(2.24) for $v = +w$ and $v = -w$, respectively. From (2.23) and (2.24) we get that ξ^+ and ξ^- must be nonpositive. Furthermore, $(\xi^+ + \xi^-)$ is

strictly negative, since otherwise $\lambda^+ = \lambda^- = 0$ would hold, implying $+w, -w \in \operatorname{recc}(P)$ and contradicting the assumption that P is pointed. Hence, the quotients in (2.25) are well-defined. Adding (2.22) for both LPs yields

$$\sum_{s \in \operatorname{vert}(P)} (\lambda_s^+ + \lambda_s^-)s + \sum_{r \in \operatorname{ext.rays}(P)} (\mu_r^+ + \mu_r^-)r + \hat{x}(\xi^+ + \xi^-) = 0,$$

which proves Equation (2.22). Note that adding (2.23) for both LPs yields $\sum_{s \in \operatorname{vert}(P)}(-\lambda_s^+ - \lambda_s^-) = \xi^+ + \xi^-$, which shows that the equation expresses $\hat{x}$ as a convex combination of points in P plus a conic combination of extreme rays of P.

Now suppose that the combination is the trivial one, i.e., $\mu^+ = \mu^- = 0$ holds, and $\lambda_s^+ = \lambda_s^- = 0$ holds for all $s \in \operatorname{vert}(P) \setminus \{s_1, s_2\}$. From (2.22) and (2.23) we obtain $\lambda_{s_1}^+(s_1 - \hat{x}) + \lambda_{s_2}^+(s_2 - \hat{x}) = w$, which shows that w is parallel to $(s_2 - s_1)$. This concludes the proof. □

In order to make use of the proposition algorithmically, we only have to track when Algorithm 2.4.2 finds an equation different from $\langle \hat{x}, c \rangle - \delta = 0$. The two calls to the optimization oracle $\mathfrak{O}_C$ then yield dual solutions that can be combined as in Proposition 2.6.5.

2.6.4 Detecting Higher-Dimensional Faces

We now come back to one of main motivations, namely to decide for given finite non-empty $S \subseteq P$ and finite $R \subseteq \operatorname{recc}(P)$ whether $F = \operatorname{conv.hull}(S) + \operatorname{conic.hull}(R)$ is a face of P. The cases of $\dim F \in \{0, 1\}$ were handled explicitly in the previous subsections, and we now consider the more general setting. If $\dim F \geq 3$ holds, the situation is more complicated since it is not just a question of the correct dimension. Let $\hat{F}$ be the smallest face of P that contains F. Clearly, since F and $\hat{F}$ are polyhedra and $F \subseteq \hat{F}$ holds, either equality holds or there exists a facet $\langle a, x \rangle \leq \beta$ of F that is not valid for $\hat{F}$, i.e., the maximum objective value when optimizing a over F and $\hat{F}$ differs. This justifies the following algorithm.

Algorithm 2.6.2: Detecting a Face

Input: Optimization oracle $\mathfrak{O}$ for a rational polyhedron $P \subseteq \mathbb{R}^n$, a non-empty finite rational set $S \subseteq P$, and a finite rational set $R \subseteq \operatorname{recc}(P)$

Output: A rational point x that is not in F but in the smallest face $\hat{F}$ of P containing $F := \operatorname{conv.hull}(S) + \operatorname{conic.hull}(R)$, if such a point exists.

1 Compute a relative interior point $\hat{x}$ of F by Equation (2.15).
2 Call Algorithm 2.6.1 with $\mathfrak{O}$ and $\hat{x}$ to obtain $\hat{c} \in \mathbb{Q}^n$.
3 Using $\mathfrak{O}$ and $\hat{c}$, construct an optimization oracle $\mathfrak{O}_{\hat{F}}$ for the smallest face $\hat{F}$ containing $\hat{x}$ (see Algorithm 2.3.1).
4 Compute an outer description $Ax \leq b$ of F with m inequalities using some external tool.
5 **for** $i = 1, 2, \ldots, m$ **do**
6 **if** $\mathfrak{O}_{\hat{F}}(A_{i,*}).\text{val} > b_i$ **then**
7 **return** $\mathfrak{O}_{\hat{F}}(A_{i,*}).\text{point}$.
8 **end**
9 **end**
10 **return** `// F is a face of P.`

2.6.5 Strengthening Inequalities

Consider a face F of $P \subseteq \mathbb{R}^n$ induced by an inequality $\langle a, x \rangle \leq \beta$ valid for P. Suppose we want to find a facet $\hat{F}$ containing this face F, that is, to strengthen the valid inequality as much as possible. By Lemma 2.6.1, a facet $\hat{F}$ contains F if and only if $\hat{F}$ contains the point $\hat{x} \in \operatorname{relint}(F)$ obtained by Equation (2.15) for F being the smallest face with $S \subseteq F$ and $R \subseteq \operatorname{recc}(F)$. Such a facet can be found using the techniques from Section 2.5.4.

Our suggested strategy is to create an optimization oracle $\mathfrak{O}_F$ for F using Algorithm 2.3.1, and use it for Algorithm 2.4.2 to obtain S and R, and compute $\hat{x} \in \operatorname{relint}(F)$ by Equation (2.15). As argued in Section 2.5.4, we then try to separate the point $(2\hat{x} - o)$ for some point o in P's relative interior, to prevent the LP from returning an equation instead of a facet-defining inequality.

INVESTIGATING POLYHEDRA BY ORACLES:

2.7 Solving the Polar Linear Programs

We already encountered two situations, namely the computation of facets (see Section 2.5.2) and of the smallest containing face (see Section 2.6.1), in which we searched for certain inequalities valid for a rational non-empty polyhedron P using a linear program, where P is given by an optimization oracle. In both applications the LP consisted of the following inequalities in addition to only a few others.

$$\langle s, a\rangle - \beta \leq 0 \qquad \text{for all } s \in S \tag{2.26}$$
$$\langle r, a\rangle \leq 0 \qquad \text{for all } r \in R \tag{2.27}$$

Here, $S \supseteq \text{vert}(P)$ is a finite superset of P's vertices, while $R \supseteq \text{ext.rays}(P)$ is a finite superset of its extreme rays. We furthermore assume that the origin $(a, \beta) = (\mathbb{O}, 0)$ is feasible, i.e., none of the additional constraints cuts it off.

This section deals with different aspects for solving such linear programs in practice. We first settle how to solve the separation problem for both constraints using $\mathfrak{O}$. It allows us to start with only a few (or even none) of the inequalities, adding inequalities that are violated by the current solution until all are satisfied. Despite this lazy addition, the LPs may become very large in terms of rows, and hence we implemented a so-called *cut aging* strategy that is described in the second part. In the last part we discuss stabilization, a technique that is applied to contain tailing-off effects.

2.7.1 The Separation Problem

The next simple lemma settles separability of (2.26) and (2.27) by means of the oracle $\mathfrak{O}$.

Lemma 2.7.1. *Let $\mathfrak{O}$ be an optimization oracle for a rational non-empty polyhedron $P \subseteq \mathbb{R}^n$. Then the separation problem for* (2.26) *and* (2.27) *can be solved using a single invocation of $\mathfrak{O}$.*

Proof. In order to separate a point $(\hat{a}, \hat{\beta}) \in \mathbb{R}^n \times \mathbb{R}$, we simply call $\mathfrak{O}$ with $\hat{a}$ as an objective. If $\mathfrak{O}(\hat{a})$ returns an unbounded direction $r \in \operatorname{recc}(P)$, then $\langle r, \hat{a} \rangle > 0$ holds, i.e., $\langle r, a \rangle \leq 0$ is violated. Suppose $\mathfrak{O}(\hat{a})$ returns a point $s \in P$. If $\langle \hat{a}, s \rangle > \hat{\beta}$, then $\langle s, a \rangle - \beta \leq 0$ is violated by $(\hat{a}, \hat{\beta})$. Otherwise, from $\max \{\langle \hat{a}, x \rangle \mid x \in P\} \leq \hat{\beta}$ we can easily deduce that both constraints are satisfied. □

2.7.2 Stabilization

A common problem for cutting-plane based methods is unstable behavior of the intermediate solutions. It is the reason that a lot more cutting-planes (in our case Inequalities (2.26) and (2.27)) are generated than actually needed to prove optimality. In our case this corresponds to too many oracle calls, which is typically our main bottleneck. To get an idea of this behavior we ran IPO's facet computation algorithm in its pure form and projected the sequence of intermediate solutions on two arbitrary variables. The result is depicted on the left side of Figure 2.4, showing a lot of oscillation in the two coordinates. In the literature one can find several approaches to stabilize the procedure, typically by penalizing oscillations in a certain way. Most recent work in that direction was done in the context of column generation, where columns are added to the LP instead of rows. By dualization, both cases constitute the same problem. A simple stabilization technique that works by means of linear programming is due to du Merle, Villeneuve, Desrosiers and Hansen [22]. A more involved one is described in [2].

Du Merle et al. suggested to consider every variable x_i (in our case $x_i = a_i$ for $i \in [n]$ and $x_{n+1} = \beta$), set up an interval $[\delta_i^-, \delta_i^+]$ (the so-called *trust region*) and linearly penalize if x_i is away from that region. More precisely, one adds variables $w_i^+, w_i^- \geq 0$ having nonpositive coefficients $-\varepsilon_i^+, -\varepsilon_i^-$ (we call $\varepsilon^+, \varepsilon^- \in \mathbb{R}_+^n$ *penalties*) in the objective (note that we

maximize) and adds constraints $x_i \leq w_i^+ + \delta_i^+$ and $x_i \geq -w_i^- + \delta_i^-$. Now if $x_i \in [\delta_i^-, \delta_i^+]$ holds, then both extra variables can be set to zero for no cost. But if x_i is outside the trust region, e.g., if $x_i > \delta_i^+$, then the LP has to pay the costs for each unit of $w_i^+ = x_i - \delta_i^+$.

Clearly one has to come up with update strategies for the trust regions (as we don not know the right value of x_i in advance) and for the penalty values. Moreover, in the mentioned papers the authors can usually specify certain initial values since they consider specific problems. We are not in this situation as we do not know the sizes of our inequalities' coefficients a priori. This is particularly important for the penalties: If they are too large, the solution is bound to the trust region, and if they are too small, we do not stabilize at all. In order to calibrate the penalty values we decided to start with an initially large uniform penalty value $\varepsilon \geq 0$ (that is, every $w_i^\pm$-variable has objective value $-\varepsilon$) and the trivial interval $[0, 0]$. This will lead to the origin (i.e., $a = \mathbb{O}, \beta = 0$) as the optimal solution. As long as this is the case we decrease ε by 50 %, until the optimum is different from the origin.

A very interesting observation that we made is that by just continuing this "calibration" we can already reduce the number of iterations significantly. So suppose we actually obtained an optimal solution that is not equal to the origin. Then we separate inequalities as long as possible, that is, either until the optimum is again equal to the origin, or until the current solution is indeed feasible. After this separation sequence we decrease ε further and repeat. Finally, ε will be very small, having no impact on the solution, we can set it to zero and obtain the original LP.

Since a thorough investigation of stabilization approaches and corresponding update strategies is out of the scope of this thesis, we decided to keep this behavior. At the end of this section we will present computational results for an example instance that justify this decision.

In order to obtain correct solutions we eventually have to solve the LP with exact arithmetic. But in fact we do not care about exactly computed stabilization values $w_i^\pm$ and also not about the precision of the computed intermediate solutions. This motivates to run the described stabilization procedure with floating-point precision only and then warm-start a non-stabilized procedure that uses exact arithmetic. To carry out the warm-start we only have to extract those points $\hat{S}$ and directions $\hat{R}$ that were

required to prove optimality of the stabilization LP, i.e., those for which the corresponding slack variables were nonbasic. In our experiments, this strategy worked well in the sense that we rarely had to add more inequalities to the LP after the warm-start.

2.7.3 Cut Aging

Despite stabilization, usually many cutting planes need to be separated, in particular more than actually needed to assert optimality. According to our knowledge, all state-of-the-art solvers implement an aging strategy, which is a simple heuristic that removes inequalities from the LP that were not useful for a certain number of cut rounds. For this, one typically maintains a number called the *age* of a row. If a row is not "useful", its age is incremented, while the age is set to zero for useful ones. The maximum age of a row is 20 by default, but can be specified by the user. We do not maintain a *cut pool*, that is, we do not explicitly collect the removed inequalities. Instead we store the set of known points and directions returned by the oracle and search among those for violated inequalities before calling the oracle. Hence, re-adding inequalities that were removed due to aging is not a costly operation.

The cutting-plane procedure might get stuck in the sense that it only adds back inequalities that were removed due to aging without making any progress. If the number of rounds in which only known inequalities are added and the objective did not improve exceeds the maximum age, then this age is increased. This strategy guarantees that we will eventually terminate. IPO considers a row as "useful" if its dual is nonzero, which is the same strategy that is implemented in SCIP.

2.7.4 Effect of Stabilization and Cut Aging

We consider, as an example instance, an integer program with 105 variables over the *master edge cover polytope*, defined in Section 2.10.2, for the complete graph with 14 nodes. For a fixed random objective, IPO detected six facets and we report statistics for the whole facet-computation. Note that we just want to illustrate the effect of stabilization and cut aging, and by no means claim that the effects will be the same (or similar) for other polyhedra. In fact they will highly depend on the dimension and the relative running times of heuristics and oracles.

Table 2.1: Effects of stabilization and cut aging for lazy separation of Inequalities (2.26)–(2.27).

	No stabilization				Stabilization			
Maximum age	∞	30	20	10	∞	30	20	10
Overall time	286.7	257.4	248.7	260.6	46.9	47.8	50.2	49.6
Approx. LP time	173.3	126.2	120.4	133.8	21.0	21.4	20.6	22.4
Approx. LPs	2742	5102	5869	7956	804	865	1021	1169
Exact LP time	1.3	1.4	1.3	1.3	1.3	1.4	1.8	1.7
Exact LPs	7	7	7	7	9	9	11	10
Oracle time	1.5	1.3	1.1	1.2	4.1	4.0	5.1	4.1
Oracle calls	14	14	14	14	38	38	38	38
Heur. time	101.2	113.7	110.6	104.5	19.6	20.0	21.5	20.3
Heur. calls	1695	1883	1816	1825	429	402	443	407

Table 2.1 shows the results for stabilization turned on or off and for different maximum ages. It contains the number of invocations and running times for several parts. Here, the "approximate LP" is the one that is solved with floating-point precision only, and used for stabilization, while the "exact LP" is the final one that also produces solutions in rational arithmetic. Clearly, stabilization pays off, while the effect of cut aging is relatively small. The latter is probably due to the fact that the running time portion used for solving the LPs is not very large, e.g., compared to a MIP solver. Note that the number of invocations of the exact LP is slightly higher if stabilization is turned on. This is clearly due to a different warm-start of the exact LP, but apart from this we have no good explanation.

For Figure 2.4 we considered the same instance, but just the generation of the first facet, with cut aging disabled. We compare the run without stabilization (plot on the left) and the run with stabilization (plot on the right). The plots show the projections of all LP solutions' trajectories onto two variables. We selected the variables such that the optimum has one variable fixed to zero and the other one nonzero. The first solutions are colored blue, while the last ones are colored red, the

final solution being the same for both runs, namely $(0, -0.38)$. When stabilization is disabled, one can see many oscillations, but they are not gone when we stabilize. Of course we cannot expect a smooth converging behavior due to the fact that the simplex method returns extreme solutions.

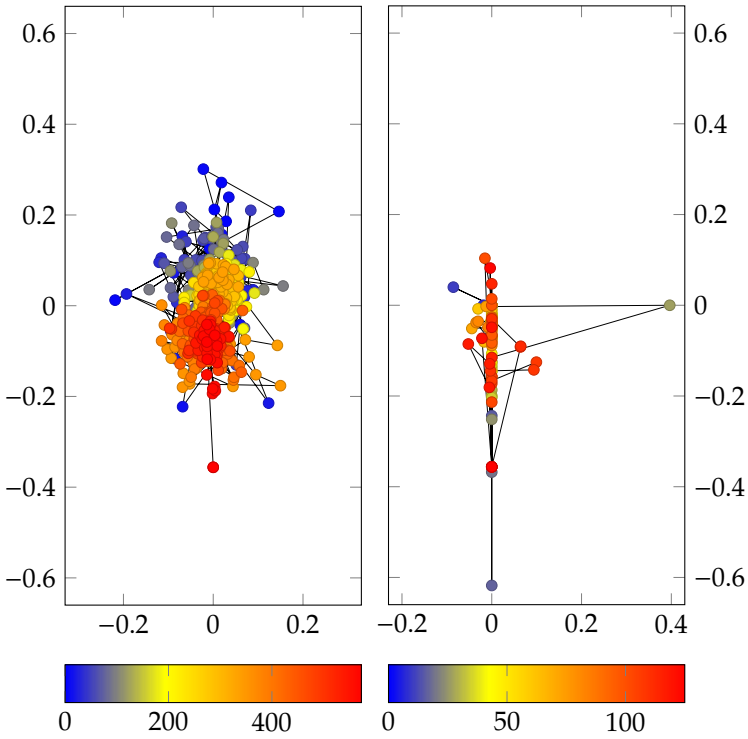


Figure 2.4: Projection of unstabilized and stabilized solution sequence onto two variables. The trajectories show the solutions in sequence of computation, colored by iteration number. Both cases are without cut aging, stabilization disabled in the left, and enabled in the right plot.

INVESTIGATING POLYHEDRA BY ORACLES:

2.8 Improving Readability of Equations and Inequalities

The main use case for IPO is the computation of equations and inequalities valid for a polyhedron of interest with the goal that the user hopefully understands and generalizes them. As this final step is clearly not mechanical, the actual representation of these objects matters a lot. First we only consider *integral* representations, that is, all equations and inequalities presented to a user must have integral normal vectors. Second, we believe that verifying the validity of an equation or an inequality by hand usually requires a certain amount of time per nonzero, and that the effort for larger (absolute values of) coefficients is higher. In case of complicated coefficients a user may even think that there are number-theoretic reasons for the validity, although there could be a simple combinatorial explanation. This motivates our decision to prefer sparse normal vectors over dense ones, but also try to avoid large coefficients. For equations we already designed the affine-hull algorithm such that sparse objective vectors are preferred (see Section 2.4.5).

In this section we consider a simple technique to post-process equations and inequalities to find a nicer representation. For this we try to minimize Manhattan norms $\|a\|_1 := \sum_{i=1}^{n} |a_i|$ of the corresponding normal vectors $a \in \mathbb{Z}^n$. It clearly avoids coefficients of large absolute value, but also cares a bit for sparsity, e.g., in contrast to the maximum norm. Since finding normal vectors with a global minimum norm is hard (and out of the scope of this thesis), we present a local-improvement heuristic.

2.8.1 Manhattan Norm Problems

Improving the readability of a set of given equations as discussed above is essentially the following problem.

Problem 2.8.1 (ℓ_1-shortest Basis)**.** *Given a vector space $V \subseteq \mathbb{Q}^n$ by means of a rational basis, find an integral basis $v_1, \ldots, v_k \in \mathbb{Z}^n$ of V whose sum of ℓ_1-norms $\sum_{i=1}^{k} \|v_i\|_1$ is minimum.*

Although we are not aware of an NP-hardness proof, there is a result by Haviv and Regev [35], who showed that unless $\mathsf{NP} \subseteq \mathsf{RP}$[8] holds, there is no polynomial-time constant-factor approximation algorithm for the *shortest vector problem* for the ℓ_1-norm. It is not too hard to see that an optimum solution to Problem 2.8.1 always contains a vector of minimum ℓ_1-norm since otherwise we could add to to the basis and remove any other linearly dependent vector. Note that Haviv and Regev's result holds for any ℓ_p norm for $1 \leq p \leq \infty$.

Short integral vectors in the span of the equations' normal vectors and the normal vector of a given inequality are only useful if we can actually replace the latter by such a short vector. Let $v_1, \ldots, v_k$ be a basis of any subspace V of $\mathbb{R}^n$. We say that a vector $v \in V$ *depends on v_i (w.r.t. this basis)* if the multiplier λ_i in the (unique) linear combination $v = \sum_{i=1}^{k} \lambda_i v_i$ is nonzero. Assuming that $v_2, \ldots, v_k$ are the equations' normals, and that v_1 is that of a given inequality, our problem of interest is as follows.

Problem 2.8.2 (ℓ_1-shortest Vector with One Dependency)**.** *Given a vector space $V \subseteq \mathbb{Q}^n$ by means of a basis $v_1, \ldots, v_k \in \mathbb{Z}^n$, find a vector $v \in \mathbb{Z}^n \setminus \{0\}$ of minimum ℓ_1-norm which depends on v_1.*

It is again not hard to see this new problem is not simpler than the shortest vector problem: We reorder the vectors $v_1, \ldots, v_k$ in k ways such that the first vector is always a different one, and solve Problem 2.8.2 for each of these instances. Then, among the k solutions we will find our shortest vector since it must depend on some of the v_i.

Due to the hardness of these problems we decided to implement a local improvement heuristic that we present in the next section.

[8] RP consists of those decision problems that can be solved in polynomial time on a randomized Turing machine.

2.8.2 Two Vectors: An Exact Algorithm

Our main result in this section is the following theorem that enables us to approximately reduce the ℓ_1-norm sum of a basis by iteratively considering pairs of basis vectors, solving the corresponding subproblem to optimality and replacing one of the vectors by the solution.

Theorem 2.8.3. *Problem 2.8.2 for $k = 2$ can be solved in polynomial time.*

Let us first start with a simple lemma that tells us how to reduce the problem to one where we only have to consider *integral* combinations of the vectors. This is ensured by replacing them by a lattice basis of their span.

Lemma 2.8.4. *Let $\hat{u}, \hat{v} \in \mathbb{Z}^n$ be two linearly independent vectors, and let the matrix $H = \begin{pmatrix} \alpha & 0 \\ \beta & \gamma \end{pmatrix} \in \mathbb{Z}^{2\times 2}$ be such that $[H \mid \mathbb{O}_{2\times(n-2)}]$ is the Hermite normal form of $\begin{pmatrix} \hat{v}^\intercal \\ \hat{u}^\intercal \end{pmatrix} \in \mathbb{Z}^{2\times n}$. Then $v = \frac{1}{\alpha}\hat{v}$ and $u = \frac{1}{\alpha\gamma}(\alpha\hat{u} - \beta\hat{v})$ are a lattice basis of $\operatorname{lin.hull}(\hat{u}, \hat{v}) \cap \mathbb{Z}^n$ and a vector $(\lambda u + \mu v)$ depends on $\hat{u}$ if and only if it depends on u.*

Proof. Let $U \in \mathbb{Z}^{n\times n}$ be the unimodular matrix such that we have

$$\left[H \mid \mathbb{O}_{2\times(n-2)}\right] = \begin{pmatrix} \hat{v}^\intercal \\ \hat{u}^\intercal \end{pmatrix} \cdot U^\intercal,$$

which implies $\alpha e^{(1)} = U\hat{v}$ and $\beta e^{(1)} + \gamma e^{(2)} = U\hat{u}$. From the first equation we derive $Uv = e^{(1)}$ and from both equations we obtain

$$Uu = \frac{1}{\alpha\gamma}(\alpha U\hat{u} - \beta U\hat{v}) = \frac{1}{\alpha\gamma}(\alpha\beta e^{(1)} + \alpha\gamma e^{(2)} - \alpha\beta e^{(1)}) = e^{(2)}.$$

Since U is unimodular (in particular, $U^{-1} \in \mathbb{Z}^{n\times n}$), the vectors u and v are integral. Consider any pair of multipliers $\lambda, \mu \in \mathbb{R}$ for which $w = \lambda u + \mu v$ is an integral vector. Since then also $Uw = \lambda Uu + \mu Uv = \lambda e^{(2)} + \mu e^{(1)}$ is an integral vector, λ and μ must be integers and thus u and v form a lattice basis.

We now prove the dependency property. A vector w does not depend on $\hat{u}$ w.r.t. the basis $\hat{u}, \hat{v}$ if and only if it is a multiple of $\hat{v}$. Similarly, w

does not depend on u w.r.t. the basis u, v if and only if it is a multiple of v. Since $v = \frac{1}{\alpha}\hat{v}$ holds, the dependency properties are also equivalent, which concludes the proof. □

For the rest of this section we can now assume that $u, v \in \mathbb{Z}^n$ are a lattice basis and we want to find *integral* multipliers λ, μ with $\lambda \neq 0$ such that

$$\|\lambda u + \mu v\|_1 = \sum_{i=1}^{n} |\lambda u_i + \mu v_i| \tag{2.28}$$

is minimum. Since (λ, μ) and $(-\lambda, -\mu)$ have the same objective values, we will even require $\lambda \geq 1$. One may be afraid of doing a case distinction in order to resolve the n absolute value expressions in (2.28) since in principle this results in 2^n cases. It turns out that this combinatorial explosion does not occur because the sign of $\lambda u_i + \mu v_i$ depends on the comparison of μ/λ with $-u_i/v_i$ for every $i \in [n]$. This leaves at most $n + 1$ intervals such that for all λ, μ (with $\lambda \geq 1$) such that μ/λ is in one interval, the objective function (2.28) is linear. This observation is made precise in the following lemma.

Lemma 2.8.5. *Let $u, v \in \mathbb{Z}^n$ be two linearly independent vectors and let $-\infty = q_0 < q_1 < \ldots < q_{k-1} < q_k = \infty$ be the sorted elements of the set $Q := \left\{-\frac{u_i}{v_i} \mid i \in [n] \text{ with } v_i \neq 0\right\} \cup \{\pm\infty\}$. Then the sign pattern of $\lambda u + \mu v$ is constant over all multiplier pairs (λ, μ) with $\lambda > 0$ for which μ/λ is in any fixed interval $[q_{j-1}, q_j]$ for $j \in [k]$.*

Proof. Let (λ, μ) and (λ', μ') with $\lambda, \lambda' > 0$ be two such multiplier pairs satisfying $\frac{\mu}{\lambda}, \frac{\mu'}{\lambda'} \in [q_{j-1}, q_j]$ for some $j \in [k]$, but with different signs on position i: $\lambda u_i + \mu v_i < 0 < \lambda' u_i + \mu' v_i$. We have $v_i \neq 0$ since otherwise $\lambda u_i < 0 < \lambda' u_i$ holds, contradicting $\lambda, \lambda' > 0$.

Hence $-\frac{u_i}{v_i} \in Q$, that is, $\frac{\mu}{\lambda}$ and $\frac{\mu'}{\lambda'}$ are either both less than or equal to $-\frac{u_i}{v_i}$ or both greater than $-\frac{u_i}{v_i}$. This implies that $\frac{\mu}{\lambda} + \frac{u_i}{v_i}$ and $\frac{\mu'}{\lambda'} + \frac{u_i}{v_i}$ have the same sign, which in turn implies that $\lambda u_i + \mu v_i$ and $\lambda' u_i + \mu' v_i$ have the same sign (using the fact that $\lambda, \lambda' > 0$). This concludes the proof. □

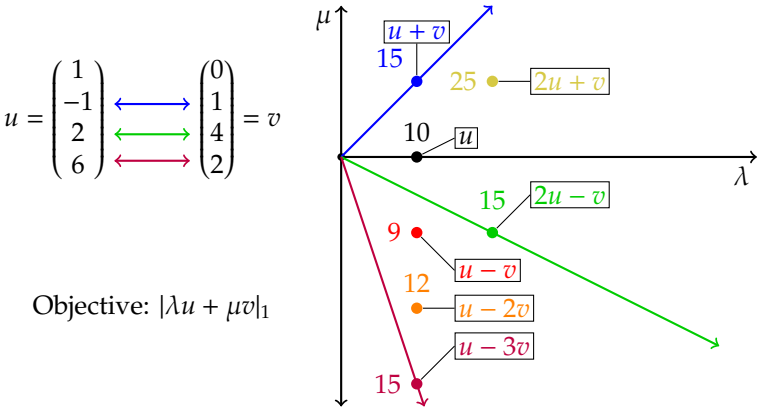


Figure 2.5: Illustration of Lemma 2.8.5.

Given $q \in \mathbb{Q} \cup \{\pm\infty\}$, we define the *ray* $r \in \mathbb{Z}^2$ *representing* q as follows:

$$r = \begin{cases} (r_1, r_2) & \text{for } q \in \mathbb{Q},\ q = \frac{r_2}{r_1} \text{ with } r_1 \in \mathbb{Z}_+ \text{ and } r_2 \in \mathbb{Z} \text{ coprime} \\ (0,1) & \text{for } q = \infty \\ (0,-1) & \text{for } q = -\infty \end{cases} \tag{2.29}$$

The case distinction is shown in Figure 2.5 for two vectors $u, v \in \mathbb{Z}^4$. The three colored rays emanating from the origin are the (scaled) rays representing q_1, q_2 and q_3. They partition the set of pairs $(\lambda, \mu) \in \mathbb{R}_+ \times \mathbb{R}$ into four cones each of which has a linear objective function. Note that the λ-axis does not represent an element of Q, i.e, $0 \notin Q$, since u has no zero entry.

According to Lemma 2.8.5 we have to solve at most $n + 1$ integer programs

$$\begin{aligned} \min \quad & c_1\lambda + c_2\mu && && (2.30)\\ \text{s.t.} \quad & \lambda && > 0 && (2.31)\\ & -q_{j-1}\lambda + && \mu \geq 0 && (2.32)\\ & -q_j\lambda + && \mu \leq 0 && (2.33)\\ & && \lambda, \mu \in \mathbb{Z}, && (2.34) \end{aligned}$$

where c is a suitable objective function we will specify later and $j \in [k]$. Note that (2.31) is a strict inequality constraint that could also be replaced by $\lambda \geq 1$. Since integer programming is polynomial-time solvable in fixed dimension (see [48]) these IPs can be solved in polynomial time, which proves Theorem 2.8.3.

For efficiency reasons, instead of solving IP (2.30)–(2.34) by a general algorithm we will now devise a problem-specific approach. Since we can evaluate objective for the pairs (λ, μ) with $\mu/\lambda = q_i$ for some $i \in [k-1]$ directly, we can restrict our search for (integral) λ, μ satisfying $-q_{j-1}\lambda + \mu > 0$ and $-q_j\lambda + \mu < 0$, which implies $\lambda > 0$. To see this, observe that the cone whose extreme rays are the rays representing q_{j-1} and q_j is contained in $\mathbb{R}_+ \times \mathbb{R}$, and hence its interior is contained in $\mathbb{R}_{>0} \times \mathbb{R}$. The next lemma captures the objective function representing $\|\lambda u + \mu v\|_1$ for such a cone.

Lemma 2.8.6. *Let $u, v \in \mathbb{Z}^n$ be two linearly independent vectors and let $-\infty = q_0 < q_1 < \ldots < q_{k-1} < q_k = \infty$ be as in Lemma 2.8.5. Consider a fixed $j \in [k]$ and define*

$$I_j^+ := \left\{ i \in [n] \mid v_i > 0, -\frac{u_i}{v_i} \leq q_{j-1} \text{ or } v_i < 0, -\frac{u_i}{v_i} \geq q_j \text{ or } v_i = 0, u_i > 0 \right\}$$
$$I_j^- := \left\{ i \in [n] \mid v_i < 0, -\frac{u_i}{v_i} \leq q_{j-1} \text{ or } v_i > 0, -\frac{u_i}{v_i} \geq q_j \text{ or } v_i = 0, u_i < 0 \right\}.$$

Then all $(\lambda, \mu) \in \mathbb{R}_{>0} \times \mathbb{R}$ with $q_{j-1} \leq \mu/\lambda \leq q_j$ satisfy

$$\|\lambda u + \mu v\|_1 = \sum_{i \in I_j^+} (\lambda u_i + \mu v_i) - \sum_{i \in I_j^-} (\lambda u_i + \mu v_i).$$

Proof. We have to prove that I_j^+ (resp. I_j^-) contains the set of $i \in [n]$ for which $w_i := \lambda u_i + \mu v_i$ is positive (resp. negative). Suppose $\lambda u_i + \mu v_i > 0$ holds. If $v_i = 0$, then the sign of w_i agrees with the sign of u_i. Otherwise, the condition is equivalent to $v_i(\frac{u_i}{v_i} + \frac{\mu}{\lambda}) > 0$. For $v_i > 0$ (resp. $v_i < 0$) this implies $-\frac{u_i}{v_i} < \frac{\mu}{\lambda}$ (resp. $-\frac{u_i}{v_i} > \frac{\mu}{\lambda}$). Using the fact that $-\frac{u_i}{v_i}$ is equal to q_ℓ for some $\ell \in [k-1]$, we observe that this is equivalent to $-\frac{u_i}{v_i} \leq q_{j-1}$ for $v_i > 0$ and to $-\frac{u_i}{v_i} \geq q_j$ for $v_i < 0$. This proves the statement for positive $\lambda u_i + \mu v_i$, the case of negative $\lambda u_i + \mu v_i$ being similar. □

Having the objective function available, we can now turn to the description of the algorithm that solves the modified version of IP (2.30)–(2.34). For this, we are given two linearly independent rays $q, r \in \mathbb{Q}^2$ and an objective vector $c \in \mathbb{Q}^2$ satisfying $\langle q, c\rangle, \langle r, c\rangle \geq 0$. The goal is to find an integral point $p \in \text{int}(\text{conic.hull}(\{q, r\}))$ with minimum c-value.

Note that there always exists such a point since $(q + r)$ is a rational ray contained in the interior of conic.hull($\{q, r\}$), and a suitable scaled version is integral. Note also Lemma 2.8.6 implies that the objective is nonnegative for the whole cone, proving that its extreme rays have nonnegative scalar product with the objective vector.

For a direction $a \in \mathbb{Z}^2$ the *width* of a polytope P along a is defined as $w_a(P) := \max\{\langle a, x\rangle \mid x \in P\} - \min\{\langle a, x\rangle \mid x \in P\}$. The *lattice width* $w(P)$ of P is defined as the minimum of all widths over all integral directions $a \neq \mathbb{O}$. The mentioned theorem reads as follows:

Proposition 2.8.7 (Flatness Theorem for Dimension Two, see [44] and [38])**.** *Every polygon $P \subseteq \mathbb{R}^2$ with lattice width greater than $1 + 2/\sqrt{3} \approx 2.15$ contains at least one lattice point in its interior.*

In dimension two we can easily approximate the lattice width of triangles by a factor of 2:

Proposition 2.8.8 (Section 2.3 in [24])**.** *Let $T = \text{conv.hull}\{q, r, \mathbb{O}\}$ be a triangle with $q, r \in \mathbb{Q}^2$. Then a vector $a \in \mathbb{Z}^2 \setminus \{\mathbb{O}\}$ with minimum ∞-norm of $\binom{q^\intercal}{r^\intercal} \cdot a$ can be computed in polynomial time and $w(T) \leq w_a(T) \leq 2w(T)$ holds.*

Assume for a moment $\langle c, q\rangle, \langle c, r\rangle > 0$ and scale r to $\hat{r} := \langle c, q\rangle / \langle c, r\rangle\, r$ in order to make the c-objectives of q and $\hat{r}$ equal. In order to solve our refined integer program we can now compute a "thin" direction a for the triangle $T = \text{conv.hull}\{q, \hat{r}, \mathbb{O}\}$ using Proposition 2.8.8. We can then scale q and $\hat{r}$ by a factor of $\tau > 0$ such that $w_a(\tau T) = 5$ holds. The approximation factor then implies that τT satisfies the conditions of Proposition 2.8.7, and hence contains an integer point. Furthermore, its integer points can be covered by at most 6 lines that are orthogonal to a. In order to solve the one-dimensional problem we apply a unimodular transformation such that the "thin" direction a is mapped to $a' = \mathbb{e}^{(1)}$, which makes the covering lines vertical and thus easy to analyze. The initial scaling of r ensures optimality, since the edge of τT connecting τq with τr is orthogonal to c.

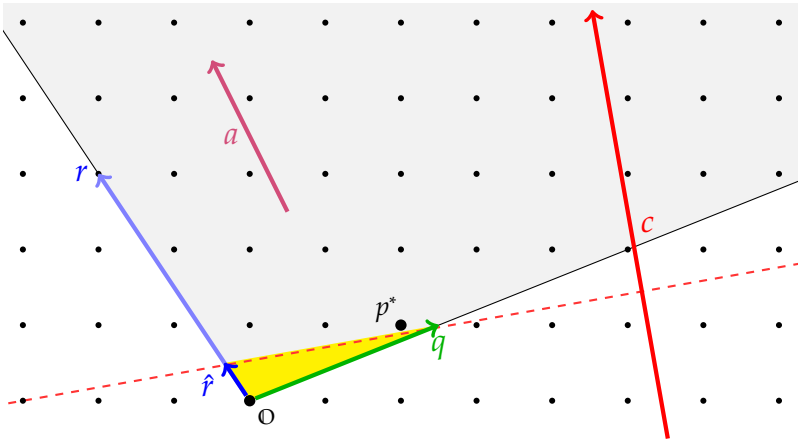


Figure 2.6: Illustration of the instance from Example 2.8.9.

Example 2.8.9. *Consider the instance defined by* $q = \frac{1}{2}\binom{5}{2}$, $r = \binom{-2}{3}$, *and* $c = \frac{1}{3}\binom{-3}{17}$. *The scaled ray is* $\hat{r} = \frac{19/6}{57/3} r = \frac{1}{6}\binom{-2}{3}$ *and a "thin" vector minimizing the* ∞*-norm of* $\begin{pmatrix} 5/2 & 1 \\ -1/3 & 1/2 \end{pmatrix} \cdot a$ *is* $a = \binom{-1}{2}$. *Figure 2.6 depicts these objects, including the triangle* $T = \operatorname{conv.hull}\{q, \hat{r}, \mathbb{O}\}$.

The "thin" direction a *induces the unimodular transformation with matrix* $U = \begin{pmatrix} -1 & 2 \\ -1 & 1 \end{pmatrix}$ *and transposed inverse* $U^{-\intercal} = \begin{pmatrix} 1 & 1 \\ -2 & -1 \end{pmatrix}$. *The transformed objects are*

$$q' := Uq = \frac{1}{2}\begin{pmatrix} -1 \\ -3 \end{pmatrix}, \qquad r' := U\hat{r} = \frac{1}{6}\begin{pmatrix} 8 \\ 5 \end{pmatrix},$$

$$a' := U^{-\intercal}a = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{and} \qquad c' := U^{-\intercal}c = \frac{1}{3}\begin{pmatrix} 14 \\ -11 \end{pmatrix}.$$

This leads to a scaling factor of $\tau := 5/(r'_1 - q'_1) = \frac{30}{11}$ *and the vertical covering lines are chosen for integers* x *between the values* $x_{\min} := \tau q'_1 \approx -1.36$ *and* $x_{\max} := \tau r'_1 \approx 3.64$. *Figure 2.7 depicts the situation after transformation and scaling by* τ.

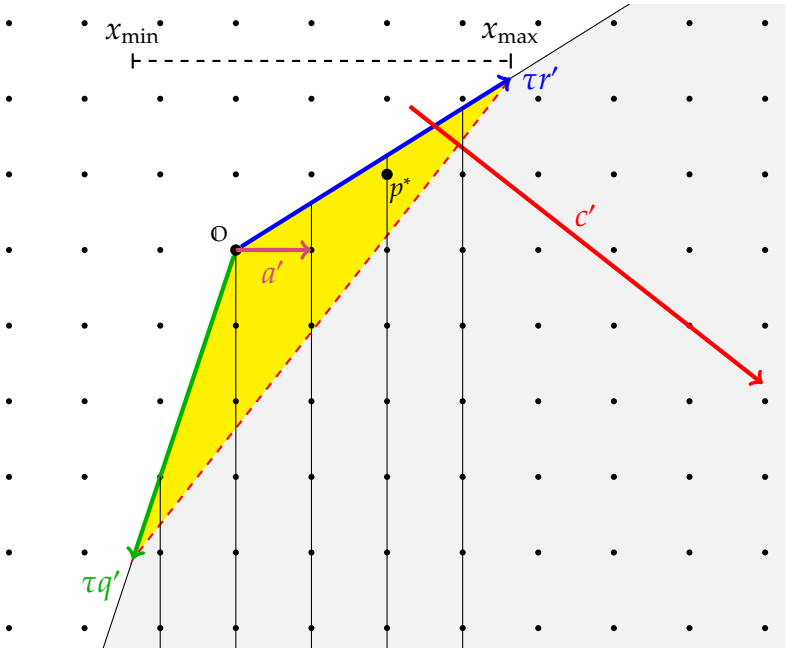


Figure 2.7: Illustration of the transformed instance from Example 2.8.9.

The following algorithm makes these ideas precise:

Lemma 2.8.10. *Algorithm 2.8.1 is correct and runs in polynomial time.*

Proof. By Proposition 2.8.8, Step 4 can be carried out in polynomial time and the vector a satisfies $w(T) \le w_a(T) \le 2w(T)$ for the triangle $T := \text{conv.hull}\{q, \hat{r}, \mathbb{O}\}$. By construction a is primitive, i.e., $\gcd(a_1, a_2) = 1$ holds. Hence, Step 6 can be performed using the extended Euclidean algorithm. The (unimodular) transformation in Step 7 transforms the rays in such a way that the direction a corresponds to $a' := (1, 0)$. Observe that $\langle U^{-\intercal} c, Up\rangle = \langle c, p\rangle$ holds for all points p, i.e., the new objective vector c' corresponds to c after the transformation.

If $\langle c, q\rangle, \langle c, r\rangle > 0$ does not hold, we can assume w.l.o.g. that $\langle c, r\rangle = 0$ holds. In this case, every feasible point p with $\langle a, p\rangle = 1$ is optimal. Hence, after the transformation, any point (x, y) with $x = 1$ is suitable, and the algorithm will find it since Step 17 is executed due to $c'_2 = 0$.

If $\langle c, q\rangle, \langle c, r\rangle > 0$ holds, we have $\langle c', q'\rangle = \langle c', r'\rangle$ due to the preliminary scaling in Step 3. The factor $\tau > 0$ computed in Step 11 ensures that $w_a(\tau T) = 5$ holds, ensuring that τT satisfies the condition of Proposition 2.8.7 and hence contains an integer point. By unimodularity of the transformation, also $\tau T'$ (with $T' := \{Up \mid p \in T\}$) contains an integer point. Furthermore, the integer points in $\tau T'$ can be covered by at most six vertical lines with x-coordinates between $x_{\min}$ and $x_{\max}$. It is easy to verify that Steps 14–17 find for a specified $x \in \mathbb{Z}$ an integer y such that $(x, y) \in \text{int}(\text{conic.hull}\{q', r', \mathbb{O}\})$ is of minimum c'-value (if such a y exists). Note that by the definition of Y we do not necessarily have $(x, y) \in \tau T'$. If a y exists, its objective value $\left\langle c, U^{-1}(x, y)^{\intercal}\right\rangle = \langle c', (x, y)^{\intercal}\rangle$ is compared to that of an already known best solution.

It remains to show optimality, i.e., that an optimal point is among those considered. Suppose an optimal point p^* lies outside of $\tau T'$. Since the line connecting q' and r' is orthogonal to c' due to $\langle c', q'\rangle = \langle c', r'\rangle$, observe that c' is the normal vector of this edge of $\tau T'$. Hence, any point inside of T' has better objective than p^*, which concludes the proof. □

Algorithm 2.8.1: Conic Integer Program in 2D

Input: Lin. indep. rays $q, r \in \mathbb{Q}^2$, vector $c \in \mathbb{Z}^2 \setminus \{\mathbb{O}\}$ with $\langle c, q\rangle, \langle c, r\rangle \geq 0$.

Output: An integer point $p \in \text{int(conic.hull}\{q, r\}) \cap \mathbb{Z}^2$ minimizing $\pi := \langle c, p\rangle$, together with the objective value π.

1 **if** $\langle c, q\rangle = 0$ **or** $\langle c, r\rangle = 0$ **then** Compute $a := \frac{1}{\gcd(c_1, c_2)} c$ and $\hat{r} := r$.
2 **else**
3 | Compute $\hat{r} := \frac{\langle c,q\rangle}{\langle c,r\rangle} r$.
4 | Compute $a \in \mathbb{Z}^2 \setminus \{\mathbb{O}\}$ minimizing the ∞-norm of $\binom{q^\intercal}{\hat{r}^\intercal} a$.
5 **end**
6 Compute $s, t \in \mathbb{Z}$ with $1 = sa_1 + ta_2$ and let $U := \begin{pmatrix} a_1 & a_2 \\ -t & s \end{pmatrix}$.
7 Compute $q' := Uq$, $r' := U\hat{r}$ and $c' := U^{-\intercal} c$.
8 **if** $\langle c, q\rangle = 0$ **or** $\langle c, r\rangle = 0$ **then** Let $x_{\min} := 1$, $x_{\max} := 1$, $\tau := 1$, and $\pi := \infty$.
9 **else**
10 | Compute $x_{\min} := \min\left\{q'_1, r'_1, 0\right\}$ and $x_{\max} := \max\left\{q'_1, r'_1, 0\right\}$.
11 | Compute $\tau := 5/(x_{\max} - x_{\min})$ and $\pi := \infty$.
12 **end**
13 **for** $x \in [\tau x_{\min}, \tau x_{\max}] \cap \mathbb{Z}$ **do**
14 | Let $Y := \{y \in \mathbb{R} \mid (x, y) \in \text{int(conic.hull}\{r', q'\})\}$.
15 | **if** $c'_2 > 0$ **then** compute $y := \inf(Y \cap \mathbb{Z})$.
16 | **if** $c'_2 < 0$ **then** compute $y := \sup(Y \cap \mathbb{Z})$.
17 | **if** $c'_2 = 0$ **then** compute $y \in Y \cap \mathbb{Z}$, setting $y := \infty$ if $Y \cap \mathbb{Z} = \emptyset$.
18 | **if** $y \neq \pm\infty$ **and** $\langle c', (x, y)\rangle < \pi$ **then** let $p := U^{-1}(x, y)^\intercal$ and $\pi := \langle c, p\rangle$.
19 **end**
20 **return** (p, π)

Algorithm 2.8.2: Exact ℓ_1-Shortest Vector for Lattice Bases in 2D

Input: Linearly independent vectors $u, v \in \mathbb{Z}^n$.

Output: Multipliers $\lambda^*, \mu^* \in \mathbb{Z}$ with $\lambda > 0$ minimizing $\pi^* := \|\lambda u + \mu v\|_1$.

1 Compute $Q := \left\{-\frac{u_i}{v_i} \mid i \in [n] \text{ with } v_i \neq 0\right\} \cup \{\pm\infty\}$.
2 Sort elements of Q as $-\infty = q_0 < q_1 < \ldots < q_{k-1} < q_k = \infty$.
3 Initialize $(\lambda^*, \mu^*, \pi^*) := (1, 0, \|u\|_1)$.
4 **for** $j = 1, 2, \ldots, k-1$ **do**
5 Let $(\lambda, \mu) \in \mathbb{Z}^2$ be the ray representing q_j and let $\pi := \|\lambda u + \mu v\|_1$.
6 **if** $\pi < \pi^*$ **then** set $(\lambda^*, \mu^*, \pi^*) := (\lambda, \mu, \pi)$.
7 **end**
8 **for** $j = 1, 2, \ldots, k$ **do**
9 Compute the rays $r, r' \in \mathbb{Z}^2$ representing q_{j-1} and q_j, respectively.
10 Let I_j^+ and I_j^- be defined as in Lemma 2.8.6.
11 Compute $c := (v(I_j^+) - v(I_j^-), u(I_j^+) - u(I_j^-))^\intercal$.
12 Call Algorithm 2.8.1 for r, r' and c, obtaining (λ, μ) and π.
13 **if** $\pi < \pi^*$ **then** set $(\lambda^*, \mu^*, \pi^*) := (\lambda, \mu, \pi)$.
14 **end**
15 **return** $(\lambda^*, \mu^*, \pi^*)$

We can immediately turn to the verification of the algorithm's correctness.

Theorem 2.8.11. *Algorithm 2.8.2 is correct and runs in polynomial time.*

Proof. By Lemma 2.8.5, the pair (λ, μ) minimizing $\|\lambda u + \mu v\|_1$ can be found among the $k-1$ candidates tested in Steps 4–7 or in the interior of one of the k cones defined by $q_{j-1} < \mu/\lambda < q_j$ (and $\lambda \geq 0$) for $j \in [k]$.

Let us consider such a cone C for any $j \in [k]$. Then the vectors r, r' computed in Step 9 are the extreme rays of C and Lemma 2.8.6 shows that the vector c computed in Step 11 satisfies $c_1\lambda + c_2\mu = \|\lambda u + \mu v\|_1$ for all $(\lambda, \mu) \in C$. Algorithm 2.8.1 correctly determines the c-minimum pair $(\lambda, \mu) \in C$ by Lemma 2.8.10 and runs in polynomial time, which concludes the proof. □

Using Algorithm 2.8.2 as a subroutine, we can now easily provide an algorithm that solves Problem 2.8.2 in dimension two.

Algorithm 2.8.3: Exact ℓ_1-Shortest Vector in 2D
Input: Linearly independent vectors $\hat{u}, \hat{v} \in \mathbb{Z}^n$.
Output: Multipliers $\hat{\lambda}, \hat{\mu} \in \mathbb{Q}$ with $\hat{\lambda} > 0$ minimizing $\hat{\pi} := \|\hat{\lambda}\hat{u} + \hat{\mu}\hat{v}\|_1$ such that $\hat{\lambda}\hat{u} + \hat{\mu}\hat{v}$ is integral.

1 Compute the Hermite normal form $\begin{pmatrix} \alpha & 0 \\ \beta & \gamma \end{pmatrix}$ of $\begin{pmatrix} \hat{v}^\intercal \\ \hat{u}^\intercal \end{pmatrix}$.
2 Call Algorithm 2.8.2 for $\frac{1}{\alpha\gamma}(\alpha\hat{u} - \beta\hat{v})$ and $\frac{1}{\alpha}\hat{v}$, and obtain $(\lambda^*, \mu^*, \gamma^*)$.
3 **return** $(\frac{1}{\gamma}\lambda^*, \frac{1}{\alpha}\mu^* - \frac{\beta}{\alpha\gamma}\lambda^*, \gamma^*)$

The next corollary just combines the results of this section to establish the correctness of our final algorithm.

Corollary 2.8.12. *Algorithm 2.8.3 is correct and runs in polynomial time.*

Proof. By Theorem 2.8.11, Lemma 2.8.4 and the simple calculation

$$\lambda^*\left(\frac{1}{\alpha\gamma}(\alpha\hat{u} - \beta\hat{v})\right) + \mu^*\left(\frac{1}{\alpha}\hat{v}\right) = \left(\frac{1}{\gamma}\lambda^*\right)\hat{u} + \left(\frac{1}{\alpha}\mu^* - \frac{\beta}{\alpha\gamma}\lambda^*\right)\hat{v},$$

we obtain that the returned multipliers are correct, establishing the correctness of Algorithm 2.8.3. Since the Hermite normal form can be computed in polynomial time (see [43]), and by Theorem 2.8.11, it also runs in polynomial time. □

2.8.3 A Fast Heuristic

Solving Problem 2.8.2 using the ideas from the previous section can be time-consuming for very large numbers. We highlight a few places where, for the price of optimality, a significant speedup can be obtained. The algorithm consists of three stages:

1. Replace (u, v) by a lattice basis of their span.

2. Find (at most $n+1$) regions for the multipliers (λ, μ) having a linear objective.

3. For each region, solve the corresponding integer program with two variables.

Clearly, if we skip Step 1, we do not reach all possible combinations $\lambda u + \mu v \in \mathbb{Z}^n$ in general. Still, all produced solutions are valid.

The most time-consuming step is Step 3, and we can save a lot of computation time if we restrict our search only to those pairs (λ, μ) with $\lambda/\mu \in Q$. Although this clearly limits the set of vectors $w = \lambda u + \mu v \in \mathbb{Z}^n$ considered, every w having an entry $w_i = 0$ with $u_i, v_i \neq 0$ is checked. In particular, if there is a combination that is much sparser than u and v, the heuristic will still consider it.

2.8.4 Implementation

The problem considered in this section is clearly not central to this thesis, and hence we spent only a limited amount of time on the implementation. One of the simplest ideas for using the algorithms is to repeatedly check pairs of vectors for possible norm reductions until no such improvement is possible. IPO contains precisely this functionality, except that it works in two rounds: in the first round, the cheaper heuristic is run, and in the second round, the more expensive Algorithm 2.8.3 is tried.

We did not carry out a computational study, but would like to mention that our observations are moderate: Running the heuristic does not take a lot of time, which is why we sticked to the simple pair-checking strategy described in the previous paragraph. Furthermore, for vectors with less than 20 coefficients it produces good results in the sense that we could not improve them by looking at the results. For dense vectors usually not much changes, but in general it is hard to judge the results, as we do not know the optimal bases.

INVESTIGATING POLYHEDRA BY ORACLES:

2.9 Computational Studies: Dimensions

In this section we describe results of three experiments we conducted on some well-known benchmark MIPs. In Section 2.9.1 we report on the dimensions of polyhedra related to the instances, and discuss particular cases that give insight into the capabilities and limitations of our implementation. In Section 2.9.2 we consider the maximum faces for the objectives specified in the respective instances. For the last experiment, presented in Section 2.9.3, we use our code to analyze the inequality constraints of the mentioned instances with respect to the dimensions of the induced faces.

Hardware. The experiments were conducted on a 64-bit Intel i3-2310M CPU at 2.1 GHz with 3 MB cache and 4 GB main memory.

Software. As an exact LP solver we used the development version of SoPlex from January 2015. IPO was compiled with GCC 4.6.3[9] and linked to the external libraries GMP 3.5.2[10]. As an oracle we used the exact MIP solver ExactSCIP 3.0.0 linked to GMP 4.3.1 (the recommended version for ExactSCIP), MPFR 2.4.2[11], QSopt_ex 2.5.10 (see [6, 5]), and EGlib-2.6.20[12]. Using IPO's `MixedIntegerProgramCorrectorOracle` on

[9] GCC, the GNU Compiler Collection, available at `http://gcc.gnu.org/`

[10] The GNU Multiple Precision Arithmetic Library, available at `http://gmplib.org/`

[11] The GNU MPFR Library, available at `http://www.mpfr.org/`

[12] EGlib, available at `http://www.dii.uchile.cl/~daespino/EGlib_doc/main.html`

top of SCIP-3.1.1, the approximate oracle we used, we could turn the floating-point solutions into rational ones (see Section 2.3.4).

2.9.1 Dimensions of Polyhedra from MIPLIB Instances

In order to demonstrate the capabilities and also the limitations of our implementation of Algorithm 2.4.3 we computed the affine hulls for various polyhedra stemming from the MIPLIB 2.0. We selected all instances with at most 1000 variables that are solved *exactly* within 5 minutes, since the running time of the algorithm depends on the ambient dimension and on the running time of the oracle. For each instance we considered the linear relaxation P, the linear relaxation Q of the presolved instance and their respective mixed-integer hulls P_I and Q_I. The presolve was done by SCIP using default settings in floating-point arithmetic.

First, we ran our implementation of the affine hull algorithm to compute the dimensions of these polyhedra. One goal was to show that we can compute the affine hull of polyhedra associated to MIPs with several dozens – and in some cases also hundreds – of variables. Second, we were curious about how well the affine hull of the mixed-integer hull is actually known just from the model. To be more precise, denote by n' the initially known upper bound on the dimension, i.e., the ambient dimension n minus the number of irredundant equations explicitly stated in the MIP (or -1 if a trivially violated equation is given). The two simple inequalities

$$n' \geq \dim P \geq \dim P_I$$

are clearly valid, and we were curious how often they are actually tight.

Tables 2.2 and 2.3 report the main results of this experiment. For more detailed statistics, e.g., time measurements for different parts of the algorithm, we refer to Tables A.1–A.8. Note that all timings displayed in tables and figures are in seconds.

The numbers showing $\dim P$ (resp. $\dim Q$) are colored red if they differ from n', i.e., if an inequality (or bound) constraint of the relaxation is satisfied with equality by all *LP* solutions. Similarly, the numbers showing $\dim P_I$ (resp. $\dim Q_I$) are colored green if they differ from $\dim P$ (resp. $\dim Q$).

Original instances. We first discuss the qualitative results on the original (i.e., not presolved) instance set. Unfortunately, the dimension of P_I for `p0548` could not be computed, since in one of the oracle calls, the exact MIP solver was unable to close the gap, even though we allowed several additional hours of computation time.

First note that 12 of the 34 instances have no color at all, that is, only the equations specified in the models[13] are valid for the mixed-integer hull P_I. Except for two of these instances, there is in fact no equation at all, i.e., P_I is full-dimensional.

For the 10 instances highlighted in red at least one of the inequalities in the model is an implicit equation, i.e., could have been set to equality keeping the relaxation intact. From a modeling point of view there may be different reasons for that. In addition to improper modeling it could well be that a mathematical model in general has no equations missing, but for specific data it does. Of course due to the nature of the instance set it is hard to find out a specific reason since the underlying models are not available.

Regarding the dimension of P_I it is interesting to see in which range it lies, relative to the dimension of P. The blue bars in Figure 2.8 show this distribution[14] and suggest that the instances fall into two problem classes: Those for which P_I is low-dimensional, even infeasible and those for which P_I's dimension almost equals that of P. The majority of the (admittedly very small) instance set falls into the second class but still has a gap between $\dim P_I$ and $\dim P$. If it is indeed typical that many practical problems have such a gap (but are not "feasibility problems"), then looking for the missing equations using a tool like IPO should be a natural part of model analysis. In fact, IPO could be used to measure the impact of adding such missing equations to the model on the running time of a MIP solver.

Presolved instances. We clearly see much less red in Table 2.3 compared to Table 2.2. In fact, one of the three "red instances" should not have been red: The table reports an initial upper bound of $n' = 0$ for `diamond` which should have been a $n' = -1$ since the presolver had detected the infeasibility. This error occurs because our code does not

[13] Of course all linear combinations of these equations are also valid.

[14] Note that since $\dim P_I$(`p0548`) could not be computed, this instance it is not represented.

Table 2.2: Dimensions for original MIPLIB 2.0 instances.

Instance	n	n'	$\dim P$	$\dim P_I$	t_P	t_{P_I}
air01	771	750	732	617	63.5	49.5
bell3b	133	133	133	115	24.5	4.9
bell5	104	104	104	97	13.2	3.0
bm23	27	27	27	27	1.2	1.7
cracpb1	572	484	484	478	1,918.5	122.0
dcmulti	548	470	470	467	2,969.2	1,420.7
diamond	2	2	2	-1	0.0	0.0
egout	141	68	68	41	2.0	2.5
enigma	100	79	79	3	3.2	109.9
flugpl	18	12	12	9	0.1	3.3
gen	870	720	720	540	28,870.4	620.7
lseu	89	89	89	89	1.0	1.2
misc01	83	68	60	44	3.1	9.8
misc02	59	47	41	37	0.6	1.4
misc03	160	136	121	116	13.1	33.3
misc05	136	108	100	98	24.8	32.2
misc07	260	228	207	204	67.1	141.4
mod008	319	319	319	319	22.2	19.3
mod013	96	83	83	83	1.1	0.9
p0033	33	33	33	27	0.3	1.1
p0040	40	40	30	30	0.4	0.6
p0201	201	201	145	139	6.9	20.9
p0282	282	282	282	282	7.9	5.8
p0291	291	291	291	291	8.5	6.7
p0548	548	548	545		62.4	
pipex	48	32	32	31	0.3	1.4
rgn	180	160	160	160	13.9	3.7
sample2	67	44	44	32	0.4	1.1
sentoy	60	60	60	60	0.4	0.3
stein15	15	15	15	15	0.1	0.1
stein27	27	27	27	27	0.3	0.3
stein45	45	45	45	45	0.7	0.7
stein9	9	9	9	9	0.1	0.1
vpm1	378	336	288	288	17.0	17.1

Table 2.3: Dimensions for presolved MIPLIB 2.0 instances.

Instance	n	n'	$\dim Q$	$\dim Q_I$	t_Q	t_{Q_I}
air01	760	363	363	361	15.7	13.6
bell3b	113	91	91	86	9.4	1.9
bell5	87	56	56	56	0.9	0.8
bm23	27	27	27	27	1.3	1.8
cracpb1	518	478	478	478	1,346.3	108.1
dcmulti	548	469	469	467	3,161.6	368.8
diamond	2	0	-1	−1	0.0	0.0
egout	118	41	41	41	0.9	1.0
enigma	100	79	79	3	3.4	153.1
flugpl	16	10	10	7	0.1	1.8
gen	699	509	411	411	60.2	57.9
lseu	89	85	85	85	1.6	1.1
misc01	82	56	56	44	2.1	7.2
misc02	58	37	37	33	0.3	0.9
misc03	159	115	115	110	9.0	31.2
misc05	128	100	100	98	22.2	31.8
misc07	259	201	201	198	43.8	111.4
mod008	319	319	319	319	20.2	19.6
mod013	96	83	83	83	0.7	1.0
p0033	29	26	26	20	0.3	0.7
p0040	40	20	20	20	0.1	0.1
p0201	201	163	127	127	7.4	12.2
p0282	281	200	200	200	5.0	4.1
p0291	264	67	67	67	1.2	1.1
p0548	527	362	362	357	63.6	175.1
pipex	48	32	32	31	0.5	1.8
rgn	175	160	160	160	7.5	2.8
sample2	55	32	32	32	0.2	0.3
sentoy	60	60	60	60	0.4	0.4
stein15	15	15	15	15	0.1	0.1
stein27	27	27	27	27	0.3	0.3
stein45	45	45	45	45	0.8	0.8
stein9	9	9	9	9	0.1	0.1
vpm1	362	168	168	168	6.0	5.5

check explicitly whether the initial set of equations contains a trivially violated equation. Hence, only the two presolved instances `gen` and `p0201` do not contain all equations necessary to describe the affine hull of the relaxation. This is a sign that in general the presolver does a good job in detecting inequalities that are satisfied with equality by all solutions.

The red bars in Figure 2.8 correspond to the presolved instances. We observe that almost two thirds of the instances have $\dim Q = \dim Q_I$. This shows the effects of the so-called "dual tightening" and "dual fixing" operations, which are the presolve techniques that are allowed to cut off integer-feasible points as long as at least one optimal solution remains feasible[15]. For example, the dimensions of P_I(`air01`) and Q_I(`air01`) are 617 and 361, respectively, that is, many feasible solutions have been cut off.

Large encoding lengths. We now turn to the quantitative results. The most prominent case is clearly instance `gen` for which the computation of $\dim P$ needed more than 8 hours, whereas the computation of $\dim P_I$ was performed in less than 11 minutes. This already indicates that the issue cannot be the running time of the heuristic or the oracle.

Figure 2.9 displays the relative amounts of running time used for the different components of Algorithm 2.4.3 for some instances and some modifications of the algorithm.

Since the heuristic for P(`gen`) only has to solve an LP, the majority of running time is spent for the linear algebra parts, in particular the computation of the direction vectors, the most complicated one having an encoding length of almost 20 million bits. In order to avoid many computations with such huge numbers, IPO heuristically selects the next nonbasic index in Steps 6 and 7 of Algorithm 2.4.2 such that the corresponding direction vector becomes as sparse as possible. Sparsity of the directions is also interesting for the user since sparse equations are typically easier to understand. This is done by computing all directions first in floating-point precision and estimating the number of nonzeros[16]. First, observe that the overhead of this effort is relatively small, as in only very few cases the time spent for the approximate directions is as

[15]The term "dual" comes from the fact that such an operation must take the objective into account.

[16]Experiments suggested to consider numbers smaller than 10^{-7} as zero.

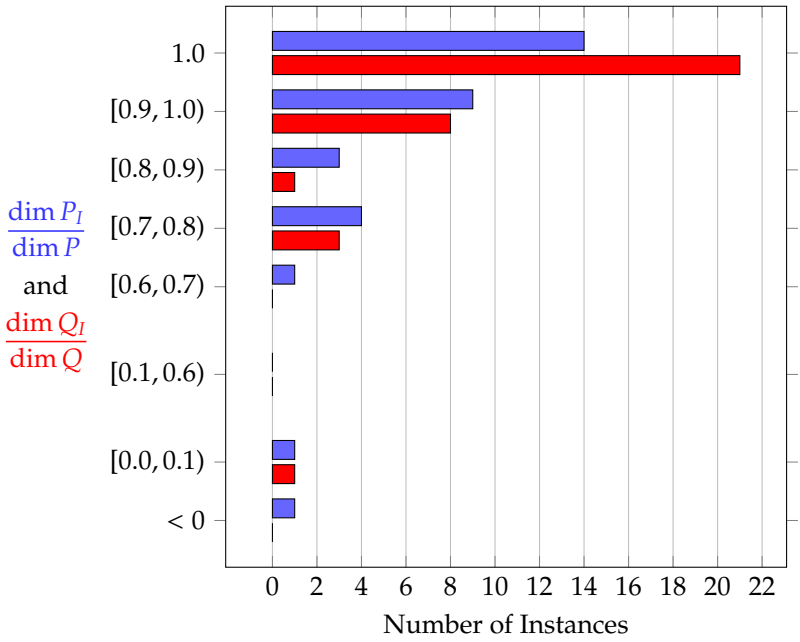
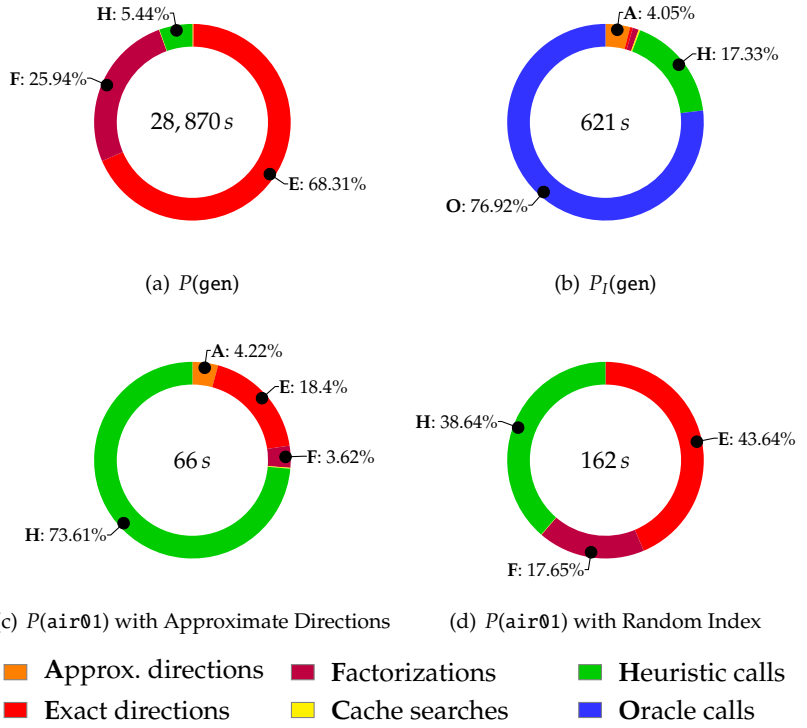


Figure 2.8: Distribution of original and presolved MIPLIB 2.0 instances by dimension ratio of hull and relaxation.

large as the time spent for the single exact direction (see Tables A.1–A.8). Second, the encoding lengths become large much earlier if we do not choose the index carefully. To demonstrate this, we compared the run for P(`air01`) in the current implementation with one where we instead chose the index uniformly at random. In the former case, the sum of the encoding lengths of all computed directions vectors is about 29 million, whereas it is about 56 million for the latter. The effect on the running time for this instance is shown in Figures 2.9(c) and 2.9(d), respectively. Although not visible in the figures, there is even a difference in the running times of the heuristic: Complicated objective vectors also degrade the LP solver's performance, which may happen solely since it has to (pre-)process large numbers, but it could have other

reasons as well.



(a) P(gen) (b) P_I(gen)

(c) P(air01) with Approximate Directions (d) P(air01) with Random Index

Figure 2.9: Different running time distributions for affine hull computation.

Slow oracles. Although encoding lengths can be a limiting factor for our implementation, in many cases the linear algebra is no visible problem, and the algorithm spends most of the time with the heuristic and oracle calls. In general, the quality of a heuristic and its running time relative to that of an oracle will greatly vary across problems and instances. In our case, this variance is enormous: First, all equation verifications were successful, that is, at least for the corresponding objectives our heuristic MIP solver returned a truly optimal solution. Second, our

exact MIP solver ExactSCIP is very slow. Often, finding a primal solution at all takes several minutes, if not hours. This is due to the fact that ExactSCIP has no primal heuristics, i.e., it finds solutions only as LP solutions during branch & bound. Furthermore, since IPO accesses ExactSCIP using an external call, it cannot make use of warm-start features. If linear algebra becomes no issue, then the typical running time profile looks like the one in Figure 2.9(b).

We also tried to compute the affine hull for some instances without a heuristic, but the results are not very surprising: For some easy instances, there is almost no visible effect, for hard instances, it would have taken days to compute the affine hull. Furthermore, since ExactSCIP sometimes has trouble to close the gap[17] or to find a feasible solution, we observe dramatic running-time differences among the oracle calls for single instances already.

2.9.2 Dimensions of Optimal Faces

In addition to the affine hull computations of P, P_I, Q and Q_I we also use IPO to compute the dimensions of the optimal faces of the four polyhedra for the objectives specified in the respective instances. Clearly, for the empty polyhedron associated to instance `diamond`, the optimal face is also empty. In addition to this, several cells in the Tables 2.4 and 2.5 are empty. Due to the maximum allowed time of 1 hour per oracle call, the computations failed for P_I(`bell3b`), P_I(`bell5`), P_I(`dcmulti`), P_I(`egout`), P_I(`p0548`), Q_I(`p0548`) and Q_I(`vpm1`).

Tables 2.4 and 2.5 report the results of this experiment. In particular for the mixed-integer hulls the running times are very large. This is mostly due to the fact that because of small resulting dimensions, the algorithm has to verify many equations, most of them being fixed variables. Despite the simplicity of the objective vectors the exact MIP solver sometimes needs a lot of time. In fact, the dual bound is often optimal very early, but the solver fails to find a feasible solution (which is also optimal if the equation is correct).

A first observation is that, regardless of the instance type, most of the optimal faces are either high-dimensional or very low-dimensional. To

[17] There are cases where SCIP states a gap of 0.0 % and still continues to branch, and we verified that there is still a tiny difference between the primal and dual bounds.

make this more precise we produced Figures 2.11 and 2.10 for which we grouped the instances according to the dimension of the optimal face relative to the dimension of the host polyhedron.

Relaxations. For the linear relaxations P and Q we observe that 6 of the 33 feasible[18] instances have high-dimensional optimal faces, and the remaining 27 instances have optimal faces of dimension less that one third of that of P, or Q, respectively. In particular, 10 (resp. 12) of them are even 0-dimensional, that is, they have a unique LP optimum. Reasons for having high-dimensional optimal faces may be very symmetric models or very simple (e.g., sparse) objectives.

Symmetry may be a disadvantage of an LP model, as for example cutting planes may cut off several LP-optimal solutions, but as long as they do not cut off all of them, there is either no dual improvement, or one needs many cutting planes to make progress. Hence, in order to analyze problems for which such a behavior is observed, a researcher may use IPO to check the dimension of the optimal face of the linear relaxation.

Mixed-Integer Hulls. As mentioned at the beginning of this section we had trouble to compute the dimensions of the optimal faces for several instances. For this reason we only used those 27 instances[19] for which we had results for both, the original and the presolved when creating Figure 2.11. There are again only small differences between the original and the presolved instances. During presolve the polyhedron itself and the optimal face can be cut, reducing the respective dimensions. As expected, there are no serious changes with respect to the dimension ratio, since usually the presolvers cannot reduce the main dimension to the order of magnitude of the optimal face if the latter is rather small. Since there is no general relationship between the optimal faces of linear relaxations and their mixed-integer hulls, it is somewhat interesting that the four instances with largest ratio are also among the group of six instances for which the relaxations had a ratio of more than 70 %. Among the latter are `enigma` whose mixed-integer hull is only 3-dimensional, reducing the ratio to a third, as well as `cracpb1`, for which the optimal face dimensions for the mixed-integer hulls were 202, while those for

[18] We excluded the infeasible instance `diamond`.

[19] In addition to the infeasible instance `diamond` we excluded `bell3b`, `bell5`, `dcmulti`, `egout`, `p0548` and `vpm1`.

Table 2.4: Dimensions of optimal faces for original MIPLIB 2.0 instances.

Instance	n	$\dim F^*(P)$	$\dim P$	$t_{F^*(P)}$	$\dim F^*(P_I)$	$\dim P_I$	$t_{F^*(P_I)}$
air01	771	1	732	40.1	1	617	218.8
bell3b	133	15	133	8.8		115	
bell5	104	10	104	7.7		97	
bm23	27	0	27	0.4	0	27	5.6
cracpb1	572	430	484	794.1	202	478	1,079.2
dcmulti	548	0	470	41.6		467	
diamond	2	0	2	0.0		−1	
egout	141	0	68	2.7		41	
enigma	100	63	79	3.3	1	3	80.1
flugpl	18	0	12	0.2	0	9	9.4
gen	870	35	720	492.8	7	540	51,253.1
lseu	89	11	89	1.8	1	89	277.2
misc01	83	3	60	1.9	5	44	31.9
misc02	59	8	41	1.0	3	37	8.0
misc03	160	18	121	6.7	10	116	249.3
misc05	136	4	100	7.8	17	98	445.3
misc07	260	22	207	21.4	21	204	8,177.5
mod008	319	0	319	11.6	4	319	3,001.6
mod013	96	0	83	1.4	0	83	135.3
p0033	33	12	33	0.6	4	27	4.0
p0040	40	0	30	0.4	0	30	1.9
p0201	201	2	145	6.6	3	139	487.9
p0282	282	0	282	11.6	0	282	702.1
p0291	291	0	291	9.9	0	291	101.1
p0548	548	132	545	55.0			
pipex	48	0	32	0.6	0	31	15.0
rgn	180	36	160	6.4	36	160	251.7
sample2	67	14	44	1.2	0	32	13.2
sentoy	60	0	60	1.5	0	60	34.8
stein15	15	14	15	0.2	14	15	0.1
stein27	27	26	27	0.7	26	27	0.5
stein45	45	44	45	4.7	44	45	121.1
stein9	9	8	9	0.0	8	9	0.1
vpm1	378	28	288	22.1		288	

Table 2.5: Dimensions of optimal faces for presolved MIPLIB 2.0 instances.

Instance	n	$\dim F^*(Q)$	$\dim Q$	$t_{F^*(Q)}$	$\dim F^*(Q_I)$	$\dim Q_I$	$t_{F^*(Q_I)}$
air01	760	0	363	19.0	0	361	96.5
bell3b	113	1	91	2.3	0	86	7,759.8
bell5	87	1	56	1.2	0	56	41.7
bm23	27	0	27	0.3	0	27	5.7
cracpb1	518	424	478	896.9	202	478	1,078.0
dcmulti	548	0	469	39.7	2	467	4,576.4
diamond	2		−1			−1	
egout	118	0	41	1.5	0	41	90.6
enigma	100	63	79	3.4	1	3	70.8
flugpl	16	0	10	0.1	0	7	27.6
gen	699	12	411	59.6	7	411	1,089.2
lseu	89	7	85	1.5	1	85	119.7
misc01	82	3	56	1.4	5	44	28.6
misc02	58	4	37	0.7	3	33	6.1
misc03	159	12	115	3.9	10	110	163.0
misc05	128	4	100	5.1	17	98	172.4
misc07	259	16	201	14.1	21	198	5,452.6
mod008	319	0	319	11.1	4	319	753.3
mod013	96	0	83	1.2	0	83	44.5
p0033	29	6	26	0.3	4	20	1.7
p0040	40	0	20	0.3	0	20	1.4
p0201	201	5	127	7.3	3	127	379.3
p0282	281	0	200	6.6	0	200	418.9
p0291	264	0	67	3.7	0	67	28.1
p0548	527	17	362	37.8		357	
pipex	48	0	32	0.5	0	31	16.7
rgn	175	36	160	6.3	36	160	355.8
sample2	55	2	32	0.7	0	32	7.2
sentoy	60	0	60	1.5	0	60	35.3
stein15	15	14	15	0.2	14	15	0.1
stein27	27	26	27	0.7	26	27	0.6
stein45	45	44	45	4.6	44	45	131.6
stein9	9	8	9	0.1	8	9	0.1
vpm1	362	26	168	14.6		168	

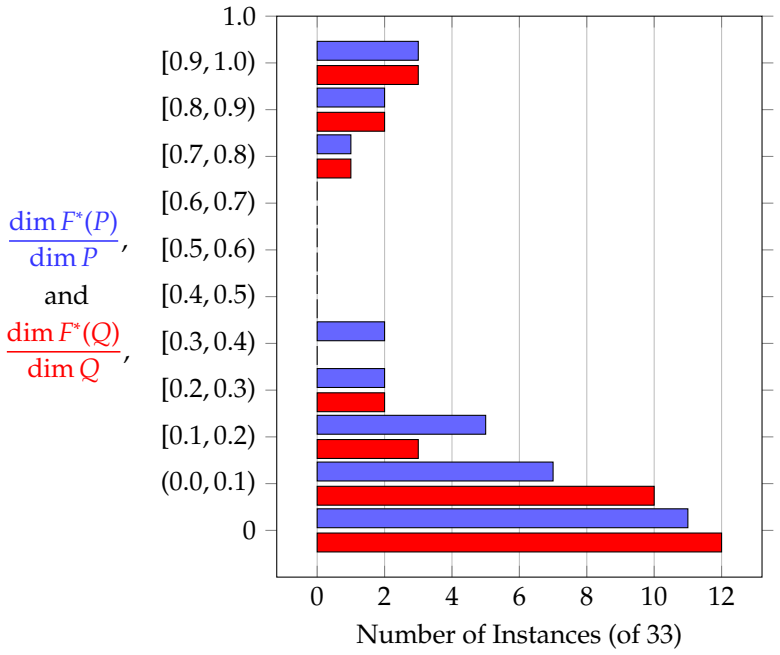


Figure 2.10: Distribution of relaxations of original and presolved MIPLIB 2.0 instances by relative dimension of the optimal face.

the two relaxations were equal to 430 (original) and 424 (presolved), respectively.

Once again, due to the small instance set, we cannot deduce any meaningful relationships, but demonstrated that we can compute optimal faces with our software tool within reasonable time.

2.9.3 Faces Induced by Model Inequalities

Several classical integer programming models from the literature have the property that their inequality constraints are facet-defining. With IPO we are now able to check whether this holds for our benchmark

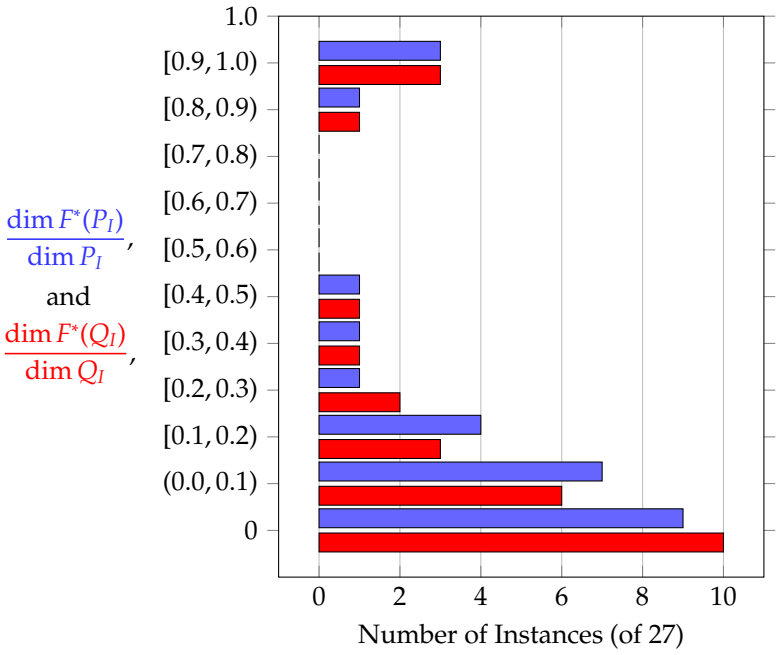


Figure 2.11: Distribution of original and presolved MIPLIB 2.0 instances by relative dimension of the optimal face.

instances, too. For a subset of the previous instances we determined the dimensions of the faces induced by model inequalities. We did not consider bound constraints, but only proper inequalities, which is counter-intuitive at first glance since from a polyhedral point of view bounds are not different from other inequalities. On the other hand, if a bound inequality is not facet-defining and attained by at least one solution, then every strengthening does not lead to a bound, but a "proper" inequality. In this case the model is best-possible in the sense that the modeler could not have set up a better bound (at least without considering the objective).

Starting from the instance set of the previous experiments, we ex-

cluded instances air01 and enigma since they do not have any such inequality constraints, as well as instance diamond since its integer hull is empty. Furthermore, we do not report results on gen, lseu, mod008, p0201, p0282 and p0548 as for them the exact MIP solver exceeded the maximum allowed time of 1 hour per oracle call. As before we ran the experiment for the original and presolved instances.

Results. Figures 2.12 and 2.13 display the results. For every instance, the corresponding bar is divided into parts of different color that show the portion of the instances' inequalities of a certain type. The type is defined by the dimension k of the face F induced by the inequality, relative to the dimension d of the integer hull. Special types are the empty faces ($k = -1$), facets ($k = d - 1$) and equations ($k = d$). The remaining faces are grouped in five groups by the ratio $k/(d-1)$.

The results are clearly non-uniform: While instances mod013, rgn and sentoy only have facet-defining inequality constraints, for P_I(p0033) only 20 % are facet-defining, but 60 % define the empty-face. When we move from original to presolved instances, we observe that the latter have fewer inequalities that define an empty face or an equation. For example, instance p0040 had 3 inequalities of the former, 20 inequalities of the latter type and no further inequalities. After presolve, the 20 equation-defining inequalities are equations and 2 of the 3 remaining inequalities were strengthened to be facet-defining. A similar improvement was obtained for egout. To summarize, we once again observed that SCIP's presolve does a good job.

Cached Points and Directions. As explained in Section 2.4.5, we collect known points and unbounded directions, and check the collection before we optimize over our polyhedra. Although this behavior slightly improves the running time of a single run, its full power is demonstrated in the following experiment. Before starting Algorithm 2.4.3 we check the set of known points (and similarly the set of known unbounded directions) and collect those that satisfy the current inequality with equality. Depending on the inequality and the number of so-far collected points, these can be many, which reduces the number of calls to the heuristic. In fact, sometimes we can start with thousands of points, and have to take care that searching the cache is fast enough. For these situations we developed a floating-point based selection of candidate points, which avoids to compute one scalar product per known point in

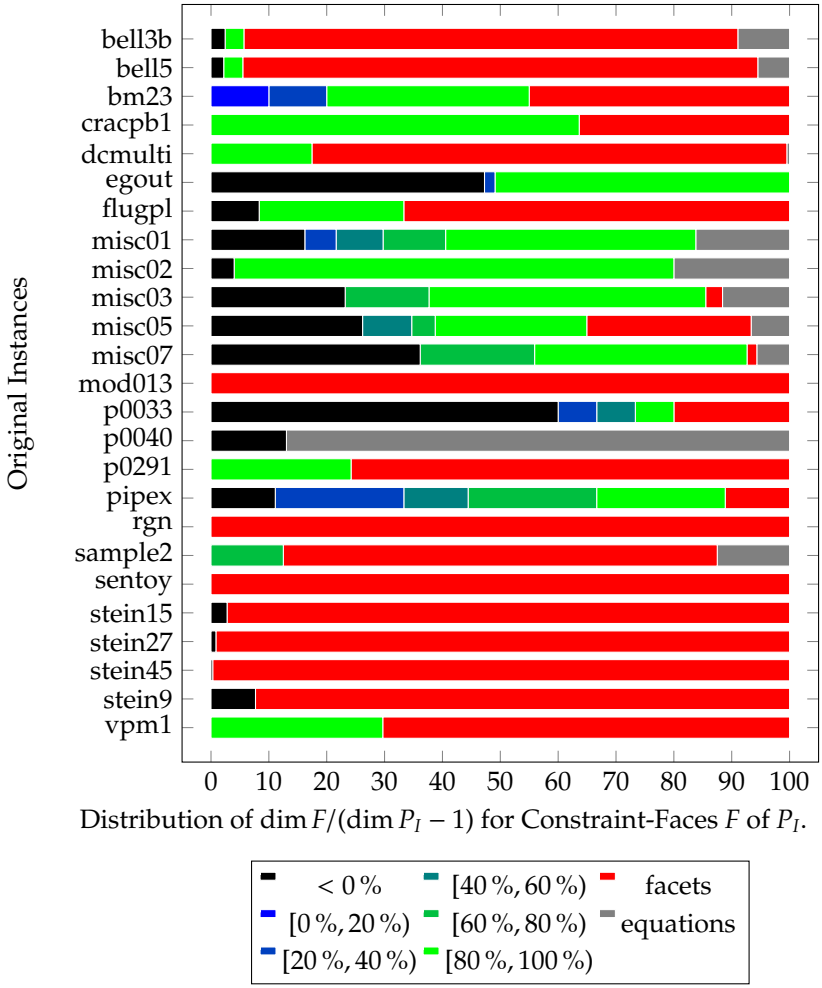


Figure 2.12: Distribution of constraint dimensions for original MIPLIB 2.0 instances.

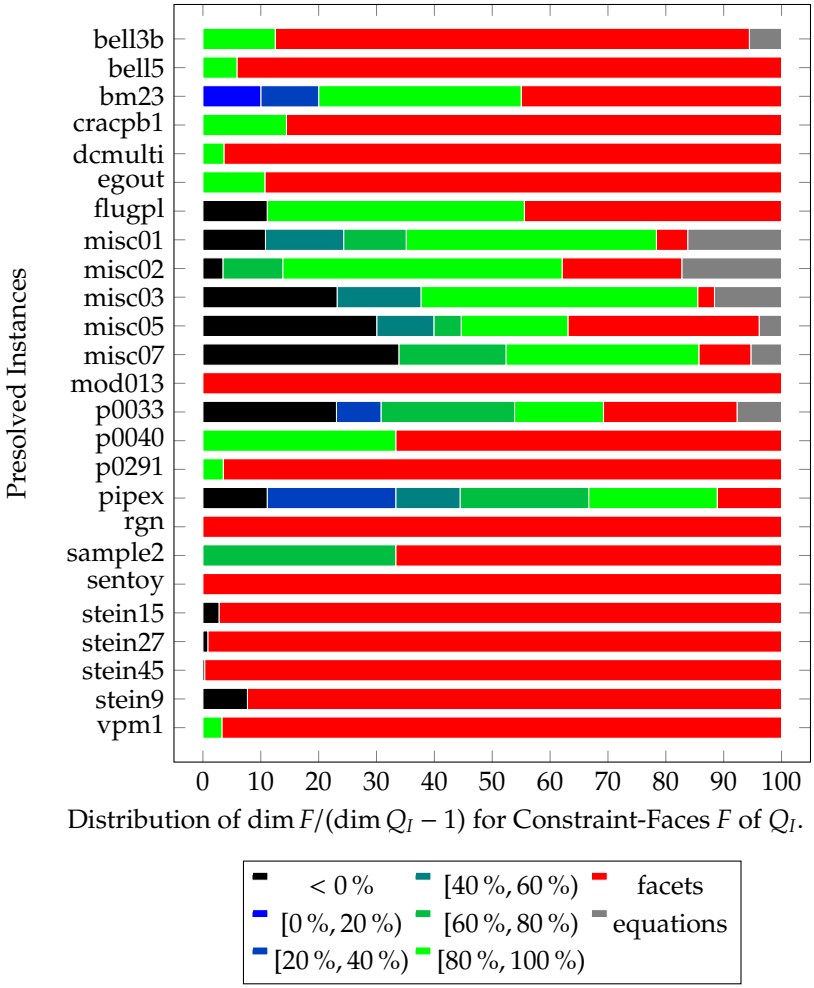


Figure 2.13: Distribution of constraint dimensions for presolved MIPLIB 2.0 instances.

rational arithmetic. We refer to Section 2.4.5 for details.

We analyze the effect for instance P_I(p0291), which has 252 inequality constraints, and for which the computation of the affine hull takes roughly 9 seconds. We selected this instance since heuristic and oracle are fast, and we expected a visible effect due to the rather large number of inequalities and thus a large number of collected points.

We ran IPO without the cache ("Off"), with cache search by exact arithmetic ("Exact") and with the mentioned candidate selection using floating-point arithmetic ("Approximate"). We measured (summing over all affine hull computations) the overall running time t, the time spent for the cache searches t^{cach}, for heuristic calls t^{heur} and for oracle calls t^{orac}, as well as the number of heuristic calls k^{heur}. Table 2.6 shows the results.

Table 2.6: Running times for P_I(p0291) for different caching strategies.

Cache	t	t^{cach}	t^{heur}	k^{heur}	t^{orac}
Off	4858	0	2019	85110	2491
Exact	3660	205	679	30251	2472
Approximate	3563	88	688	30251	2475

The number of oracle calls was always equal to 1559 since the number of verified equations only depends on the heuristically determined dimensions, which were always correct. Hence, the times for the oracle calls (which might have been for different objectives) are almost the same. The cache reduces the number of heuristic calls (and thereby their running time) by two thirds, for only a small cost of 200 s in the exact case. In addition, our floating-point candidate selection furthermore reduces this search time even more without any negative effect: Since the number of heuristic calls agree for the rows "exact" and "approx", we can conclude that our candidate selection always found a suitable point in the cache whenever one existed.

Having in mind the characteristics of this single instance, we want to emphasize that one should not expect such a speed-up in general, but it may be of advantage for some applications.

INVESTIGATING POLYHEDRA BY ORACLES:

2.10 Computational Study: Facets

2.10.1 Matching Polytopes with One Quadratic Term

Let $\hat{e}, \hat{f} \in E$ be two disjoint edges of the complete graph $K_{2n} = (V, E)$ on $2n \geq 4$ nodes, and let $\hat{S} := \hat{e} \cup \hat{f}$ be the set of the four endnodes. In their paper [37], Hupp, Liers and Klein considered the maximum-weight matching problem with one quadratic term, that is, the weight of a matching M is equal to $\sum_{e \in M} w_e$ (for $w \in \mathbb{R}^E$) plus some extra weight $W \in \mathbb{R}$ if $\hat{e} \in M$ and $\hat{f} \in M$ hold at the same time. This can be modeled using an additional variable $y = x_{\hat{e}} \cdot x_{\hat{f}}$ having W as a coefficient in the objective. To obtain a valid IP formulation, the well-known McCormick linearization [51] can be used:

$$y \leq x_{\hat{e}}, \qquad y \leq x_{\hat{f}}, \qquad y \geq x_{\hat{e}} + x_{\hat{f}} - 1 \tag{2.35}$$

Note that this linearization uses the fact that $x_{\hat{e}}$ and $x_{\hat{f}}$ are binary variables. Unfortunately, the resulting LP formulation for the matching problem is not exact anymore, i.e., the polyhedron of interest

$$\begin{aligned} P_{\text{QMP0}} := \text{conv.hull}\{(\chi(M), y) \in \{0,1\}^E \times \{0,1\} \mid &\text{M is a matching,} \\ &\text{and } y = 1 \text{ holds if and only if } \hat{e}, \hat{f} \in M \text{ holds.}\} \end{aligned} \tag{2.36}$$

is not just described by (2.35) and the linear inequalities

$$x_e \geq 0 \qquad \text{for all } e \in E \tag{2.37}$$

$$x(\delta(v)) \leq 1 \qquad \text{for all } v \in V \tag{2.38}$$

$$x(E[S]) \leq \frac{|S|-1}{2} \qquad \text{for all } S \subseteq V \text{ with } |S| \text{ odd and } |S| \geq 3 \tag{2.39}$$

required for the matching polytope (see Edmonds [23]). Let $\hat{e} = \{\hat{u}, \hat{v}\}$ and $\hat{f} = \{\hat{w}, \hat{z}\}$. The authors describe several classes of facets valid for P_{QMP0}:

- (2.37) for all $e \in E \setminus \left\{\hat{e}, \hat{f}\right\}$
- (2.38) for all $v \in V$
- $y \leq x_{\hat{e}}$ and $y \leq x_{\hat{f}}$
- (2.39) for all $S \subseteq V$ with $|S|$ odd, $|S| \geq 3$, and $S \nsubseteq \hat{S}$
- $x(E[S]) + y \leq \frac{|S|-1}{2}$ for all $S \subseteq V$ with $|S|$ odd and $\hat{e}, \hat{f} \in \delta(S)$
- $x(E[S]) + x(E[S \setminus \hat{e}]) + x_{\hat{f}} - y \leq |S| - 2$ for all $S \subseteq V$ with $|S|$ odd, $\hat{e} \in E[S]$ and $\hat{f} \in \delta(S)$
- $x(E[S]) + x_{\hat{v},a} + x_{\hat{w},a} + x_{\hat{z},a} + y \leq \frac{|S|}{2}$ for all $a \in V$ and all $S \subseteq V$ with $|S|$ even with $\hat{v}, \hat{w}, \hat{z} \in S$ and $\hat{u}, a \notin S$ or with $\hat{v}, \hat{w}, \hat{z} \notin S$ and $\hat{u}, a \in S$.
- $x(E[S]) + x(E[\{\hat{u}, \hat{v}, a\}]) + x(E[\{\hat{w}, \hat{z}, b\}]) - y \leq \frac{|S|}{2} + 1$ for all $S \subseteq V \setminus \hat{S}$ with $|S|$ even and all $a, b \in S$.

They also note that the linear description of P_{QMP0} is not complete. In fact in Example 1 in [37] they give a fractional point that satisfies all the above inequalities, but is not in P_{QMP0}. As an argument they provide an objective vector such that the fractional point has objective 11, and they claim that the maximum over P_{QMP0} is equal to 10, which would have proved that the point lies outside the polytope. Unfortunately, the maximum is equal to 12, that is, it does not prove anything. Nevertheless, the objective vector helped us to identify a yet unclassified facet of P_{QMP0}. In order to find such a facet we implemented the IP model

```
param n := 6;
set V := { 1 to n };
set E := { <u,v> in V*V with u < v };
set F := { <1,2>,<3,4>,<1,5>,<2,5>,<3,6>,<4,6>,<1,3>,<2,4> };
var x[E] binary;
var y binary;
maximize weights:
  10*x[1,2] + 10*x[3,4] + 2*x[1,5] + 2*x[2,5] + 2*x[3,6]
  + 2*x[4,6] + 4*x[1,3] + 4*x[2,4] -10*y
  + sum <u,v> in E-F: -1000*x[u,v];
subto degree:   forall <w> in V:
    (sum <u,v> in E with u == w or v == w: x[u,v]) <= 1;
subto product1: y <= x[1,2];
subto product2: y <= x[3,4];
subto product3: y >= x[1,2] + x[3,4] - 1;
```

Figure 2.14: ZIMPL model for the matching problem with one quadratic term in the objective.

```
  Found a new facet: x#1#2 + x#1#3 + x#1#4 + x#3#4 - y <= 1,
  Found a new facet: x#1#2 + x#1#4 + x#2#4 + x#3#4 - y <= 1,
  Found a new facet: x#1#2 + x#1#5 + x#2#5 <= 1,
  Found a new facet: x#3#4 + x#3#6 + x#4#6 <= 1,
  Found a new facet: x#1#2 + x#1#3 + x#1#4 + x#2#3 + x#2#4
    + 2*x#3#4 + x#3#6 + x#4#6 - y <= 2,
```

Figure 2.15: Results for matching problem with one quadratic term in the objective.

with the ZIMPL modeling language, and also incorporated the objective vector from the paper (see Figure 2.14).

We then used IPO to compute facets that are violated by the optimum of the LP relaxation given in the model. Figure 2.15 shows the relevant lines of the output. It is not hard to check that the first four facets in the list belong to the classes mentioned above, and that the last facet does not. Furthermore, it was also not hard to find a whole class of valid inequalities that generalizes this particular facet, and, using IPO again, to compute the dimensions of the corresponding faces for other members of the class. Using this approach we gained some computational evidence that all members of the class are in fact facet-defining. A proof of this fact is about to follow.

A New Class of Facets. We now describe the class of facet-defining inequalities that we found using IPO.

Lemma 2.10.1. *Let $S \subseteq V$ be a subset of nodes of odd cardinality such that $S \cap \hat{S} = \{\hat{w}, \hat{z}\}$ holds. Then the inequality*

$$x(E[S]) + x(E[\hat{S}]) - y \leq \frac{|S| + 1}{2} \tag{2.40}$$

defines a facet of P_{QMP0}.

Proof. Denote by y the binary-valued map that returns for a matching M the corresponding y-value, i.e., $y(M) = 1$ holds if and only if $\hat{e}, \hat{f} \in M$ hold.

We first show that the inequality is valid. Note that the inequality $x(E[S]) + x(E[\hat{S}]) \leq \frac{|S|-1}{2} + 2 = \frac{|S|+1}{2} + 1$ holds by (2.38) and (2.39). Let, for the sake of contradiction, M be a matching such that $x = (\chi(M), y(M))$ does not satisfy (2.40). Hence, $x(E[S]) = \frac{|S|-1}{2}$ and $x(E[\hat{S}]) = 2$ must hold. The second equation implies that $\hat{w}$ and $\hat{z}$ must be matched to nodes inside $\hat{S}$, and the first equation then implies that they must be matched by their common edge, i.e., $\hat{f} \in M$. But since $\hat{u}$ and $\hat{v}$ must be matched, too, $\hat{e} \in M$ holds as well. But this implies $y(M) = 1$, yielding a contradiction to the assumption that (2.40) is violated.

To prove that the inequality is facet-defining, consider the set $\mathcal{M}$ of matchings whose characteristic vectors satisfy (2.40) with equality. Let $F := E[S] \cup E[\hat{S}]$ be the support set of the inequality and $V' := V(F)$ be its

node set. Using the fact that the inequality is equal to $x(F)+x_{\hat{f}}-y \leq \frac{|S|+1}{2}$, it is easy to see that $\mathcal{M}$ contains all matchings M satisfying

$$\begin{array}{ll} \text{(a)} & |M \cap F| = \frac{|S|+1}{2}, \text{ or} \\ \text{(b)} & |M \cap F| = \frac{|S|-1}{2},\ \hat{f} \in M \text{ and } \hat{e} \notin M. \end{array} \tag{2.41}$$

Let $\langle c, x\rangle + \delta y \leq \gamma$ be an inequality that dominates (2.40), that is, it is valid for P_{QMP0} and satisfied with equality by the vectors $(\chi(M), y(M))$ for all matchings $M \in \mathcal{M}$.

We first show that $c_e = 0$ holds for edges $e \in E \setminus F$ by specifying a matching $\hat{M} \in \mathcal{M}$ with $\hat{M} \cup \{e\} \in \mathcal{M}$. We just describe the required properties of $\hat{M}$, and leave the actual constructions to the reader. We distinguish three cases:

- $e \in E[V \setminus V']$: Consider any *near-perfect matching* $\hat{M}$ of F, that is, a matching that matches all nodes except for one. It satisfies $\hat{M} \in \mathcal{M}$ and $\hat{M} \cup \{e\} \in \mathcal{M}$ by (2.41 a).

- $e \in \delta(V')$: Let $a \in e \cap V'$ be the connecting node and consider any perfect matching $\hat{M}$ of $F[V' \setminus \{a\}]$ It satisfies $\hat{M} \in \mathcal{M}$ and $\hat{M} \cup \{e\} \in \mathcal{M}$ by (2.41 a).

- $e \in E[V'] \setminus F$: We can assume $e = \{\hat{u}, a\}$ for some node $a \in S \setminus \{\hat{w}, \hat{z}\}$ since the case of $\{\hat{v}, a\}$ is symmetric. Consider any perfect matching $\hat{M}$ of $F[S \setminus \{a\}]$ that contains $\hat{f}$. It satisfies $\hat{M} \in \mathcal{M}$ and $\hat{M} \cup \{e\} \in \mathcal{M}$ by (2.41 b).

We now show that $c_e = c_f$ holds for all pairs of (distinct) edges in $F \setminus \left\{\hat{f}\right\}$. Since the line graph of $F \setminus \left\{\hat{f}\right\}$ is connected, we only have to show this property for edges that intersect. Denote by $a \in e \cap f$ such a node and by b_e, b_f the endnodes of edges e and f, respectively. To prove the equation it suffices to specify a matching $\hat{M}$ with $\hat{M} \cup \{e\} \in \mathcal{M}$ and $\hat{M} \cup \{f\} \in \mathcal{M}$. Using (2.41 a), one only has to verify the existence of a perfect matching $\hat{M}$ in $F[V' \setminus \{a, b_e, b_f\}]$, since in this case we have $|\hat{M}| + 1 = \frac{|S|-1}{2} + 1$. This step is straight-forward and left to the reader.

We now know that our dominating inequality is of the form

$$x(E[S] \setminus \{\hat{f}\}) + x(E[\hat{S}] \setminus \{\hat{f}\}) + c_{\hat{f}} x_{\hat{f}} + \delta y \leq \gamma.$$

We continue by showing that $c_{\hat{f}} = 2c_{\hat{u},\hat{w}}$ holds. Let $\hat{M}$ be a near-perfect matching of $E[S \setminus \{\hat{w}, \hat{z}\}]$. Since it does not contain $\hat{e}$, by (2.41 b) we get that $\hat{M} \cup \{\hat{f}\} \in \mathcal{M}$ holds. From (2.41 a) we obtain $\hat{M} \cup \{\{\hat{u}, \hat{w}\}, \{\hat{v}, \hat{z}\}\} \in \mathcal{M}$. Hence the two corresponding inequalities are satisfied with equality, and their comparison yields the desired equation $c_{\hat{f}} - c_{\hat{u},\hat{w}} + c_{\hat{v},\hat{z}} = 0$.

Let again $\hat{M}$ be a near-perfect matching of $E[S]$ that contains $\hat{f}$. Since it does not contain $\hat{e}$, by (2.41 b) we get that $\hat{M} \in \mathcal{M}$ holds. From (2.41 a) we furthermore get that $\hat{M} \cup \{\hat{e}\} \in \mathcal{M}$ holds. From $y(\hat{M}) = 0$ and $y(\hat{M} \cup \{\hat{e}\}) = 1$, comparing the corresponding inequalities, we get the equation $c_{\hat{e}} + \delta = 0$.

Hence, up to scaling, the coefficient vector of the dominating inequality is equal to the one of (2.40). This, together with the fact that the right-hand side of (2.40) cannot be decreased further, proves that the inequality is facet-defining. □

It is not clear whether we now have a complete description of P_{QMP0}, and we leave a proof or disproof of this fact to the authors of [37] and to the interested reader.

2.10.2 Edge-Node-Polytopes

Many combinatorial optimization problems on undirected graphs consider certain subsets of nodes or edges. This can be modeled with 0/1-variables in the edge space or in the node space. Sometimes one can generalize them by adding variables of the other type and linking them appropriately. Motivated by this we are interested in certain types of links between edge- and node-subsets. More precisely, for nodes V and edges E of the complete graph on n nodes we are interested in descriptions of the convex hulls of characteristic vectors of pairs $(F, W) \subseteq E \times V$ satisfying a subset of the following restrictions:

(i) $f \in F$ for $f = \{u, v\}$ implies $u, v \in W$.

(ii) $w \in W$ implies $f \in F$ for some edge $f \in \delta(w)$.

Clearly, the variables have to satisfy the trivial bound constraints:

$$0 \leq x_e \leq 1 \qquad \text{for all } e \in E \tag{2.42}$$

$$0 \leq y_v \leq 1 \qquad \text{for all } v \in V \tag{2.43}$$

If we just enforce Constraint (i), then we obtain a system with totally unimodular problem matrix and an integral right-hand side,

$$x_e \leq y_v \qquad \text{for all } v \in e \in E, \tag{2.44}$$

that is, an integral polytope (see Chapter 19 in [62] for a definition and properties). It is known as the *subgraph polytope* (see [18] for an extension and applications).

On the other hand, if we just enforce Constraint (ii) using the obvious inequalities

$$y_v \leq x(\delta(v)) \qquad \text{for all } v \in V, \tag{2.45}$$

then we observe that the resulting polytope is not integral. Its integer hull is a polytope whose faces obtained by fixing all y-variables (to 0 or 1) and by fixing some of the x-variables to 0 are *edge-cover polytopes*, i.e., the convex hulls of characteristic vectors of edge covers. Hence, we call

$$P^{\text{master}}_{\text{edge-cov}}(n) := \text{conv.hull}\{(\chi(F), \chi(W)) \in \{0,1\}^E \times \{0,1\}^V \mid F \cap \delta(w) \neq \emptyset \text{ for all } w \in W\}$$

the *master edge cover* polytope, since by taking appropriate faces one obtains the edge cover polytopes for all graphs with at most n nodes. Using IPO we detected the following class of inequalities valid for $P^{\text{master}}_{\text{edge-cov}}(n)$:

$$y(S) \leq x(\delta(S) \cup E[S]) + \frac{|S|-1}{2} \qquad \text{for all } S \subseteq V \text{ with } |S| \text{ odd} \tag{2.46}$$

As in the previous section, it is not hard to establish that these inequalities are even facet-defining:

Theorem 2.10.2. *Let (V, E) be the complete graph on n nodes. Then Inequality (2.46) is facet-defining for $P^{master}_{edge\text{-}cov}(n)$ for every $S \subsetneq V$ with $|S|$ odd.*

Proof. Let $S \subsetneq V$ be an odd set and define $H := \delta(S) \cup E[S]$. To prove validity of (2.46), consider a feasible point $(x, y) \in \{0,1\}^E \times \{0,1\}^V$. For each $w \in S$ with $y_w = 1$ there must exist an edge $f \in H$ with $x_f = 1$ that can compensate this y-value in the sum $y(S) - x(H)$. Since such an edge has two endnodes, the edges $f \in H$ with $x_f = 1$ compensate at least half

of the overall y-values $y(S)$, that is, $y(S) - x(H) \leq \frac{|S|}{2}$. Since $|S|$ is odd and since the coefficient vector is integral we can round this number down which yields Inequality (2.46).

To prove that the inequality is facet-defining, consider the set $\mathcal{X}$ of feasible pairs $(F, W) \subseteq E \times V$ for which $(\chi(F), \chi(W))$ satisfies Inequality (2.46) with equality. Let $\langle c, x\rangle + \langle d, y\rangle \leq \gamma$ be an inequality that dominates Inequality (2.46) for S, that is, it is valid for $P^{\text{master}}_{\text{edge-cov}}(n)$, and satisfied with equality by the vectors $(\chi(F), y(W))$ for all pairs $(F, W) \in \mathcal{X}$.

We first consider the zero coefficients. Let $v \in V \setminus S$, $w \in S$ and let $e = \{v, w\}$ be the connecting edge. Let $M \subseteq E[S \setminus \{w\}]$ be a perfect matching of the nodes $S \setminus \{w\}$. We have $(M \cup \{e\}, S) \in \mathcal{X}$ as well as $(M \cup \{e\}, S \cup \{v\}) \in \mathcal{X}$, because $\langle \chi(S), \chi(S \cup \{v\})\rangle = \langle \chi(S), \chi(S)\rangle = |S|$ and $\langle \chi(H), \chi(M \cup \{e\})\rangle = |M| + 1 = \frac{|S|+1}{2}$ hold (note that $|S| - \frac{|S|+1}{2} = \frac{|S|-1}{2}$ holds), proving $d_v = 0$.

Also note that this proves $\mathcal{X} \neq \emptyset$, that is, the constant $\frac{|S|-1}{2}$ in (2.46) cannot be decreased. Let $f \in \delta(v) \setminus H$ be an edge incident to v but not to S. By a similar argument, we observe that $(M \cup \{e\}, S) \in \mathcal{X}$ and $(M \cup \{e, f\}, S) \in \mathcal{X}$ holds, proving $c_f = 0$. Next, consider an edge $e = \{v, w\} \in H$ with $v \in S$. Let $M \subseteq E[S \setminus \{v\}]$ be a perfect matching of the nodes $S \setminus \{v\}$. Similar to the above arguments we have $(M \cup \{e\}, S) \in \mathcal{X}$ and $(M, S \setminus \{v\}) \in \mathcal{X}$, proving that $c_e + d_v = 0$ holds. If w is also a node in S, then this proves $d_w = -c_e = d_v$, which in turn yields that $d_u = -c_f$ for all $u \in S$ and all $f \in H$. Hence, c and d are unique up to scaling, that is, the inequality $\langle c, x\rangle + \langle d, y\rangle \leq \gamma$ cannot *strictly* dominate (2.46), which concludes the proof. □

We would also like to mention that optimizing over $P^{\text{master}}_{\text{edge-cov}}(n)$ is an easy problem using a reduction to minimum weight matching: we just have to replace every edge $e = \{u, v\}$ of our graph by a path u-(e, u)-(e, v)-v of length three (adding two nodes (e, u) and (e, v) per edge). It is not too hard to see that the matching polytope of this auxiliary graph projects down to $P^{\text{master}}_{\text{edge-cov}}(n)$ using the projection defined by $x_e = 1 - z_{(e,u),(e,v)}$ for all $e = \{u, v\} \in E$ and $y_v = \sum_{e \in \delta(v)} z_{(e,v),v}$ for all $v \in V$, where z denotes matching-variables for the auxiliary graph.

Hence, a complete description of $P^{\text{master}}_{\text{edge-cov}}(n)$ is conceivable, and since the descriptions of edge-cover polytopes arise as special cases, we believe that the following holds:

Conjecture 2.10.3. *Let (V, E) be the complete graph on n nodes. We then have*

$$P^{master}_{edge\text{-}cov}(n) = \left\{(x, y) \in [0,1]^E \times [0,1]^V \mid (x, y) \text{ satisfies } (2.46)\right\}.$$

Finally, we show that enforcing Constraints (i) and (ii) at the same time (which models that W consists of all nodes of the edges in F, i.e., $W = V(F)$), yields an NP-hard problem. For a given graph $G = (V, E)$, edge weights $w \in \mathbb{R}^E$ and node weights $p \in \mathbb{R}^V$, the *induced-nodeset problem* consists of finding a subset $F \subseteq E$ of edges such that the sum of the edge weights $w(F)$ plus the node weights $p(V(F))$ is minimum. We call $(V(F), F)$ an *induced-nodeset pair*.

Theorem 2.10.4. *The induced-nodeset problem is* NP*-hard.*

Proof. We reduce the *vertex cover problem* that consists of finding, for a given graph $G = (V, E)$, a minimum cardinality *vertex cover* (a subset of nodes $W \subseteq V$ such that every edge in E is incident to at least one of the nodes in W).

Let $G = (V, E)$ be an instance of the vertex cover problem. Consider the graph $\tilde{G} = (V \cup E, \tilde{E})$ with the edge set $\tilde{E} := \{\{v, e\} \mid v \in e \in E\}$, obtained from G by splitting every edge. Define edge weights $p := \mathbb{0}_{\tilde{E}}$ and node weights $q_v := 1$ for all $v \in V$, and $q_e := -|V| - 1$ for all $e \in E$. We claim that the size of the minimum cardinality vertex cover in G is equal to the minimum weight of an induced-nodeset pair for the instance $(\tilde{G}, p, q)$ plus $(|V| + 1) \cdot |E|$.

To this end, let $W \subseteq V$ be a minimum vertex cover, in particular one that contains no isolated nodes. For the node set $W \cup E$ there exists an edge set $F := \{\{w, e\} \mid w \in e \in E \text{ and } w \in W\}$ that induces it. The induced-nodeset pair (W, F) clearly has weight $|W| - (|V| + 1) \cdot |E|$.

Let $(V(F), F)$ be a minimum weight induced-nodeset pair in $\tilde{G}$. It is easy to see that due to the small weights, $V(F)$ must contain all nodes $e \in E$. Hence, for every $\{u, v\} \in E$, F must contain the edge $\{v, \{u, v\}\}$ or the edge $\{u, \{u, v\}\}$ (or both). In the first situation, $V(F)$ contains v, and in the second, $V(F)$ contains u. To summarize, for every edge $e \in E$ of G, $V(F)$ must contain at least one endpoint, that is, $V(F)$ is a vertex cover, whose weight is equal to $|V(F)|$. This concludes the proof, since the construction of the instance $(\tilde{G}, p, q)$ can be done in polynomial time. □

2.10.3 Tree Polytopes

In this subsection we consider the tree polytopes for complete graphs $K_n = (V_n, E_n)$ on n nodes, defined as

$$P_{\text{tree}}(n) := \text{conv.hull}\{(\chi(F), \chi(W)) \in \{0,1\}^{E_n} \times \{0,1\}^{V_n} \mid (W, F) \text{ is a tree }\},$$

where we consider pairs $(\{v\}, \emptyset)$ for nodes $v \in V_n$ and also $(\emptyset, \emptyset)$ as trees. Defining the pair of empty sets as a tree has not much effect, except that it makes the polytope full-dimensional and hence the representation of facets unique up to scaling. Such polytopes are interesting since the faces obtained by setting some edge-variables to 0 and some node-variables to 1 can be projected down to *Steiner tree* polytopes. Furthermore, the well-known relaxation

$$\begin{aligned} x(E[S]) &\leq y(S \setminus \{s\}) && \text{for all } s \in S \subseteq V && (2.47)\\ x(E) &\geq y(V) - 1 && && (2.48)\\ x_e &\geq 0 && \text{for all } e \in E && (2.49)\\ y_v &\leq 1 && \text{for all } y \in V && (2.50) \end{aligned}$$

is very strong, at least for solving Steiner tree problems. We refer to [32] for a theoretical comparison of relaxations and to [46] for a computational study in which an equally strong model is used. The face of this relaxation obtained by fixing a set of edge-variables to 0 such that the remaining ones form a 2-tree (a graph constructed from a single edge by successively adding nodes and connecting them to precisely two of the previous nodes), is integral [49].

Clearly, optimizing over $P_{\text{tree}}(n)$ is NP-hard since it subsumes solving Steiner tree problems. In a moment we will present a well-known hardness reduction from the vertex cover problem that is "polyhedral", i.e., we can find for every graph $G = (V, E)$ a number n that is polynomial in $|V|$ and $|E|$ such that a certain face of $P_{\text{tree}}(n)$ projects down to the vertex cover polytope[20] of G. Having the strength of the relaxation in mind, this is a win-win situation in the following sense: Either the projection of the relaxation yields a relaxation of the vertex cover polytope that is also strong in practice or we obtain an objective vector whose maximization over (2.47)–(2.50) yields a fractional point. In the latter

[20] The vertex cover polytope is the convex hull of characteristic vectors of vertex covers

case we can use IPO to learn a new facet. Note that Bazzi et al. [12] recently proved that no polynomial-size linear relaxation approximates the vertex cover polytope very well in every direction. Our relaxation is certainly not of polynomial size, but the same reduction works for the Steiner arborescence polytope, which has a polynomial-size extended formulation using flow variables (see [32]) and yields equally strong LP solutions. Hence, the mentioned negative result also applies to tree polytopes.

We now describe the mentioned reduction. For a graph $G = (V, E)$ we construct an auxiliary graph $\tilde{G} = (\tilde{V}, \tilde{E})$ by replacing every edge by a 2-path and adding a root node r that is connected to all original nodes, and define a set $\tilde{T}$ of "terminal nodes".

$$\begin{aligned}\tilde{V} &:= V \cup E \cup \{r\}\\ \tilde{E} &:= \{\{v, e\} \mid v \in e \in E\} \cup \{\{v, r\} \mid v \in V\}\\ \tilde{T} &:= E \cup \{r\}\end{aligned}$$

We claim that for every tree $(\tilde{W}, \tilde{F})$ in $\tilde{G}$ satisfying $\tilde{W} \supseteq \tilde{T}$ the set $\tilde{W} \cap V$ is a vertex cover of G. To prove this, we just have to convince ourselves that every edge $e \in E$ is covered. Clearly, since every split node e of $\tilde{G}$ is a terminal, i.e., $e \in \tilde{T}$ holds, at least one of e's endnodes must be part of the tree.

Furthermore, for every vertex cover $W \subseteq V$ of G we consider the set $\tilde{W} := W \cup \tilde{T}$ and observe that $\tilde{G}[\tilde{W}]$ is connected. This proves that there exists a tree in $\tilde{G}$ whose node set is equal to $\tilde{W}$.

By embedding $\tilde{G}$ into the complete graph on $n = |\tilde{V}|$ nodes we get that the tree polytope $P_{\text{tree}}(n)$ has a face (obtained by fixing edge-variables for edges not in $\tilde{E}$ to 0 and node-variables for nodes in $\tilde{T}$ to 1) that projects to the vertex cover polytope of G. Clearly, the map is just the projection onto the node-variables for nodes in V.

We implemented a separation routine for Inequalities (2.47), based on the fact that the separation problem can be solved by optimizing over the *subgraph polytope* defined in Section 2.10.2. Using SCIP and IPO we computed facets of the image of the relaxation's projection onto the y-variables.

Unfortunately, computational evidence suggests that the projection always yields a very weak relaxation for the vertex cover polytopes,

namely the one consisting of just the inequalities $y_u + y_v \geq 1$ for every edge $\{u, v\} \in E$. Hence we considered complete graphs G and the strong *clique inequalities* $y(V) \geq |V| - 1$. Since they are not valid for the projection of the relaxation, their lifted version is also not valid for the tree polytope. Thus we were able to use the corresponding normal vector as an objective for IPO's facet computation.

For the clique with 3 nodes we obtained the following facet of P_{tree} (7) (with nodes indexed from 1 to 7):

$$(x_{1,4} + x_{4,2}) + (x_{1,5} + x_{5,3}) + (x_{2,6} + x_{6,3}) + x(E[\{1,2,3,7\}]) \leq 2y_1 + 2y_2 + 2y_3 + y_7$$

For the clique with 4 nodes, IPO found more than a dozen facets of P_{tree} (11), already with different types of coefficient patterns. Two of them are very similar to the above inequality:

$$(x_{1,4} + x_{4,2}) + (x_{1,5} + x_{5,3}) + (x_{2,6} + x_{6,3}) + x(E[\{1,2,3,7,8,9,10\}]) \leq 2y_1 + 2y_2 + 2y_3 + y_7 + y_8 + y_9 + y_{10}$$

$$(x_{1,4} + x_{4,2}) + (x_{1,5} + x_{5,3}) + (x_{2,6} + x_{6,3}) + x(E[\{1,2,3,7,8,9,10,11\}]) \leq 2y_1 + 2y_2 + 2y_3 + y_7 + y_8 + y_9 + y_{10} + y_{11}$$

This pattern suggests that the core of these inequalities is

$$(x_{1,4} + x_{4,2}) + (x_{1,5} + x_{5,3}) + (x_{2,6} + x_{6,3}) + x(E[\{1,2,3\}]) \leq 2y_1 + 2y_2 + 2y_3.$$

Taking one more computed inequality

$$(x_{1,5} + x_{5,2}) + (x_{1,6} + x_{6,3}) + (x_{1,7} + x_{7,4}) + (x_{2,8} + x_{8,3}) + (x_{2,9} + x_{9,4}) + (x_{3,10} + x_{10,4}) + x(E[\{1,2,3,4\}]) \leq 3y_1 + 3y_2 + 3y_3 + 3y_4$$

into account we can conjecture the following:

Conjecture 2.10.5. *Let $n, k, \in \mathbb{N}$ and $\ell \in \mathbb{Z}_+$ with $k \geq 2$ and $k + \ell + \binom{k}{2} \leq n$. Let $v_1, \ldots, v_k$, $u_1, \ldots u_\ell$, and $w_{i,j}$ for $i, j \in [k]$ with $i < j$ be distinct nodes of*

K_n. Let $U := \{u_1, \dots, u_\ell\}$. Then the inequality

$$\sum_{i,j \in [k], i<j} (x_{v_i,w_{i,j}} + x_{w_{i,j},v_j}) + x(E[\{v_1, \dots v_k\} \cup U]) \leq (k-1) \cdot y(\{v_1, \dots, v_k\}) + y(U)$$

is valid and facet-defining for $P_{tree}(n)$.

Note that for $k = 1$ and $\ell \in \mathbb{N}$ this yields an inequality of type (2.47). We conclude this section by leaving the proof (or disproof) to the interested reader.

INVESTIGATING POLYHEDRA BY ORACLES:

2.11 Computational Study: Adjacency

In this section we consider the symmetric traveling salesman polytope for the complete graph $K_n = (V, E)$ on n nodes (for $n \geq 3$), defined as

$$P_n := \text{conv.hull}\left\{\chi(T) \in \{0,1\}^E \mid T \text{ is a tour in } K_n\right\},$$

where tours are cycles that visit every node exactly once. Using appropriate heuristics and oracles we show how to test two tours (Hamiltonian cycles) for adjacency quickly using Algorithm 2.6.1 from Section 2.6.1. This work is motivated by the question regarding the diameter of (the 1-skeleton of) P_n, raised by Grötschel and Padberg [34] in 1985. We do not make direct progress on this question, but make an observation on the existence of adjacencies that do not belong to a certain class.

2.11.1 Heuristics and Oracles for TSP Polytopes

The first oracle we use is the famous software concorde [4]. With a simple Python[21] wrapper that creates the instance file for each oracle call we have access to this very fast oracle. Note that, since the traveling salesman problem is usually stated as a minimization problem, we use this notion as well, although in practice IPO's oracles only maximize.

Unfortunately, concorde relies on 32-bit integer arithmetic and can thus handle only integral objectives of small encoding length. In or-

[21] Python, available at `http://www.python.org/`

der to still allow correct behavior, we created another oracle that, if the encoding lengths are small, calls concorde, and otherwise runs an enumeration-based exact algorithm in exact arithmetic. An efficient implementation (using limited precision) was available to us as a courtesy of William J. Cook. Since the enumeration algorithm needs good primal bounds, that is, tours with good objective value, for effective pruning, it runs a heuristic beforehand. We turned this heuristic into an IPO heuristic that allows us to speed-up our affine hull and adjacency computations. The details are as follows:

Heuristic. For small numbers of nodes, a very simple heuristic works quite well. Starting from some node s, a tour is built by always choosing the nearest neighbor. Then, 2-opt local improvement steps, modifying the tour by replacing two edges by two cheaper edges, are carried out as long as possible. This is repeated for each possible starting node s, and the best tour is taken.

Enumeration. The exact algorithm consists of a routine that, given a simple path W in K_n and a primal bound $\delta \in \mathbb{R}$, enumerates all tours in K_n that contain W as a subpath. It does this by connecting W's endnode t to every other node v that is not already covered by W, and calls itself recursively with the path $W \cup \{t, v\}$. In order to avoid exploration of the whole search space, the algorithm stops searching for path W if the weight of W plus the weight of the minimum-weight tree that spans the nodes outside of W and the two endnodes of W is greater than or equal to δ. This is correct since every tour $T \supseteq W$ induces such a tree, and thus, the weight of $T \setminus W$ is at least as large as the minimum value. If the routine reaches a state where W covers all nodes, then the endnodes s, t are connected and the resulting weight δ' is compared to δ. If $\delta' < \delta$ holds, δ is replaced by δ' and the best known tour is replaced by $W \cup \{s, t\}$. In order to avoid spanning tree computations for the same node set, a hash is used that maps the set of spanned nodes to the corresponding weight.

2.11.2 Experiment & Results

We use our tool to estimate the degrees of P_n's vertices, which are all the same (for a certain n) for symmetry reasons.

Our experiment is designed as follows. First, we fix a tour T_0. Second, for $N = 10000$ repetitions, we sample a tour $T \neq T_0$ in K_n uniformly at random, and check whether it is adjacent to T_0. We then count the number k of positive answers, that is, detected edges.

In order to get a sense of how precise the approximations of the degrees actually are, we apply some standard techniques from statistics. As we select the candidate tours independently uniformly at random, we have a binomial distribution X for N experiments with a success probability p that is equal to the relative degree of each tour. Clearly, our best estimate for p is the fraction $\hat{p} := k/N$. The expected value of X is equal to $p \cdot N$, and its standard deviation equals $\sigma := \sqrt{p(1-p)/N}$. We can thus approximate $\frac{X-pN}{\sigma}$ by the normal distribution with mean 0 and standard deviation 1. For the latter we know the confidence intervals, from which we can compute the error intervals for X for a certain confidence. The resulting interval for a confidence level of 99 % (with so-called Z-value 2.5759) is thus $p \pm 2.5759\sqrt{\frac{p(1-p)}{N}}$. If we approximate p by $\hat{p}$, we obtain for the *relative* error the following formula:

$$\varepsilon^{\text{rel}}_{99\%}(k,N) \approx \frac{2.5759 \cdot \sqrt{\frac{\hat{p}\cdot(1-\hat{p})}{N}}}{\hat{p}} = 2.5759 \cdot \sqrt{\frac{N-k}{kN}} \tag{2.51}$$

Table 2.7 shows the results of this experiment, where τ is the number of tours that were computed[22] during the run of IPO and $\nu_n = \frac{1}{2}(n-1)!$ denotes the number of all tours. The column labeled t shows the time *per iteration* (in seconds), and the columns labeled $t^{\text{LP}}, t^{\text{heur}}, t^{\text{orac}}, t^{\text{cach}}$ denote the proportions used for the LP computations, heuristic calls, oracle calls and cache searches, respectively. Furthermore, $\varepsilon^{\text{rel}}_{99\%}$ was computed by (2.51).

A first observation is that the relative degree $k/10^5$ decreases, but quite slowly. In particular, it seems that k won't get below $\sqrt{\nu_n}$, which would have been a simple reason for the diameter of P_n to be greater than 2 (see [25]), disproving a conjecture by Grötschel and Padberg. We can trust the values up to a certain error: For example, assume that the relative degree for $n = 15$ is indeed equal to 0.59 %, then in 99 % of the

[22] Note that the tours corresponding to the input vertices are not considered here.

Table 2.7: Adjacency computation for random vertex pairs of TSP polytopes.

n	$k/10^5$	τ	ν_n	t	t^{LP}	t^{heur}	t^{orac}	t^{cach}	$\varepsilon^{\text{rel}}_{99\%}$
5	91.23 %	12	$1.2 \cdot 10^1$	0.3 *s*	0.5 %	0.1 %	97.9 %	0.1 %	0.80 %
6	69.32 %	45	$6.0 \cdot 10^1$	0.4 *s*	0.7 %	0.1 %	97.5 %	0.1 %	1.71 %
7	46.16 %	207	$3.6 \cdot 10^2$	0.6 *s*	1.1 %	0.1 %	96.1 %	0.5 %	2.78 %
8	28.07 %	1,189	$2.5 \cdot 10^3$	0.8 *s*	1.5 %	0.1 %	93.1 %	2.5 %	4.12 %
9	17.46 %	5,759	$2.0 \cdot 10^4$	1.0 *s*	2.0 %	0.2 %	86.1 %	8.7 %	5.60 %
10	10.52 %	15,472	$1.8 \cdot 10^5$	1.5 *s*	2.3 %	0.2 %	77.5 %	17.3 %	7.51 %
11	6.53 %	33,935	$1.8 \cdot 10^6$	2.1 *s*	2.7 %	0.2 %	67.4 %	26.9 %	9.75 %
12	3.67 %	66,510	$2.0 \cdot 10^7$	3.0 *s*	3.8 %	0.3 %	54.2 %	38.8 %	13.20 %
13	2.20 %	125,298	$2.4 \cdot 10^8$	4.9 *s*	5.1 %	0.3 %	39.6 %	52.0 %	17.17 %
14	1.13 %	232,995	$3.1 \cdot 10^9$	10.1 *s*	7.7 %	0.3 %	22.9 %	65.8 %	24.09 %
15	0.59 %	406,315	$4.4 \cdot 10^{10}$	24.3 *s*	12.9 %	0.2 %	11.0 %	71.6 %	33.43 %

repetitions of the experiment we would get a percentage that could be a third smaller or larger, that is, between 0.39 % and 0.79 %.

Second, its is interesting to see how many tours were actually computed in order to do the adjacency tests. As expected, the τ-values increase for larger n, but not as rapidly as the number of vertices ν_n of the polytopes. Note that for $n = 15$, the ratio τ/ν_n is already in the order of 10^{-5}.

Third, it is worth noting that due to the large number of collected points, the running time for the cache search increased dramatically. Although we do not report the details here, we checked the actual number of invocations, and observed that the cache slowed down the later computations, since searching takes a lot of time, whereas the running time of the heuristic is constant (for fixed number of cites).

2.11.3 Adjacent Tours with Common Edges

We now present a structural result on adjacencies of the TSP polytope, which is unrelated to the computations except that we were able to derive it after inspecting several pairs of adjacent tours that were produced by our implementation.

We start with a sufficient criterion for adjacency that the TSP polytope inherits from the perfect 2-matching polytope since every tour is also a perfect 2-matching.

Proposition 2.11.1 (Chvátal [17]). *If for two vertices $\chi(T), \chi(T') \in \{0,1\}^{E_n}$ of P_n, the edge set $T \Delta T'$ is a simple (alternating) cycle, then $\chi(T)$ and $\chi(T')$ are adjacent vertices of P_n.*

There were several unsuccessful attempts to characterize adjacency for P_n, until Papadimitriou established the following hardness result, indicating that we shouldn't expect the existence of (easily verifiable) certificates for arbitrary adjacencies in the TSP polytopes.

Theorem 2.11.2 (Papadimitriou [53]). *The problem to decide adjacency for two given vertices of P_n is* coNP*-complete.*

In order to prove upper bounds on the diameter of P_n we do not need characterizations, but only strong sufficient conditions for adjacency. Using our implementation, we searched for adjacencies that are not of the type stated in Proposition 2.11.1. We only found adjacent tours T, T' with common edges, i.e., $T \cap T' \neq \emptyset$, tried to find structural reasons for the adjacency, and observed the result stated in the next theorem.

Let $G = (V, E)$ be a graph. For an edge $e \in E$ we denote by G/e the graph we obtain by contracting edge e, i.e., by identifying e's endnodes, removing loops and merging double edges. Accordingly, for an edge set $F \subseteq E$, F/e denotes the contracted edge set.

Theorem 2.11.3. *Let T_1 and T_2 be two tours in K_n with a common edge $e = \{u, v\} \in T_1 \cap T_2$. If $\chi(T_1/e)$ and $\chi(T_2/e)$ are adjacent vertices of P_{n-1}, then*

(a) either $\chi(T_1)$ and $\chi(T_2)$ are adjacent vertices of P_n,

(b) or there exist nodes $x, y \in V$ such that T_1 and T_2 contain the paths x-u-v-y and x-v-u-y, respectively.

The theorem essentially tells us that if we start with an adjacent pair of tours and split a node, then we can reassign the incident edges in a certain way to obtain an adjacent pair again. On the other hand, reassigning them in a "crossing" way leads to a nonadjacent pair (see Figure 2.16).

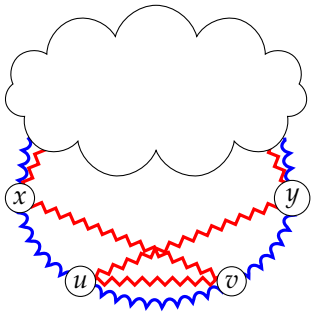


Figure 2.16: Illustration of Case (b) of Theorem 2.11.3.

Proof. We can identify K_n/e with K_{n-1} in a natural way, and hence consider T_1/e and T_2/e as tours in K_{n-1}. Throughout the proof we will denote by $u\&v$ the node to which e was contracted in K_n/e.

We consider the face $F := \{x \in P_n \mid x_e = 1\}$ of P_n and the projection map $\pi : \mathbb{R}^E \to \mathbb{R}^{E/\{u,v\}}$ defined via

$$\pi(x)_e = \begin{cases} x_e & \text{if } u\&v \notin e \\ x_{w,u} + x_{w,v} & \text{if } e = \{w, u\&v\}\,. \end{cases}$$

It is not hard to see that π projects vertices of F to vertices of P_{n-1}, i.e., considering tours in K_n that contain e, and interpreting them as tours in K_n/e. In fact every vertex of F is projected to a vertex of P_{n-1} this way.

We claim that for any tour T in K_n/e there are precisely two tours T' in K_n for which $\pi(\chi(T')) = \chi(T)$ holds. To see this, let $W = x\text{-}(u\&v)\text{-}y$ be the 2-path in T with $u\&v$ as the inner node. Then T' must be equal to one of the tours

$$(T \setminus W) \cup \{\{x, u\}, e, \{v, y\}\} \text{ and } (T \setminus W) \cup \{\{x, v\}, e, \{u, y\}\}\,, \tag{2.52}$$

since $e \in T'$ must hold and there are only these two possibilities to properly connect e to $T \setminus W$. Hence the preimage of $\chi(T)$ under π (intersected with P_n) is an edge of F whose vertices correspond to the two possible tours T'.

From the latter (observing that $\pi : F \to P_{n-1}$ maps vertices to vertices) we get that the preimage of our given edge conv.hull $\{\chi(T_1/e), \chi(T_2/e)\}$

under π (again intersected with P_n) is a polytope G with four vertices. These vertices must be the characteristic vectors of T_1, T_1', T_2 and T_2', where T_1' and T_2' are tours such that the contraction of e maps T_1, T_1' to T_1/e, and T_2, T_2' to T_2/e, respectively.

We will now prove that if (b) holds, then (a) cannot hold. So assume that $T_1 = W_1 \cup \{\{x,u\}, e, \{v,y\}\}$ and that $T_2 = W_2 \cup \{\{x,v\}, e, \{u,y\}\}$ hold. From $T_1/e = T_1'/e$ we obtain $T_1' = W_1 \cup \{\{x,v\}, e, \{u,y\}\}$, while from $T_2/e = T_2'/e$ we obtain $T_2' = W_2 \cup \{\{x,u\}, e, \{v,y\}\}$. This already proves $W_1 \neq W_2$ since otherwise $T_1 = T_2'$ would contradict the fact that the four tours correspond to distinct vertices of G. Counting edges immediately yields

$$\frac{1}{2}\chi(T_1) + \frac{1}{2}\chi(T_2) = \frac{1}{2}\chi(T_1') + \frac{1}{2}\chi(T_2'), \tag{2.53}$$

which proves the nonadjacency of $\chi(T_1)$ and $\chi(T_2)$.

Now suppose (a) does not hold, that is, $\chi(T_1)$ and $\chi(T_2)$ are not adjacent. Since the only polytopes with four vertices are 3-simplices and quadrilaterals, and since simplices have no non-adjacencies, G must be a quadrilateral. Because G is also a 0/1-polytope, its diagonals intersect in their barycenters, that is, Equation (2.53) holds, and in particular $T_1 \Delta T_2 = T_1' \Delta T_2'$ must hold. From (2.52) we know that $T_1 \Delta T_1'$ and $T_2 \Delta T_2'$ are 4-cycles having e as a chord. Now (2.53) implies that the cycles must be the same, i.e., $T_1 \Delta T_1' = T_2 \Delta T_2'$. Let x and y be the nodes of this cycle that are not incident to e, ordered such that T_1 contains the path x-u-v-y. From (2.52) we obtain that T_1' contains the path x-v-u-y. In particular, the edge $\{x,u\}$ is contained in $T_1 \setminus T_1'$, and once again by (2.53), $\{x,u\}$ cannot be contained in T_2. Hence, T_2 must contain the path x-v-u-y. □

There are adjacencies that are not inherited from the perfect 2-matching polytope, but can be explained using Proposition 2.11.1 and Theorem 2.11.3. One may be brave to conjecture that maybe all adjacencies of the TSP polytope are either of 2-matching type or stem from those by splitting. But once again, since a series of contractions would be a short certificate, the hardness result in Theorem 2.11.2 states that we should not expect such a result. In other words we might expect that there exist pairs of adjacent vertices $\chi(T_1)$ and $\chi(T_2)$ of P_n (for some $n \in \mathbb{N}$) that are not adjacent in the 2-matching polytope, and no mat-

ter which edge we contract, the resulting tours do not correspond to adjacent vertices of P_{n-1}.

In order to find such a pair, we extended our implementation in the obvious way. For every adjacent pair $\chi(T_1)$, $\chi(T_2)$ and every edge $e \in T_1 \cap T_2$ we check whether $\chi(T_1/e)$ and $\chi(T_2/e)$ are adjacent in P_{n-1} using a second set of oracles and heuristics. Using this method we found that the tour $(1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 7 \to 8 \to 1)$ together with the tour $(1 \to 4 \to 3 \to 8 \to 7 \to 2 \to 6 \to 5 \to 1)$, depicted in Figure 2.17, have this property.

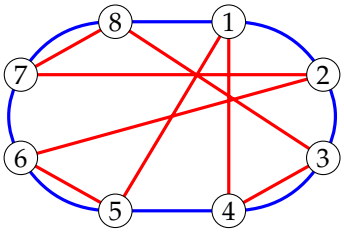


Figure 2.17: Pair of TSP tours that cannot be obtained from a 2-matching adjacency by splitting.

Contracting the common edges $\{3,4\}$, $\{5,6\}$, $\{7,8\}$ yields pairs of vertices whose smallest containing faces have dimensions 2, 3 and 2, respectively.

Chapter 3

Analyzing Simple Extensions of Polytopes

ANALYZING SIMPLE EXTENSIONS OF POLYTOPES:

3.1 Introduction

With respect to both structural and algorithmic aspects, linear optimization over a polytope P can be replaced by linear optimization over any (usually higher dimensional) polytope Q of which P can be obtained as the image under a linear map (which we refer to as a *projection*). Such a polytope Q (along with a suitable projection) is called an *extension* of P.

Defining the *size* of a polytope as its number of facets, the smallest size of any extension of the polytope P is known as the *extension complexity* $\mathrm{xc}(P)$ of P. It has turned out in the past that for several important polytopes related to combinatorial optimization problems the extension complexity is bounded polynomially in the dimension, although outer descriptions in the original space require exponentially many inequalities. One of the most prominent examples is the spanning tree polytope of the complete graph K_n on n nodes, which has extension complexity $O(n^3)$ [50].

After Rothvoss [58] showed that there are 0/1-polytopes whose extension complexities cannot be bounded polynomially in their dimensions, in 2012, Fiorini, Massar, Pokutta, Tiwary and de Wolf [27] could prove that the extension complexities of some concrete and important examples of polytopes like traveling salesman polytopes cannot be bounded polynomially. Similar results have then also been deduced for several other polytopes associated with NP-hard optimization problems, e.g., by Avis and Tiwary [9] and Pokutta and van Vyve [55]. Rothvoss [59] recently showed that also the perfect matching polytope of the complete graph (with an even number of nodes) has exponential extension

complexity, thus exhibiting the first polytope with this property that is associated with a polynomial-time solvable optimization problem.

The first fundamental research with respect to understanding extension complexities was Yannakakis' seminal paper [69] of 1991. Observing that many of the nice and small extensions that are known (e.g., the polynomial size extension of the spanning tree polytope of K_n mentioned above) have the nice property of being symmetric in a certain sense, he derived lower bounds on extensions with that special property. In particular, he already proved that both perfect matching polytopes as well as traveling salesman polytopes do not have polynomial size *symmetric* extensions.

It turned out that requiring symmetry in principle actually can make a huge difference for the minimum sizes of extensions (though nowadays we know that this is not really true for traveling salesman and perfect matching polytopes). For instance, Kaibel, Theis and Pashkovich [40] showed that the polytope associated with the matchings of size $\lfloor \log n \rfloor$ in K_n has polynomially bounded extension complexity although it does not admit symmetric extensions of polynomial size. Another example is provided by the permutahedron, which has extension complexity $\Theta(n \log n)$ [31], while every symmetric extension of it has size $\Omega(n^2)$ [54].

These examples show that imposing the restriction of symmetry may severely influence the smallest possible sizes of extensions. In this chapter, we investigate another type of restrictions on extensions, namely the one arising from requiring the extension to be a non-degenerate polytope. A d-dimensional polyhedron is called *simple* if every vertex is contained in exactly d facets. We denote by $\mathrm{sxc}(P)$ the *simple extension complexity*, i.e., the smallest size of any simple extension of the polytope P.

From a practical point of view, simplicity is an interesting property since it is the geometric interpretation of primal non-degeneracy of linear programs. In addition, large parts of combinatorial/extremal theory of polytopes deal with simple polytopes. Furthermore, as with other restrictions like symmetry, there indeed exist nice examples of simple extensions of certain polytopes relevant in optimization. For instance, generalizing the well-known fact that the permutahedron is a zonotope, Wolsey showed in the late 80's (personal communication) that, for ar-

bitrary processing times, the completion-time polytope for n jobs is a projection of an $O(n^2)$-dimensional cube. The main results of this chapter show, however, that for several polytopes relevant in optimization (among them both perfect matching polytopes and spanning tree polytopes) insisting on simplicity enforces very large sizes of the extensions. More precisely, we establish that for the following polytopes the simple extension complexity equals their number of vertices (note that the number of vertices of P is a trivial upper bound for $\mathrm{sxc}(P)$, realized by the extension obtained from writing P as the convex hull of its vertices):

- Perfect matching polytopes of complete graphs (Theorem 3.7.3)
- Perfect matching polytopes of complete bipartite graphs (Theorem 3.7.1)
- Uncapacitated flow polytopes of non-decomposable acyclic networks (Theorem 3.6.1)
- (Certain) random 0/1-polytopes (Theorem 3.3.8)
- Hypersimplices (Theorem 3.4.1)

Furthermore, we prove that

- the spanning tree polytope of the complete graph with n nodes has simple extension complexity at least $\Omega(2^{n-o(n)})$ (Theorem 3.5.2).

The chapter is structured as follows: We first focus on known construction techniques, and characterize when reflections and disjunctive programming yield simple extensions (Section 3.2). We continue with some techniques to bound the simple extension complexity of a polytope from below (Section 3.3). Then we deduce our results on hypersimplices (Section 3.4), spanning tree polytopes (Section 3.5), flow polytopes (Section 3.6) and perfect matching polytopes (Section 3.7). The core of the latter part is a strengthening of a result of Padberg and Rao's [52] on adjacencies in the perfect matching polytope (Theorem 3.7.6), which may be of independent interest.

Let us end this introduction by remarking that the concept of *simplicial extensions* is not interesting. To see this, observe that any d-polytope Q with N vertices has at least $d \cdot N$ facet-vertex incidences since every vertex lies in at least d facets. On the other hand, if Q is simplicial (i.e.,

all facets are simplices) and has f facets, the number of facet-vertex incidences is equal to $d \cdot f$, proving $f \geq N$. For every polytope P with N vertices, every extension polytope has at least N vertices, and hence a smallest possible simplicial extension polytope of P is the simplex with N vertices.

ANALYZING SIMPLE EXTENSIONS OF POLYTOPES:

3.2 Constructions

There are three major techniques for constructing extended formulations, namely dynamic programming, disjunctive programming and reflections. Extensions based on dynamic programs yield network flow polytopes for acyclic graphs, which are not simple in general and also have large simple extension complexities (see Section 3.6).

In this section we characterize for the other two techniques mentioned above in which cases the produced extensions are simple.

3.2.1 Reflections

Let $P = \{x \in \mathbb{R}^n \mid Ax \leq b\}$ be a polytope and let $H_{\leq} = \{x \in \mathbb{R}^n \mid \langle a, x\rangle \leq \beta\}$ be a halfspace in $\mathbb{R}^n$ (in particular we assume $a \neq \mathbb{0}$). Denoting by $P_1 := P \cap H_{\leq}$ the intersection of the polytope with the halfspace and by P_2 the image of P_1 under reflection at the boundary hyperplane $H_{=}$ of $H_{\leq}$, we call conv.hull$(P_1 \cup P_2)$ the *reflection* of P at $H_{\leq}$. The technique in [39] provides an extended formulation for this polytope.

Proposition 3.2.1 (Kaibel & Pashkovich [39])**.** *The polytope Q defined by*

$$Q = \{(x, y) \in \mathbb{R}^{n+n} \mid Ay \leq b, \langle a, y\rangle \leq \langle a, x\rangle \leq 2\beta - \langle a, y\rangle\,, \\ (x - y) \in \text{lin.hull}\,(a)\}$$

together with the projection onto the x-space is an extension of the polytope conv.hull $(P_1 \cup P_2)$.

Our contribution is the next theorem that clarifies under which circumstances Q is a simple polytope.

Theorem 3.2.2. *Let Q be the extension polytope for the reflection of P at $H_{\leq}$ as defined in this subsection, let $P_1 := P \cap H_{\leq}$, and let $F := P_1 \cap H_{=}$ be the intersection of P_1 with the reflection hyperplane. Then Q is simple if and only if P_1 is simple and either $P_1 = F$, or F is a facet of P_1 or $F = \emptyset$.*

Proof. We first observe that the faces

$$Q_1 := Q \cap \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^n \mid \langle a, y\rangle = \langle a, x\rangle\}$$
$$Q_2 := Q \cap \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^n \mid \langle a, x\rangle = 2\beta - \langle a, y\rangle\}$$

of Q are both affinely isomorphic to P_1. Thus Q can only be simple if P_1 is so. If $P_1 \subseteq H_{=}$ holds, $Q = Q_1 = Q_2$ and Q is simple if and only if P_1 is simple, proving the equivalence in case $P_1 = F$. Otherwise, let $d := \dim P_1$ and observe that $\dim Q = d + 1$ holds because Q_1 and Q_2 are proper faces of Q and Q's dimension cannot be larger than $d + 1$. Furthermore, $(x, y) \in Q_1 \cap Q_2$ holds if and only if $\langle a, x\rangle = \beta$ is satisfied, hence $Q_1 \cap Q_2$ is affinely isomorphic to F. Define $k := \dim F$.

We now assume that Q is simple and $F \neq \emptyset$, i.e., $k \geq 0$ holds. Let v be any vertex of $Q_1 \cap Q_2$. Since Q is simple and of dimension $d + 1$, v has $d + 1$ adjacent vertices, k of which lie in $Q_1 \cap Q_2$ (isomorphic to F). Furthermore, v has d neighbors in Q_i for $i = 1, 2$. Hence, k of these vertices lie in $Q_1 \cap Q_2$, $d - k$ lie in $Q_1 \setminus Q_2$ and $d - k$ lie in $Q_2 \setminus Q_1$. The resulting equation $k + (d - k) + (d - k) = d + 1$ yields $k = d - 1$, i.e., F is a facet of P_1. This proves necessity of the condition.

To prove sufficiency, from now on assume that P_1 is simple and $\dim Q = d + 1$ holds. We prove that every vertex (x, y) of Q not lying in $Q_1 \cap Q_2$ lies in at most (thus, exactly) $d + 1$ facets of Q. First, y can satisfy at most d inequalities of $Ay \leq b$ with equality because P_1 is simple. Second, (x, y) can satisfy at most one of the other two inequalities with equality since otherwise, $\langle a, x\rangle = \beta$ would hold, contradicting the fact that $(x, y) \notin Q_1 \cap Q_2$. Hence, the vertex lies in at most $d + 1$ facets, which proves the claim. This already proves that Q is simple in the case $F = \emptyset$, since then there are no further vertices.

It remains to show that if F is a facet of P_1 then every vertex (x, y) of $Q_1 \cap Q_2$ has at most $d + 1$ neighbors in Q. In this case, $Q_1 \cap Q_2$ is a facet of Q_1 and of Q_2, which in turn are facets of Q. Since $Q_1 \cap Q_2$ is a facet of

the simple polytope Q_i for $i = 1, 2$, the vertex (x, y) has $d - 1$ neighbors in the (simple) facet $Q_1 \cap Q_2$ and one neighbor in $Q_i \setminus (Q_1 \cap Q_2)$. In total, (x, y) has $d + 1$ neighbors, because all vertices of Q are vertices of Q_1 or Q_2 since for fixed y with $Ay \leq b$, any x with $(x - y) \in \text{lin.hull}(a)$ must satisfy one of the other two inequalities with equality if it is an extreme point. □

An interesting observation is that in case of a reflection at a hyperplane $H_=$ that does not intersect the given polytope P, the resulting extension polytope is combinatorially equivalent to $P \times [0, 1]$. This yields a (deformed) cube if such a reflection is applied iteratively if the initial polytope is a cube. One example is are the extension of size $2 \log m$ for a regular m-gon for the case of $m = 2^k$ with $k \in \mathbb{N}$.

Theorem 3.2.3. *Let $k \in \mathbb{N}$. The simple extension complexity of a regular 2^k-gon is at most $2k$.*

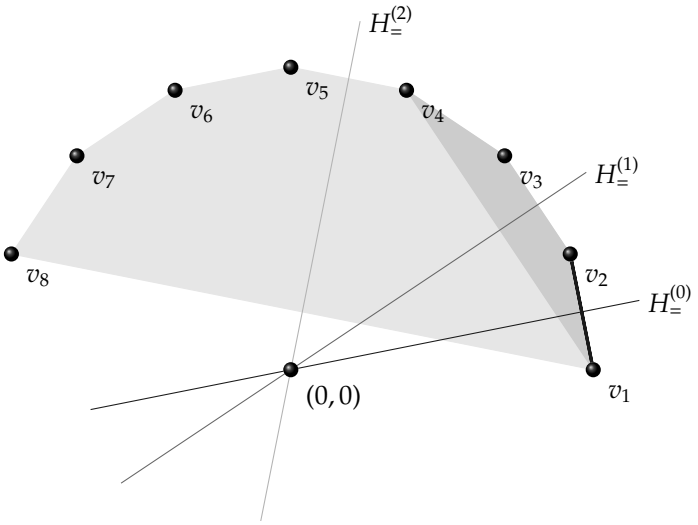


Figure 3.1: Some reflections used in the proof of Theorem 3.2.3 for a 16-gon.

Proof. We recursively define a series of polytopes as follows: The initial (simple) polytope is $P^{(0)} := \{(1,0)^\intercal\}$, i.e., a single point. Since the extensions we construct are located in increasingly higher-dimensional spaces we write coordinates as $(x,y,z)^\intercal \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^*$, where the dimension of the z-space increases, initially being zero.

We now define for $i = 0,1,2,\ldots,k-1$ the polytope $P^{(i+1)}$ as the reflection of $P^{(i)}$ at the halfspace

$$H^{(i)}_{\leq} := \{(x,y,z) \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^* \mid \\ -\sin((2^i-1)\cdot\pi/2^k)x + \cos((2^i-1)\cdot\pi/2^k)y \leq 0\}.$$

Theorem 3 in [39] shows that $P^{(k)}$ is an extension of a regular 2^k-gon. If we label the vertices of this 2^k-gon with $v_1, v_2, \ldots, v_{2^k}$ in counter-clockwise order starting with $v_1 = (1,0)^\intercal$, the proof even shows that the projection of $P^{(i)}$ onto the first two coordinates equals the convex hull of the vertices $v_1, v_2, \ldots, v_{2^i}$. Now for every $i = 0,1,\ldots,k-1$, the polytope $P^{(i)}$ does not intersect $H^{(i)}_{=}$ since the projection of such an intersection point would lie outside the mentioned convex hull.

This ensures that by induction all polytopes $P^{(i)}$ for $i = 0,1,2,\ldots,k$ are simple by Theorem 3.2.2 and that the last polytope $P^{(k)}$ is a simple extension of the regular 2^k-gon. □

3.2.2 Disjunctive Programming

The third major technique to construct extended formulations is by means of *disjunctive programming*, introduced by Balas [10],[11]. We only consider the special case of a disjunction of two polytopes $P_1, P_2 \subseteq \mathbb{R}^n$ and are interested in an extension of the convex hull of the union of the two.

A helpful tool is the *homogenization* $\text{homog}(P)$ of a polytope P, defined as $\text{homog}(P) := \text{conic.hull}(P \times \{1\})$. We say that a pointed polyhedral cone C is *weakly simple* if every extreme ray of C lies in exactly $\dim(C) - 1$ facets and *strongly simple* if C is a simple polyhedron. Clearly, a strongly simple cone is also weakly simple. Furthermore, if we have $C = \text{homog}(P)$ then C is weakly simple if and only if P is simple and C is strongly simple if and only if P is a simplex. We will need the following lemma about weak simplicity of cartesian products of cones.

Lemma 3.2.4. *Given two pointed polyhedral cones $C_1 \subseteq \mathbb{R}^{n_1}$, $C_2 \subseteq \mathbb{R}^{n_2}$, their product cone $C := C_1 \times C_2 \subseteq \mathbb{R}^{n_1+n_2}$ is weakly simple if and only if both C_1 and C_2 are strongly simple.*

Proof. It is easy to check that $C_1 \times C_2 = \{(x_1, x_2) \in \mathbb{R}^{n_1+n_2} \mid x_i \in C_i \;\; i = 1, 2\}$ is a pointed polyhedral cone again. Furthermore, the faces of $C_1 \times C_2$ are exactly the products of faces of C_1 and C_2, their dimensions add up, and a face $F_1 \times F_2$ of $C_1 \times C_2$ is contained in another face $G_1 \times G_2$ if and only if $F_1 \subseteq G_1$ and $F_2 \subseteq G_2$ hold.

Hence, the extreme rays of $C_1 \times C_2$ are either products of extreme rays of C_1 with $\mathbb{O}_{n_2}$ or products of $\mathbb{O}_{n_1}$ with extreme rays of C_2. Similarly, the facets of $C_1 \times C_2$ are either products of facets of C_1 with C_2 or products of C_1 with facets of C_2.

We consider an extreme ray of C, w.l.o.g. of the form $r \times \mathbb{O}_{n_2}$, where r is an extreme ray of C_1. It is clearly contained in the facets $F_1 \times C_2$, where F_1 is a facet of C_1 containing r. For every facet F_2 of C_2, we have $\mathbb{O}_{n_2} \subseteq F_2$ and hence $C_1 \times F_2$ contains $r \times \mathbb{O}_{n_2}$.

Thus, if r is contained in k facets of C_1 and C_2 has ℓ facets then $r \times \mathbb{O}_{n_2}$ is contained in $k + \ell$ facets of $C_1 \times C_2$.

We always have $k \geq \dim C_1 - 1$ and $\ell \geq \dim C_2$ since C_1 and C_2 are pointed polyhedral cones. Hence, $k + \ell \geq \dim C_1 + \dim C_2 - 1$ holds and we have equality if and only if $k = \dim C_1 - 1$ and $\ell = \dim C_2$ are satisfied, and hence C_1 and C_2 are strongly simple. □

We now turn to the mentioned extension of $P = \text{conv.hull}\,(P_1 \cup P_2)$ using disjunctive programming. Define

$$Q = \left\{(x^1, \lambda_1, x^2, \lambda_2) \in \text{homog}\,(P_1) \times \text{homog}\,(P_2) \mid \lambda_1 + \lambda_2 = 1\right\}.$$

It is well-known that Q together with the projection $(x^1, \lambda_1, x^2, \lambda_2) \mapsto x^1 + x^2$ yields an extension of P. We now characterize when Q is a simple polytope.

Theorem 3.2.5. *The extension polytope Q of the disjunctive program for the polytope $P = \text{conv.hull}\,(P_1 \cup P_2)$ is simple if and only if P_1 and P_2 are simplices.*

Proof. As Q is the intersection of the pointed cone $C = \text{homog}(P_1) \times \text{homog}(P_2)$ with the hyperplane defined by $\lambda_1 + \lambda_2 = 1$ (which does not contain any of C's extreme rays), we know that Q is simple if and only if C is weakly simple. Now Lemma 3.2.4 yields the result. □

ANALYZING SIMPLE EXTENSIONS OF POLYTOPES:

3.3 Bounding Techniques

Let $P \subseteq \mathbb{R}^n$ be a polytope with N vertices.

Clearly, P is the set of all convex combinations of its vertices, immediately providing an extended formulation of size N:

$$P = \text{proj}_x\{(x, y) \in \mathbb{R}^n \times \mathbb{R}_+^{\text{vert}(P)} \mid x = \sum_{v \in \text{vert}(P)} y_v v, \sum_{v \in \text{vert}(P)} y_v = 1\}$$

Note that this *trivial extension* is simple since the extension is an $(N-1)$-dimensional simplex.

An easy observation for extensions $P = \pi(Q)$ with $Q \subseteq \mathbb{R}^d$ and $\pi : \mathbb{R}^d \to \mathbb{R}^n$ is that the assignment $F \mapsto \pi^{-1}(F) \cap Q = \{y \in Q \mid \pi(y) \in F\}$ defines a map j which embeds $\mathcal{L}(P)$ into $\mathcal{L}(Q)$, i.e., it is one-to-one and preserves inclusion in both directions (see [26]). Note that this embedding furthermore satisfies $j(F \cap F') = j(F) \cap j(F')$ for all faces F, F' of P (where the nontrivial inclusion $j(F) \cap j(F') \subseteq j(F \cap F')$ follows from $\pi(j(F) \cap j(F')) \subseteq \pi(j(F)) \cap \pi(j(F')) = F \cap F'$). We use the shorthand notation $j(v) := j(\{v\})$ for vertices v of P.

We consider the *face-vertex non-incidence graph* $G_{\mathcal{N}}(P)$, a bipartite graph having the faces and the vertices of P as the node set and edges $\{F, v\}$ for all $v \notin F$ (where v is a vertex and F is a face of P). Every facet $\hat{f}$ of an extension induces two node sets of this graph in the following

way:

$$\begin{aligned} \mathcal{F}(\hat{f}) &:= \left\{F \text{ face of } P \mid j(F) \subseteq \hat{f}\right\} \\ \mathcal{V}(\hat{f}) &:= \left\{v \text{ vertex of } P \mid j(v) \not\subseteq \hat{f}\right\} \end{aligned} \tag{3.1}$$

We call $\mathcal{F}(\hat{f})$ and $\mathcal{V}(\hat{f})$ the *set of faces* (resp. *vertices*) *induced by the facet* $\hat{f}$ (with respect to the extension $P = \pi(Q)$). Typically, the extension and the facet $\hat{f}$ are fixed and we just write $\mathcal{F}$ (resp. $\mathcal{V}$). It may happen that $\mathcal{V}(\hat{f})$ is equal to the whole vertex set, e.g., if $\hat{f}$ projects into the relative interior of P. If $\mathcal{V}(\hat{f})$ is a proper subset of the vertex set we call facet $\hat{f}$ *proper*.

For each facet $\hat{f}$ of an extension of P the face and vertex sets $\mathcal{F}(\hat{f})$, $\mathcal{V}(\hat{f})$ together induce a biclique (i.e., complete bipartite subgraph) in $G_{\mathcal{N}}(P)$. It follows from Yannakakis [69] that every edge in $G_{\mathcal{N}}(P)$ is covered by at least one of those induced bicliques. We provide a brief combinatorial argument for this (in particular showing that we can restrict to proper facets) in the proof of the following proposition.

Proposition 3.3.1. *Let P be a polytope and Q be an extension with $P = \pi(Q)$. Then the subgraph of $G_{\mathcal{N}}(P)$ induced by $\mathcal{F}(\hat{f}) \cup \mathcal{V}(\hat{f})$ is a biclique for every facet $\hat{f}$ of Q. Furthermore, every edge $\{F, v\}$ of $G_{\mathcal{N}}(P)$ is covered by at least one of the bicliques induced by a proper facet.*

Proof. Let $\hat{f}$ be one of the facets and assume that an edge $\{F, v\}$ with $F \in \mathcal{F}(\hat{f})$ and $v \in \mathcal{V}(\hat{f})$ is not present in $G_{\mathcal{N}}(P)$, i.e., $v \in F$. From $v \in F$ we obtain $j(v) \subseteq j(F) \subseteq \hat{f}$, a contradiction to $v \in \mathcal{V}(\hat{f})$.

To prove the second statement, let $\{F, v\}$ be any edge of $G_{\mathcal{N}}(P)$, i.e., $v \notin F$. Observe that the preimages $G := j(F)$ and $g := j(v)$ are also not incident (i.e., $g \not\subseteq G$) since j is a lattice embedding. As G is the intersection of all facets of Q it is contained in (the face-lattice of a polytope is coatomic), there must be at least one facet $\hat{f}$ containing G but not g (since otherwise g would be contained in G), yielding $F \in \mathcal{F}(\hat{f})$ and $v \in \mathcal{V}(\hat{f})$.

If $F \neq \emptyset$, any vertex $w \in F$ satisfies $j(w) \subseteq G \subseteq \hat{f}$ and hence $\hat{f}$ is a proper facet. If $F = \emptyset$, let w be any vertex of P distinct from v. The preimages $j(v)$ and $j(w)$ clearly satisfy $j(v) \not\subseteq j(w)$. Again, since the face-lattice of Q is coatomic, there exists a facet $\hat{f}$ with $j(w) \subseteq \hat{f}$ but $j(v) \not\subseteq \hat{f}$.

Hence, $\hat{f}$ is a proper facet, and (since $\emptyset \neq F \subseteq \hat{f}$ holds) $F \in \mathcal{F}(\hat{f})$ and $v \in \mathcal{V}(\hat{f})$ holds. □

Before moving on to simple extensions we mention two useful properties of the induced sets. Both can be easily verified by examining the definitions of $\mathcal{F}$ and $\mathcal{V}$. See Figure 3.2 for an illustration.

Lemma 3.3.2. *Let $\mathcal{F}$ and $\mathcal{V}$ be the face and vertex sets induced by a facet of an extension of P, respectively. Then $\mathcal{F}$ is closed under taking subfaces and $\mathcal{V} = \{v \text{ vertex of } P \mid v \notin \bigcup_{F \in \mathcal{F}} F\}$ holds.*

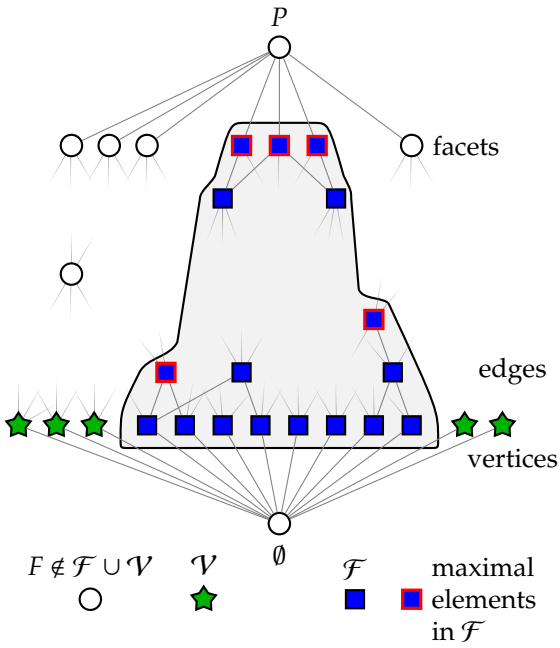


Figure 3.2: The sets $\mathcal{F}$ and $\mathcal{V}$ in the face lattice.

For the remainder of this section we assume that the extension polytope Q is a *simple* polytope and that $\mathcal{F}$ and $\mathcal{V}$ are face and vertex sets induced by a facet of Q.

Theorem 3.3.3. *Let $\mathcal{F}$ and $\mathcal{V}$ be the face and vertex sets induced by a facet of a* simple *extension of P, respectively. Then*

(a) all pairs (F, F') of faces of P with $F \cap F' \neq \emptyset$ and $F, F' \notin \mathcal{F}$ satisfy $F \cap F' \notin \mathcal{F}$,

(b) the (inclusion-wise) maximal elements in $\mathcal{F}$ are facets of P,

(c) and every vertex $v \notin \mathcal{V}$ is contained in some facet F of P with $F \in \mathcal{F}$.

Proof. Let $\hat{f}$ be the facet of Q inducing $\mathcal{F}$ and $\mathcal{V}$ and F, F' two faces of P with non-empty intersection. Since $F \cap F' \neq \emptyset$, we have $j(F \cap F') \neq \emptyset$, thus the interval in $\mathcal{L}(Q)$ between $j(F \cap F')$ and Q is a Boolean lattice, i.e., isomorphic to the face-lattice of a simplex, (because Q is simple, see Proposition 2.16 in [70]). Suppose $F \cap F' \in \mathcal{F}(\hat{f})$. Then $\hat{f}$ is contained in that interval and it is a coatom, hence it contains at least one of $j(F)$ and $j(F')$ due to $j(F) \cap j(F') = j(F \cap F')$. But this implies $j(F) \in \mathcal{F}$ or $j(F') \in \mathcal{F}$, proving (a).

For (b), let F be an inclusion-wise maximal face in $\mathcal{F}$ but not a facet of P. Then F is the intersection of two faces F_1 and F_2 of P properly containing F. Due to the maximality of F, $F_1, F_2 \notin \mathcal{F}$ but $F_1 \cap F_2 \in \mathcal{F}$, contradicting (a).

Statement (c) follows directly from (b) and Lemma 3.3.2. □

In order to use the Theorem 3.3.3 for deriving lower bounds on the sizes of simple extensions of a polytope P, one needs to have good knowledge of parts of the face lattice of P. The part one usually knows most about is formed by the vertices, and edges of P. Therefore, we specialize Theorem 3.3.3 to these faces for later use.

Let $G = (V, E)$ be a graph. Define for node subsets $W \subseteq V$ the *common neighbor operator* $\Lambda(\cdot)$ by

$$\Lambda(W) := W \cup \{v \in V \mid v \text{ has at least two neighbors in } W\}. \tag{3.2}$$

A set $W \subseteq V$ is then a (proper) *common neighbor closed* (for short Λ-*closed*) set if $\Lambda(W) = W$ (and $W \neq V$) holds. We call sets W with a minimum node distance of at least 3 (i.e., the distance-2-neighborhood of a node $w \in W$ does not contain another node $w' \in W$) *isolated*. Isolated node sets are clearly Λ-closed. Note that singleton sets are isolated and hence proper Λ-closed. In particular, the vertex sets induced by the facets of the trivial extension (see beginning of Section 3.3) are the singleton sets.

Using this notion, we obtain the following corollary of Theorem 3.3.3.

Corollary 3.3.4. *The vertex set $\mathcal{V}$ induced by a proper facet of a* simple *extension of P is a proper Λ-closed set.*

Proof. Theorem 3.3.3 implies that for every $\{u,v\}, \{v,w\}$ of (distinct) adjacent edges of P, we have

$$\{u,v\}, \{v,w\} \notin \mathcal{F} \;\Rightarrow\; \{v\} \notin \mathcal{F}.$$

Due to Lemma 3.3.2, $\mathcal{V} = \{v \text{ vertex of } P \mid v \notin \bigcup \mathcal{F}\}$, where $\mathcal{F}$ is the face set induced by the same facet. Hence, $v \notin \mathcal{V}$ implies $\{u,v\} \in \mathcal{F}$ or $\{v,w\} \in \mathcal{F}$, thus $u \notin \mathcal{V}$ or $w \notin \mathcal{V}$ and we conclude that $\mathcal{V}$ is Λ-closed.

Furthermore, $\mathcal{V}$ is not equal to the whole vertex set of P since the given facet is proper. □

We just proved that every biclique $\mathcal{F} \cup \mathcal{V}$ induced by a (proper) facet from a simple extension must satisfy certain properties. The next example shows that these properties are not sufficient for an extension polytope to be simple.

Example 3.3.5. *Define $m_1, \ldots, m_7 \in \mathbb{R}^3$ to be the columns of the matrix*

$$M := \begin{pmatrix} 1 & 5 & 1 & 0 & -1 & -5 & -1 \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 0 & -4 & 0 & 1 & 0 & -4 & 0 \end{pmatrix},$$

and let $Q := \operatorname{conv.hull}\{m_1, \ldots, m_7\} \subseteq \mathbb{R}^3$ be their convex hull. The vertex m_4 has 4 neighbors, that is, Q is not simple. Let P be the projection of Q onto the first two coordinates. Observe that P is a 6-gon and that the only relevant types of faces F, F' are adjacent edges of P. It is quickly verified that all induced face and vertex sets satisfy Theorem 3.3.3 and Corollary 3.3.4, respectively (see Figure 3.3).

Note that this example only shows that we cannot decide from the biclique covering whether the extension is simple. It may still be true that for such biclique coverings there always *exists* a simple extension.

The polytope Q from the example can be used to show that Corollary 3.3.4 is indeed a specialization of Theorem 3.3.3 (a). To see this, consider the set $\mathcal{F}$ of faces consisting of the triangles $\operatorname{conv.hull}\{m_1, m_4, m_5\}$, $\operatorname{conv.hull}\{m_3, m_4, m_7\}$ and all their subfaces. Lemma 3.3.2 implies that

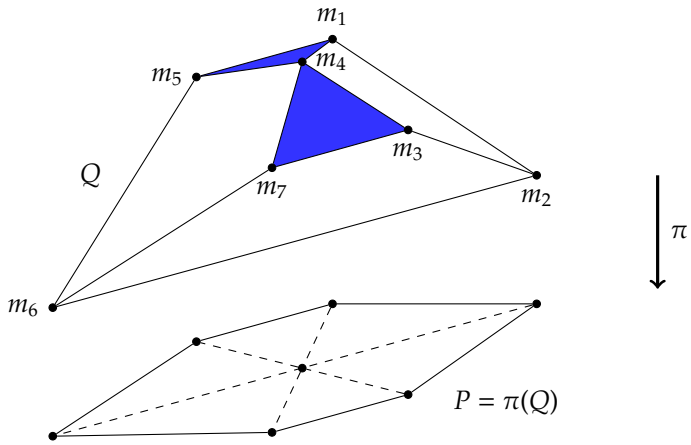


Figure 3.3: Polytope Q from Example 3.3.5 and its projection P.

$\mathcal{V} = \{m_2, m_6\}$ holds, which is proper Λ-closed. But $\mathcal{F}$ does not satisfy Theorem 3.3.3 (a) for the choice $F := \text{conv.hull}\{m_1, m_2, m_3, m_4\} \notin \mathcal{F}$, $F' := \text{conv.hull}\{m_4, m_5, m_6, m_7\} \notin \mathcal{F}$ since $F \cap F' = \{m_4\} \in \mathcal{F}$.

Nevertheless we can obtain useful lower bounds from Theorem 3.3.3 and Corollary 3.3.4.

Corollary 3.3.6. *The vertex set of a polytope P can be covered by* $\text{sxc}(P)$ *many proper Λ-closed sets.*

Lemma 3.3.7. *Let P be a polytope and G its graph. If all proper Λ-closed sets in G are isolated then the simple extension complexity of P is greater than the maximum size of the neighborhood of any node of G.*

Proof. Let w be a node maximizing the size of the neighborhood and let W be the neighborhood of w. Since no isolated set can contain more than one node from $W \cup \{w\}$, Corollary 3.3.6 implies the claim. □

Using knowledge about random 0/1-polytopes, we can easily establish the following result.

Theorem 3.3.8. *There is a constant $\sigma > 0$ such that a random d-dimensional 0/1-polytope P with at most $2^{\sigma d}$ vertices asymptotically almost surely has a simple extension complexity equal to its number of vertices.*

Proof. One of the main results of Gillmann's thesis (See Theorem 3.37 in [30] for $k = 2$) is that there is such a σ ensuring that a random d-dimensional 0/1-polytope P with at most $2^{\sigma d}$ vertices asymptotically almost surely has every pair of vertices adjacent. Since in this situation the only proper Λ-closed sets are the singletons, Corollary 3.3.6 yields the claim. □

ANALYZING SIMPLE EXTENSIONS OF POLYTOPES:

3.4 Hypersimplices

Let Δ_k^n denote the k-hypersimplex in $\mathbb{R}^n$ for some $k \in \{0, 1, 2, \ldots, n\}$, i.e., the 0/1-cube intersected with the hyperplane $\langle \mathbb{1}_n, x \rangle = k$. Note that its vertices are all 0/1-vectors with exactly k 1's, since the above linear system is totally unimodular (a row of ones together with two unit matrices). It follows from the knowledge about edges and 2-faces of the cube that two vertices of Δ_k^n are adjacent if and only if they differ in exactly two coordinates. In other words, all neighbors of a vertex x can be obtained by replacing a 1 by a 0 at some index and a 0 by a 1 at some other index. Observe that Δ_k^n is almost simple for $2 \leq k \leq n-2$ in the sense that its dimension is $n-1$, but every vertex lies in exactly n facets. With this in mind, the following result may seem somewhat surprising.

Theorem 3.4.1. *Let $1 \leq k \leq n-1$. The simple extension complexity of $\Delta_k^n \subseteq \mathbb{R}^n$ is equal to its number of vertices $\binom{n}{k}$.*

Proof. The case of $k = 1$ or $k = n-1$ is clear since then Δ_k^n is an $(n-1)$-dimensional simplex.

Let $2 \leq k \leq n-2$ and $\mathcal{F}$ and $\mathcal{V}$ be face and vertex sets induced by a proper facet of a simple extension of Δ_k^n.

Since every vertex v of Δ_k^n has $v_i = 0$ or $v_i = 1$, at most one of the facets $x_i \geq 0$ or $x_i \leq 1$ can be in $\mathcal{F}$ for every $i \in [n]$ (otherwise $\mathcal{V}$ would be empty). We can partition $[n]$ into $L \cup U \cup R$ such that L (resp. U) contains those indices $i \in [n]$ such that the facet corresponding to $x_i \geq 0$ (resp. $x_i \leq 1$) is in $\mathcal{F}$ and R contains the remaining indices. Lemma 3.3.2

yields

$$\mathcal{V} = \left\{ v \text{ vertex of } \Delta_k^n \mid v_L = \mathbb{1},\ v_U = \mathbb{O} \right\}. \tag{3.3}$$

We now prove that a node set $\mathcal{V}$ of this form is proper Λ-closed only if $|\mathcal{V}| = 1$. Then, Corollary 3.3.6 yields the claim.

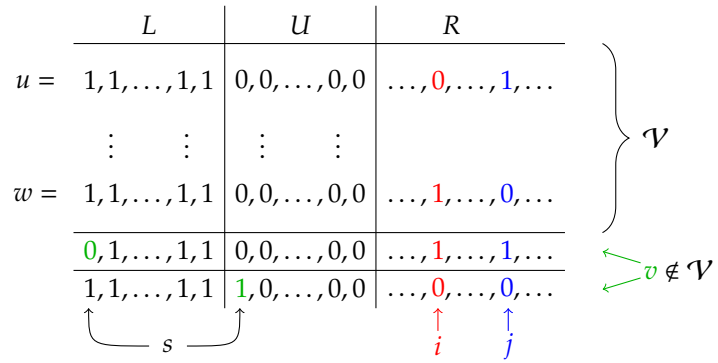


Figure 3.4: Vertices of Δ_k^n in $\mathcal{V}$ for a biclique.

Indeed, if we have $|\mathcal{V}| > 1$, then there exist vertices $u, w \in \mathcal{V}$ and indices $i, j \in R$ such that $u_i = w_j = 1$, $u_j = w_i = 0$ and $u_\ell = w_\ell$ for all $\ell \notin \{i, j\}$ (see Figure 3.4). Choose any $s \in L \cup U$ and observe that, since $u, w \in \mathcal{V}$, $u_s = w_s = 1$ if $s \in L$ and $u_s = w_s = 0$ if $s \in U$. The following vertex is easily checked to be adjacent to u and w (min and max must be read component-wise):

$$v := \begin{cases} \max(u, w) - \mathbb{e}^{(s)} & \text{if } s \in L \\ \min(u, w) + \mathbb{e}^{(s)} & \text{if } s \in U \end{cases}$$

As $v_s = 0$ if $s \in L$ and $v_s = 1$ if $s \in U$, $v \notin \mathcal{V}$. This contradicts the fact that $\mathcal{V}$ is Λ-closed. □

ANALYZING SIMPLE EXTENSIONS OF POLYTOPES:

3.5 Spanning Tree Polytopes

In this section we bound the simple extension complexity of the spanning tree polytope $P_{\text{spt}}(K_n)$ of the complete graph K_n with n nodes. In order to highlight different perspectives we mention three equivalent adjacency characterizations that all follow from the fact that the spanning tree polytope is the base polytope of a graphic matroid (see [63], Theorem 40.6.). The vertices corresponding to spanning trees T and T' are adjacent in the spanning tree polytope if and only if ...

- ... $|T \Delta T'| = 2$ holds.
- ... T' arises from T by removing one edge and reconnecting the two connected components by another edge.
- ... T' arises from T by adding one additional edge and removing any edge from the cycle that this edge created.

From the third statement it is easy to see that the maximum degree of the 1-skeleton of $P_{\text{spt}}(K_n)$ is in $O(n^3)$, since there are $O(n^2)$ possible choices for the additional edge, each of which yields $O(n)$ choices for a cycle-edge to remove.

Lemma 3.5.1. *All proper Λ-closed sets in the graph of $P_{spt}(K_n)$ are isolated.*

Proof. Throughout the proof, we will identify vertices with the corresponding spanning trees.

Suppose $\mathcal{V}$ is a proper Λ-closed set that is not isolated. Then there are spanning trees $T_1, T_2 \in \mathcal{V}$ and $T_3 \notin \mathcal{V}$, such that T_1 is adjacent to both T_2 and T_3, but T_2 and T_3 are not adjacent.

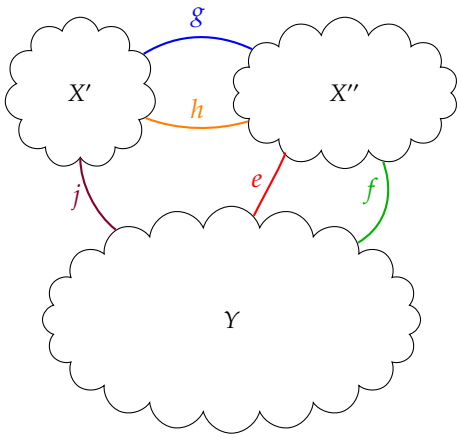


Figure 3.5: Case 2 of Lemma 3.5.1.

Let e be the unique edge that is in T_1 but not in T_2, i.e., $\{e\} = T_1 \setminus T_2$. Analogously, let $\{f\} = T_2 \setminus T_1$, $\{g\} = T_1 \setminus T_3$ and $\{h\} = T_3 \setminus T_1$. Since T_2 and T_3 are not adjacent in the polytope, their symmetric difference $T_2 \Delta T_3 \subseteq \{e, f, g, h\}$ must have cardinality greater than 2. Because the symmetric difference of two spanning trees consists of an even number of edges, this cardinality must be equal to 4, proving $e \neq g$. Let us define $F := T_1 \setminus \{e, g\}$, which is a tree with two edges missing, i.e., a forest with three connected components X', X'', Y. W.l.o.g., g connects X' with X'' and e connects X'' with Y. Then T_2 and T_3 can be written as $F \cup \{f, g\}$ and $F \cup \{e, h\}$, respectively. There are two possible cases for h:

Case 1: h connects Y with X' or X''.
Let $T' := F \cup \{g, h\}$ and observe that T' is a spanning tree since g connects X' with X'' and h connects one of both with Y. Obviously, T' is adjacent to T_1, T_2 and T_3. Since T' is adjacent to T_1 and T_2, $T' \in \Lambda(\mathcal{V}) = \mathcal{V}$. Since T_3 is adjacent to $T_1, T' \in \mathcal{V}$, this in turn implies the contradiction $T_3 \in \mathcal{V}$.

Case 2: h connects X' with X'' (see Figure 3.5).
Let j be any edge connecting X' with Y (recall that we dealing with a complete graph) and let $T' := F \cup \{g, j\}$, which is a spanning tree adjacent to T_1 and T_2 and hence $T' \in \Lambda(\mathcal{V}) = \mathcal{V}$. Clearly, $T'' := F \cup \{e, j\}$ is a spanning tree adjacent to T_1 and T' and hence $T'' \in \mathcal{V}$. Finally, let $T''' := F \cup \{h, j\}$ be a third spanning tree adjacent to T' and T''. Again, we have $T''' \in \mathcal{V}$ due to $\Lambda(\mathcal{V}) = \mathcal{V}$.

Since T_3 is adjacent to T_1 and T''', exploiting $\Lambda(\mathcal{V}) = \mathcal{V}$ once more yields the contradiction $T_3 \in \mathcal{V}$. □

Using this result we immediately get a lower bound of $\Omega(n^3)$ for the simple extension complexity of $P_{\text{spt}}(K_n)$ since the maximum degree of its graph is of that order. However, we can prove a much stronger result.

Theorem 3.5.2. *The simple extension complexity of the spanning tree polytope of K_n is in $\Omega(2^{n-o(n)})$.*

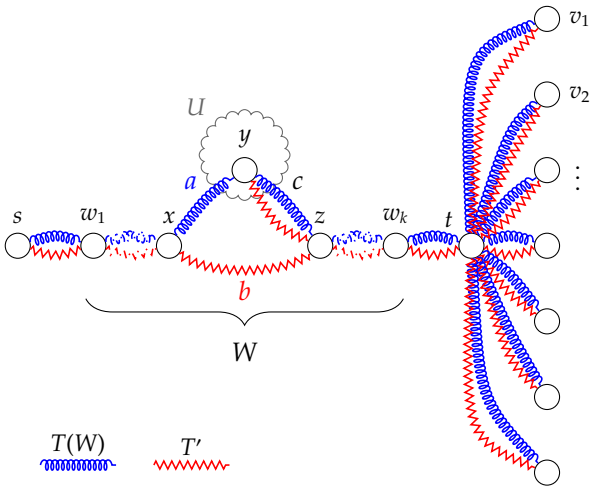


Figure 3.6: Construction for Theorem 3.5.2.

Proof. Assume $n \geq 5$ and let s, t be any two distinct nodes of K_n. Consider the following set of subsets of the nodes $V \setminus \{s, t\}$

$$\mathcal{W} := \{W \subseteq V \setminus \{s, t\} \mid |W| = \lfloor n/2 \rfloor\}.$$

Let $k := \lfloor n/2 \rfloor$, fix some ordering of the nodes $w_1, w_2, \ldots, w_k \in W$ for each $W \in \mathcal{W}$ and define a specific tree $T(W)$

$$\begin{aligned} T(W) \quad := \quad & \{\{s, w_1\}, \{w_k, t\}\} \\ & \cup \{\{w_i, w_{i+1}\} \mid i \in [k-1]\} \\ & \cup \{\{t, v\} \mid v \notin (W \cup \{s, t\})\} \end{aligned}$$

as depicted in Figure 3.6. We will now prove that for each simple extension of $P_{\text{spt}}(K_n)$ every such $T(W)$ must be in a different induced vertex set (w.r.t. the facets of a simple extension).

Let $W \in \mathcal{W}$ be some set W with tree $T(W)$. Let $\mathcal{F}$ and $\mathcal{V}$ be the face and vertex sets, respectively, induced by a proper facet of a simple extension such that $T(W)$ is in $\mathcal{V}$. Construct an adjacent tree T' as follows.

Choose some vertex $y \in W$ and let x-y-z be a subpath of the s-t-path in $T(W)$ in that order. Note that $\{x, y, z\} \subseteq W \cup \{s, t\}$. Define $a := \{x, y\}$, $b := \{x, z\}$ and $c := \{y, z\}$.

Let $T' = T(W) \setminus \{a\} \cup \{b\}$. Because T' is adjacent to $T(W)$, Lemma 3.5.1 implies $T' \notin \mathcal{V}$. Hence, due to Lemma 3.3.2, there must be a facet $F \in \mathcal{F}$ defined by $x(E[U]) \leq |U| - 1$ (with $|U| \geq 2$) which contains T'. Furthermore, this facet does not contain $T(W)$ because $T(W) \in \mathcal{V}$ holds. Hence, we have $|T(W)[U]| < |U| - 1$ and $|T'[U]| = |U| - 1$. This implies $|T(W) \cap \delta(U)| \geq 2$ and $|T' \cap \delta(U)| = 1$. Obviously, $a \in \delta(U)$ and $b \notin \delta(U)$.

Then $x, z \in U$ if and only if $y \notin U$ because $a \in \delta(U)$ and $b \notin \delta(U)$. Hence, $c \in \delta(U)$, i.e., $T \cap \delta(U) = \{c\}$. Due to $|U| \geq 2$, this implies $U = V \setminus \{y\}$.

As this can be argued for any $y \in W$, we have that the facets defined by $V \setminus \{y\}$ are in $\mathcal{F}$ for all $y \in W$. Hence, $\mathcal{V}$ contains only trees T for which $|T \cap \delta(V \setminus \{y\})| = |T \cap \delta(\{y\})| \geq 2$, i.e., no leaf of T is in W.

This shows that for distinct sets $W, W' \in \mathcal{W}$, any vertex set $\mathcal{V}$ induced by a proper facet of a simple extension that contains $T(W)$ does not contain $T(W')$ because any vertex $v \in W \setminus W'$ is a leaf of $T(W')$. Hence,

the number of simple bicliques is at least

$$|\mathcal{W}| = \binom{n-2}{\lfloor n/2 \rfloor} \in \Omega(2^{n-o(n)}).$$

□

ANALYZING SIMPLE EXTENSIONS OF POLYTOPES:

3.6 Flow Polytopes for Acyclic Networks

Many extended formulations model the solutions to the original formulation via a path in a specifically constructed directed acyclic graph. A simple example is the linear-size formulation for the parity polytope by Carr and Konjevod [16], and a more elaborate one is the approximate formulation for 0/1-knapsack polytopes by Bienstock [13].

Let $D = (V, A)$ be a directed acyclic graph with fixed source $s \in V$ and sink $t \in V$ ($t \neq s$). By $\mathcal{P}_{s,t}(D)$ we denote the arc-sets of s-t-paths in D. For some path $P \in \mathcal{P}_{s,t}(D)$ and nodes $u, v \in V(P)$, we denote by $P|_{(u,v)}$ the subpath of P going from u to v if it exists.

For acyclic graphs, the convex hull of the characteristic vectors of all s-t-paths is equal to the uncapacitated s-t-flow polytope $P_{\text{s-t-flow}}(D)$ with flow-value 1, since the linear description of the latter is totally unimodular. The inequalities in this description correspond to nonnegativity constraints of the arc variables, and a vertex corresponding to the path P is obviously non-incident to a facet corresponding to $y_a \geq 0$ if and only if $a \in P$ holds. Adjacency in the path polytope was characterized by Gallo and Sodini [28], and can be stated as follows: Two s-t-paths P, P' correspond to adjacent vertices of the polytope if and only if their symmetric difference consists of two paths from x to y ($x, y \in V$, $x \neq y$) without common inner nodes. In other words, they must split and merge exactly once.

Such a network formulation can be easily decomposed into two

independent formulations if a node v exists such that every s-t-path traverses v. We are now interested in the simple extension complexities of flow polytopes of s-t-networks that cannot be decomposed in such a trivial way. Our main result in this section is the following:

Theorem 3.6.1. *Let $D = (V, A)$ be a directed acyclic graph with source $s \in V$ and sink $t \in V$ such that for every node $v \in V \setminus \{s, t\}$ there exists an s-t-path in D that does not traverse v. Then the simple extension complexity of $P_{s\text{-}t\text{-}flow}(D) \subseteq \mathbb{R}^A_+$ is equal to the number of distinct s-t-paths $|\mathcal{P}_{s,t}(D)|$.*

Proof. Let $\mathcal{F}$ and $\mathcal{V}$ be the face and vertex sets induced by a proper facet of a simple extension of $P_{\text{s-t-flow}}(D)$, respectively. The goal is to prove $|\mathcal{V}| = 1$. Let us assume for the sake of contradiction $|\mathcal{V}| \geq 2$. By Theorem 3.3.3 (b), the (inclusion-wise) maximal faces in $\mathcal{F}$ are facets. Let $\emptyset \neq B' \subseteq A$ be the arc set corresponding to these facets. By Lemma 3.3.2, $\mathcal{V}$ is the set of (characteristic vectors of) paths $P \in \mathcal{P}_{s,t}(D)$ satisfying $P \supseteq B'$. Let $B \subseteq A$ be the set of arcs common to all such paths and note that $B \supseteq B' \neq \emptyset$ by definition of B'.

By construction, for any path $P \in \mathcal{V}$ and any arc $a \in P \setminus B$, there is an alternative path $P' \in \mathcal{V}$ with $a \notin P'$.

Let us fix one of the paths $P \in \mathcal{V}$. Let, without loss of generality, $(x', x) \in B$ be such that the arc of P leaving x (exists and) is not in B. If such an arc does not exist, since $B \neq P$, there must be an arc $(x, x') \in B$ such that the arc of P entering x is not in B. In this case, revert the directions of all arcs in D and exchange the roles of s and t, and apply the following arguments to the new network. Let y be the first node on $P|_{(x,t)}$ different from x and incident to some arc in B or, if no such y exists, let $y := t$. Paths in $\mathcal{V}$ must leave x and enter y but may differ in-between. The set of traversed nodes is defined as

$$S := \{v \in V \setminus \{x, y\} \mid \exists\, x\text{-}v\text{-}y\text{-path in } D\}.$$

By construction, $x \notin \{s, t\}$ and by the assumptions of the Theorem there exists a path $P' \in \mathcal{P}_{s,t}(D)$ that does not traverse x. Let s' be the last node on $P|_{(s,x)}$ that is traversed by P'. Analogously, let t' be the first node of $V(P|_{(x,t)}) \cup S$ that is traversed by P'. Note that $t' \neq x$ since t' is traversed by P' but x is not. We now distinguish two cases for which we show that $\mathcal{V}$ is not Λ-closed yielding a contradiction to Corollary 3.3.4:

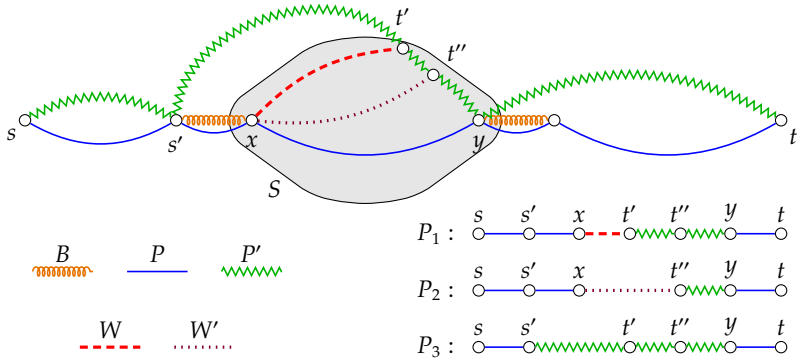


Figure 3.7: Construction for Case 1 in the proof of Theorem 3.6.1.

Case 1: $t' \in S$.
By definition of S there must be an x-t'-y-path W. Note that t' could be equal to y and then W could agree with $P|_{(x,y)}$ as well. Let $(z,t') \in W$ be the arc of W entering t'. By definition of y, we conclude that $(z,t') \notin B$. Hence, there is an alternative x-y-path $W' \neq W$ that does not use (z,t'). We choose W' such that it uses as many arcs of $W|_{(t',y)}$ as possible. Construct the following three paths (see Figure 3.7):

$$\begin{aligned} P_1 &:= P|_{(s,x)} \cup W \cup P|_{(y,t)} \\ P_2 &:= P|_{(s,x)} \cup W' \cup P|_{(y,t)} \\ P_3 &:= P|_{(s,s')} \cup P'|_{(s',t')} \cup W|_{(t',y)} \cup P|_{(y,t)} \end{aligned}$$

By construction $P_1, P_2 \in \mathcal{V}$ but $P_3 \notin \mathcal{V}$. P_1 and P_3 are adjacent in $P_{\text{s-t-flow}}(D)$ since they only differ in the disjoint paths from s' to t'. Analogously, P_2 and P_3 are adjacent and thus, contradicting the fact that $\mathcal{V}$ is Λ-closed.

Case 2: $t' \notin S$.
Let $W := P|_{(x,y)}$ and let W' be a different x-y-path that must exist by definition of y. Construct the following three paths (see Figure 3.8):

$$\begin{aligned} P_1 &:= P = P|_{(s,x)} \cup W \cup P|_{(y,t)} \\ P_2 &:= P|_{(s,x)} \cup W' \cup P|_{(y,t)} \\ P_3 &:= P|_{(s,s')} \cup P'|_{(s',t')} \cup P|_{(t',t)} \end{aligned}$$

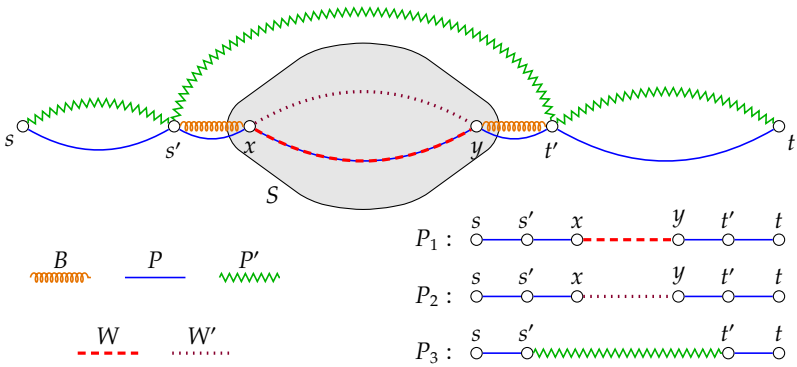


Figure 3.8: Construction for Case 2 in the proof of Theorem 3.6.1.

By construction $P_1, P_2 \in \mathcal{V}$ but $P_3 \notin \mathcal{V}$ since it does not use $(x', x) \in B$. P_1 and P_3 as well as P_2 and P_3 are adjacent in $P_{\text{s-t-flow}}(D)$ since they only differ in the disjoint paths from s' to t'. Again, this contradicts the fact that $\mathcal{V}$ is Λ-closed. □

ANALYZING SIMPLE EXTENSIONS OF POLYTOPES:

3.7 Perfect Matching Polytopes

The *matching polytope* and the *perfect matching polytope* of a graph $G = (V, E)$ are defined as

$$P_{\text{match}}(G) := \text{conv.hull}\{\chi(M) \mid M \text{ matching in } G\} \text{ and}$$
$$P^{\text{perf}}_{\text{match}}(G) := \text{conv.hull}\{\chi(M) \mid M \text{ perfect matching in } G\},$$

where $\chi(M) \in \{0,1\}^E$ is the characteristic vector of the set $M \subseteq E$, i.e., $\chi(M)_e = 1$ if and only if $e \in M$. Edmonds [23] showed that $P^{\text{perf}}_{\text{match}}(G)$ is the set of $x \in \mathbb{R}^E$ satisfying

$$x_e \geq 0 \qquad \text{for all } e \in E \tag{3.4}$$
$$x(\delta(v)) = 1 \qquad \text{for all } v \in V \tag{3.5}$$
$$x(\delta(U)) \geq 1 \qquad \text{for all } U \subseteq V \text{ with } |U| \text{ odd.} \tag{3.6}$$

If G is bipartite then only the nonnegativity constraints (3.4) and degree constraints (3.5) are required because the constraint matrix is totally unimodular.

Clearly, the symmetric difference $M \Delta M'$ of two perfect matchings is always a disjoint union of alternating cycles, so-called *M-M'-cycles*. Chvátal [17] showed that (the vertices corresponding to) two perfect matchings M and M' are adjacent if and only if $M \Delta M'$ forms a single alternating cycle.

This section establishes the simple extension complexities of the perfect matching polytopes (yielding lower bounds for the correspond-

ing matching polytopes' simple extension complexities) of two graph classes: Subsection 3.7.1 addresses the complete bipartite graphs $K_{n,n}$ with n nodes on both sides of the bipartition. Subsection 3.7.2 reduces the question for complete graphs K_{2n} with $2n$ nodes to an adjacency result on $P^{\text{perf}}_{\text{match}}(K_{2n})$. The latter is then established in Subsection 3.7.3.

3.7.1 Complete Bipartite Graphs

Our main theorem here reads as follows:

Theorem 3.7.1. *The simple extension complexity of the perfect matching polytope of $K_{n,n}$ is equal to the number $n!$ of its vertices.*

Proof. Let V and E denote the node and edge sets of $K_{n,n}$, respectively. Let $\mathcal{F}$ and $\mathcal{V}$ be the face and vertex sets induced by a proper facet of a simple extension of $P^{\text{perf}}_{\text{match}}(K_{n,n})$, respectively. The goal is to prove $|\mathcal{V}| = 1$, so let us assume for the sake of contradiction $|\mathcal{V}| \geq 2$. By Theorem 3.3.3 (b), the (inclusion-wise) maximal faces in $\mathcal{F}$ are facets. Since $K_{n,n}$ is bipartite, every facet is of the type $x_e \geq 0$ for some edge $e \in E$. Let $\emptyset \neq D' \subseteq E$ be the edge set corresponding to these facets. By Lemma 3.3.2, $\mathcal{V}$ is the set of (characteristic vectors of) perfect matchings $M \subseteq E$ satisfying $M \supseteq D'$. Let $D \supseteq D'$ be the set of *fixed* edges, i.e., edges common to all such matchings, and note that D is not the empty set, since D' is not.

Let M_1 be a perfect matching with $\chi(M_1) \in \mathcal{V}$, let $e_0 \in D$ be some fixed edge, and let $e_1, e_2 \in M_1 \setminus D$ be two of its edges that are not fixed. Note that e_1 and e_2 exist because otherwise $|M_1 \setminus D| \leq 1$ and then M_1 would be the only perfect matching containing D. Let $u_0, u_1, u_2 \in V$ be the endnodes of e_0, e_1 and e_2 that are on one side of the bipartition and $v_0, v_1, v_2 \in V$ be the other endnodes of e_0, e_1 and e_2, respectively.

Define $R := M_1 \setminus \{e_0, e_1, e_2\}$ and the two new perfect matchings $M_2 := R \cup \{e_0, \{u_1, v_2\}, \{u_2, v_1\}\}$ and $M_3 := R \cup \{\{u_0, v_1\}, \{u_1, v_2\}, \{u_2, v_0\}\}$. We clearly have $\chi(M_2) \in \mathcal{V}$ because $D \subseteq M_2$ holds and $\chi(M_3) \notin \mathcal{V}$, because $e_0 \notin M_3$.

Furthermore, we have that $M_1 \Delta M_3$ is the cycle u_0-v_0-u_2-v_2-u_1-v_1-u_0 and that $M_2 \Delta M_3$ is the cycle u_0-v_0-u_2-v_1-u_0, that is, M_3 (satisfying $\chi(M_3) \notin \mathcal{V}$) is adjacent to both M_1 and M_2 (satisfying $\chi(M_1), \chi(M_2) \in \mathcal{V}$). This contradicts Corollary 3.3.4, i.e., the fact that $\mathcal{V}$ is Λ-closed. □

Since $P^{\text{perf}}_{\text{match}}(K_{n,n})$ is a face of $P_{\text{match}}(K_{n,n})$ and simple extensions of polytopes induce simple extensions of their faces we obtain the following corollary for the latter polytope.

Corollary 3.7.2. *The simple extension complexity of the matching polytope of $K_{n,n}$ is at least $n!$.*

3.7.2 Complete Nonbipartite Graphs

From Theorem 3.7.1 and from the fact that the perfect matching polytope of the $K_{n,n}$ is a face of the perfect matching polytope of K_{2n} we immediately obtain the exponential lower bound of $n!$ on the simple extension complexity of the perfect matching polytope of K_{2n}.

Our main theorem of this section improves this bound drastically, and reads as follows:

Theorem 3.7.3. *The simple extension complexity of the perfect matching polytope of K_{2n} is equal to its number of vertices $\frac{(2n)!}{n!\cdot 2^n}$.*

We first give a high-level proof that uses a structural result presented in Subsection 3.7.3, since the latter may be interesting on its own.

Proof. The proof is based on Theorem 3.7.6. It states that for any three perfect matchings M_1, M_2, M_3 in K_{2n}, where M_1 and M_2 are adjacent (i.e., the corresponding vertices are adjacent), M_3 is adjacent to both M_1 and M_2 or there exists a fourth matching M' adjacent to all three matchings.

Let $P = P^{\text{perf}}_{\text{match}}(K_{2n})$ and suppose that $\mathcal{V}$ is a proper Λ-closed set with $|\mathcal{V}| \geq 2$. Since the polytope's graph is connected there exists a perfect matching M_1 with $\chi(M_1) \notin \mathcal{V}$ adjacent to some perfect matching M_2 with $\chi(M_2) \in \mathcal{V}$. Let M_3 be a perfect matching with $\chi(M_3) \in \mathcal{V} \setminus \{\chi(M_2)\}$. As $\mathcal{V}$ is Λ-closed and $\chi(M_3) \in \mathcal{V}$ holds, $\{M_1, M_2, M_3\}$ cannot be a triangle. Hence, by Theorem 3.7.6 mentioned above, there exists a common-neighbor perfect matching M'. Since M' is adjacent to M_2 and M_3 we conclude $\chi(M') \in \mathcal{V}$. But then M_1 (satisfying $\chi(M_1) \notin \mathcal{V}$) is adjacent to the two matchings M_2 and M' whose characteristic vectors are contained in $\mathcal{V}$, which contradicts the fact that $\mathcal{V}$ is Λ-closed.

Hence all proper Λ-closed sets are singletons, which implies the claim due to Corollary 3.3.6. □

Since $P^{\text{perf}}_{\text{match}}(K_{2n})$ is a face of $P_{\text{match}}(K_{2n})$ and simple extensions of polytopes induce simple extensions of their faces we obtain the following corollary for the latter polytope.

Corollary 3.7.4. *The simple extension complexity of the matching polytope of K_{2n} is at least $\frac{(2n)!}{n!\cdot 2^n}$.*

3.7.3 Adjacency Result

We now turn to the mentioned result on the adjacency structure of the perfect matching polytope of K_{2n}. It is a generalization of the diameter result of Padberg and Rao's in [52].

For an edge set F we denote by $V(F)$ the set of nodes covered by the edges of F. We start with an easy construction and modify the resulting matching later.

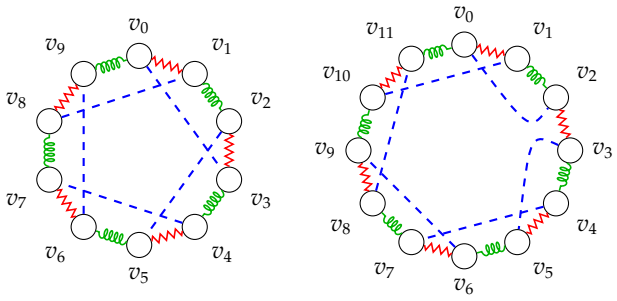


Figure 3.9: Lemma 3.7.5 for a 10-cycle and a 12-cycle.

Lemma 3.7.5. *For any adjacent perfect matchings M_1, M_2 there exists a perfect matching M' adjacent to M_1 and M_2 that satisfies*

$$V(M_1 \Delta M_2) = V(M_1 \Delta M') = V(M_2 \Delta M')$$

and $M' \cap (M_1 \Delta M_2) = \emptyset$.

Proof. Let $v_0, v_1, \ldots, v_{2\ell-1}, v_{2\ell} = v_0$ be the set of ordered nodes of the cycle $M_1 \Delta M_2$ and identify $v_{2\ell+1} = v_1$. If ℓ is odd,

$$M' := \{\{v_i, v_{i+3}\} \mid i = 0, 2, 4, 6, \ldots, 2l-2\}$$

induces M_i-M'-cycles visiting the nodes in the following order:

$$\begin{aligned} M_1 \Delta M' : &\quad v_0, v_3, v_2, v_5, v_4, v_7, v_6, \ldots, v_{2\ell-1}, \\ &\quad v_{2\ell-2}, v_1, v_0 \\ M_2 \Delta M' : &\quad v_0, v_3, v_4, v_7, v_8, \ldots, v_{2l-3}, v_{2\ell-2}, \\ &\quad v_1, v_2, v_5, v_6, \ldots, v_{2\ell-4}, v_{2\ell-1}, v_0 \end{aligned}$$

If ℓ is even,

$$M' := \{\{v_i, v_{i+3}\} \mid i = 4, 6, \dots, 2\ell - 2\} \\ \cup \{\{v_0, v_2\}, \{v_3, v_5\}\}$$

induces M_i-M'-cycles visiting the nodes in the following order:

$$\begin{aligned} M_1 \Delta M' : \quad & v_0, v_2, v_3, v_5, v_4, v_7, v_6, \dots, v_{2\ell-1}, \\ & v_{2\ell-2}, v_1, v_0 \\ M_2 \Delta M' : \quad & v_0, v_2, v_1, v_{2\ell-2}, v_{2\ell-3}, \dots, v_6, v_5, v_3, \\ & v_4, v_7, v_8, \dots, v_{2\ell-4}, v_{2\ell-1}, v_0 \end{aligned}$$

Figure 3.9 shows examples for both cases. It is easy to see that the node sets of the cycles equals the node set of $M_1 \Delta M_2$ and that $M' \cap (M_1 \Delta M_2) = \emptyset$ holds. In order to produce a perfect matching on all nodes we simply add $M_1 \cap M_2$ to M', which does not change any of the two required properties. □

Suppose there is a third perfect matching M_3 and we want to make M' adjacent to this matching as well. The remainder of this section is dedicated to the proof of the following result.

Theorem 3.7.6. *Let M_1 and M_2 be two adjacent perfect matchings and M_3 a third perfect matching. Then the three matchings are pairwise adjacent or there exists a perfect matching M' adjacent to all three.*

Before we state the proof, we introduce the notion of good perfect matchings. The first part of the proof is dedicated to proving their existence, while the second part shows that good perfect matchings that are minimal in a certain sense satisfy the properties claimed by Theorem 3.7.6.

We first fix some notation for the rest of this section. Let M_1, M_2 and M_3 be three perfect matchings such that M_1 and M_2 are adjacent. Denote by $V^* := V(M_1 \Delta M_2)$ the node set of the single alternating M_1-M_2-cycle.

For a perfect matching M' we denote by M_3-M'-*components* the connected components of $M_3 \cup M'$ and by $c(M_3, M')$ their number. We call a perfect matching M' *good* if the following five properties hold:

(A) M' is adjacent to M_1 and M_2.

(B) All M_3-M'-components touch the node-set V^* of $M_1 \Delta M_2$.

(C) All M'-edges that also belong to the M_1-M_2-cycle, i.e., the edges from $M' \cap (M_1 \Delta M_2)$, are contained in the same M_3-M'-component.

(D) $M_3 \neq M'$ and $c(M_3, M') \leq \frac{1}{2}|M_1 \Delta M'| + \frac{1}{2}|M_2 \Delta M'| - 3$ holds.

(E) $c(M_3, M') \leq \frac{1}{2}|M_j \Delta M'|$ holds for $j = 1, 2$ and equality holds only if we have $V(M_k \Delta M') \supseteq V^*$ for $k = 3 - j$, i.e., $\{M_j, M_k\} = \{M_1, M_2\}$.

We first establish the existence of good perfect matchings.

Lemma 3.7.7. *Let M_1, M_2, M_3 be three perfect matchings of K_{2n} such that $M_1 \cap M_2 \cap M_3 = \emptyset$ holds and such that M_1 and M_2 are adjacent, but M_3 is not adjacent to both of them. Then there exists a good perfect matching M'.*

Proof. Let $\overline{M}$ be the perfect matching adjacent to M_1 and M_2 constructed in Lemma 3.7.5.

Note that it satisfies $\overline{M} \cap (M_1 \Delta M_2) = \emptyset$ as well as $|M_1 \Delta \overline{M}| = |M_2 \Delta \overline{M}| = |M_1 \Delta M_2| = |V^*| \geq 4$. We now enlarge the M_i-$\overline{M}$-cycles ($i = 1, 2$) in order to remove M_3-$\overline{M}$-cycles that do not touch V^* in order to satisfy Property (B).

Let $\{u_0, v_0\}$ be an $\overline{M}$-edge with $u_0, v_0 \in V^*$. Let $C_1, C_2, \ldots, C_s$ be all M_3-$\overline{M}$-cycles with $V(C_i) \cap V^* = \emptyset$ and let, for $i = 1, 2, \ldots, s$, $\{u_i, v_i\} \in C_i \cap \overline{M}$ be any $\overline{M}$-edge of C_i. Define M' to be

$$M' := \left(\overline{M} \setminus \{\{u_i, v_i\} \mid i = 0, 1, \ldots, s\}\right) \cup \{\{u_i, v_{i+1}\} \mid i = 0, 1, \ldots, s\}, \tag{3.7}$$

where $v_{s+1} = v_0$ (see Figure 3.10).

We now verify Property (A), i.e., that M' is adjacent to M_i ($i = 1, 2$). Since the cycles $C_1, \ldots, C_s$ do not touch V^*, $\overline{M}$ and M_i coincide outside V^*. Hence, the modification replaces the $\overline{M}$-edge $\{u_0, v_0\}$ by an alternating $\overline{M}$-M_i-path from u_0 to v_0 that visits exactly 2 nodes of each C_i, thus indeed M' is adjacent to both M_1 and M_2.

In order to prove Properties (B) and (E), let us recall some properties of the M_3-M'-components: The M_3-M'-cycle constructed above contains nodes u_0 and v_0. All other M_3-M'-cycles were also M_3-$\overline{M}$-cycles, and hence, by definition of the C_i above, have at least two nodes of V^* in

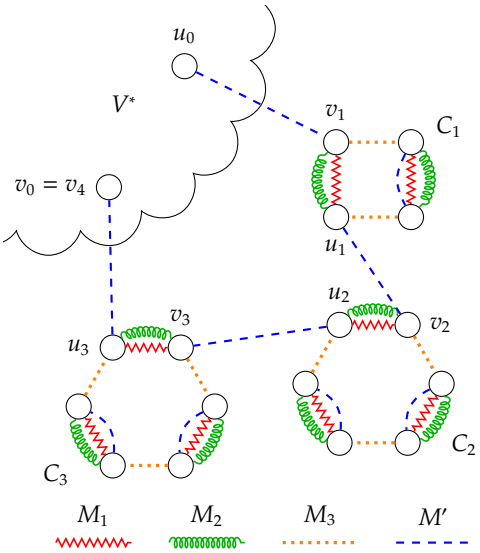


Figure 3.10: Construction in Lemma 3.7.7 with 3 outer cycles.

common since one of their $\overline{M}$-edges has both endpoints in V^*. All edges in $M_3 \cap M'$ must also lie in V^* since outside $V^* \cup V(C_1) \cup \ldots V(C_s)$ the matchings M', M_1 and M_2 are the same and $M_1 \cap M_2 \cap M_3 = \emptyset$ holds.

Hence all M_3-M'-components have at least two nodes in V^*, in particular, Property (B) holds. It also proves the inequality statement of Property (E) because $V(M_j \Delta M') \supseteq V^*$ for $j = 1, 2$. Furthermore, the containment statement is due to Lemma 3.7.5, since we have $V(M_k \Delta M') \supseteq V(M_k \Delta \overline{M}) = V^*$ for $k = 1, 2$.

In order to verify Property (C), observe that $\overline{M} \cap (M_1 \Delta M_2)$ is empty. Since all edges in M' that were not in $\overline{M}$ have at least one endpoint outside V^*, we also have $M' \cap (M_1 \Delta M_2) = \emptyset$. Hence, Property (C) is satisfied trivially.

It remains to show that Property (D) holds. Clearly, since M_3 is adjacent to at most one of M_1, M_2, we have $M_3 \neq M'$. Since we established as part of Property (E), that $c(M_3, M') \leq \frac{1}{2}|M_1 \Delta M'|$ holds, it suffices to show that $|M_2 \Delta M'|$ is at least 6. Suppose, for the sake of contradic-

tion, that this is not the case, i.e., $|M_2 \Delta M'| \leq 4$ holds, which in turn implies $c(M_3, M') \leq 2$. Also $|M_2 \Delta M'| \geq 4$ holds since both matchings are adjacent. This implies that we have equality in the containment $V(M_2 \Delta M') \supseteq V(M_2 \Delta \overline{M}) = V^*$, from which we conclude that $s = 0$ holds, i.e., $M' = \overline{M}$. These properties already prove that the M'-edges that match the nodes of V^* are exactly the two chords of the M_1-M_2-cycle (see Figure 3.11). It is now easy to verify that then M_1, M_2 and M_3 must be pairwise adjacent, a contradiction to the assumptions of this lemma.

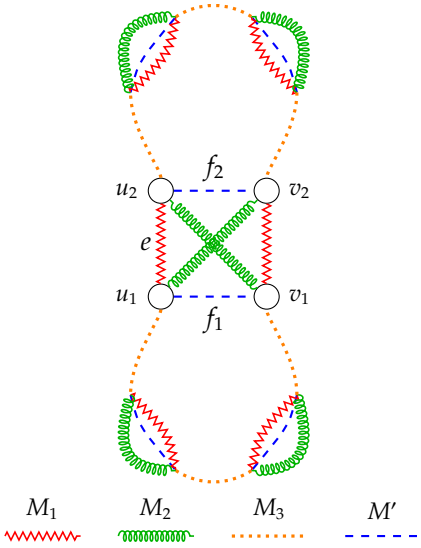


Figure 3.11: A special case in the proof of Lemma 3.7.7 where M_3 is adjacent to M_1 and M_2.

□

Proof of Theorem 3.7.6. Let M_1, M_2, M_3 be as stated in the Theorem. We assume, without loss of generality, that $M_1 \cap M_2 \cap M_3 = \emptyset$ holds, since otherwise we can restrict ourself to the graph with the nodes of this set deleted. We also assume that M_1, M_2 and M_3 are not pairwise adjacent, since otherwise there is nothing to prove.

In this situation, Lemma 3.7.7 guarantees the existence of a good perfect matching. Let M' be a good perfect matching with minimum $c(M_3, M')$. If $c(M_3, M') = 1$ holds, M' is adjacent to M_3 (and also adjacent to M_1 and M_2 by Property (A)) and we are done. Hence, for the sake of contradiction, we from now on assume that $c(M_3, M')$ is at least 2. The strategy is to construct another good matching M^* with $c(M_3, M^*) < c(M_3, M')$.

Due to Property (C), there exists an M_3-M'-component $\widehat{C}$ containing all edges (if any) from $M' \cap (M_1 \Delta M_2)$. $M_1 \Delta M_2$ is a single cycle visiting all nodes in V^* all of which are in some M_3-M'-component as they are matched by M'. Thus, by Property (B), the component $\widehat{C}$ is connected to at least one other M_3-M'-component by an edge from M_1-M_2. Let us choose such an edge $e \in M_j \setminus M_k$ for some $j = 1, 2$ and $k = 3 - j$, and if such edges exist for both values of j, choose e such that $|M_j \Delta M'|$ is maximum.

We claim that $|M_j \Delta M'| \geq 6$ holds. Suppose, for the sake of contradiction, $|M_j \Delta M'| = 4$. Property (E) implies that $c(M_3, M') \leq 2$ holds. Since $c(M_3, M') \geq 2$ also holds, we have equality and then Property (E) implies that the M_k-M'-cycle covers all nodes in V^*. This cycle connects the only two M_3-M'-components $\widehat{C}$ and C' via at least two M_k-edges f, f' since the M'-edges of the cycle are inside their respective components. Note that $|M_k \Delta M'| \geq 2c(M_3, M') + 6 - |M_j \Delta M'| \geq 6$ holds by Property (D). Hence, by the maximality assumption for the choice of edge e, this implies that there is no edge in $M_k \setminus M_j$ that connects $\widehat{C}$ to C'. Since $f, f' \in M_k$ both connect $\widehat{C}$ to C', it follows that $f, f' \in M_j$ holds as well. Because $|M_j \Delta M'| = 4$ holds we have $M_j \setminus M' = \{f, f'\}$. Hence, the alternating M_j-M'-cycle of length 4 is also an alternating M_k-M'-cycle. But the latter has at least length 6 as argued above, which yields a contradiction.

To summarize, we now have two distinct M_3-M'-components $\widehat{C}$ and C' connected by an edge $e \in M_j \setminus M_k$ for some $j = 1, 2$ such that $|M_j \Delta M'| \geq 6$ holds and all edges from $M' \cap (M_1 \Delta M_2)$ (if any) are in $\widehat{C}$.

Let $u_1 \in V(\widehat{C})$ and $u_2 \in V(C')$ be the endpoints of edge e. Let $f_1 = \{u_1, v_1\}, f_2 = \{u_2, v_2\} \in M'$ be the edges matching u_1 and u_2. We clearly have $v_1 \in V(\widehat{C})$ and $v_2 \in V(C')$ as well since f_1 and f_2 are contained in their respective M_3-M'-components. In particular we have $f_1, f_2 \neq e$ and u_1, u_2, v_1, v_2 are pairwise distinct nodes. Because $f_2 \notin M_j$ ($e \in M_j$ and

$e \cap f_2 \neq \emptyset$) and $f_2 \notin V(\widehat{C})$ hold, Property (C) implies $f_2 \notin M_k$ (note that $\widehat{C}$ is the component mentioned in Property (C)).

If also $f_1 \notin M_k$ holds, f_1 and f_2 belong to $M_k \Delta M'$, which is a single cycle by Property (A), and hence there exists a walk W on $M_k \Delta M'$ starting in u_1 with edge f_1 that visits nodes u_2 and v_2 (in some order).

We are now ready to create a new good perfect matching M^* related to M' by small changes. For this, we distinguish two cases. For each case we establish Property (A) separately, and afterwards prove the remaining properties for both cases in parallel.

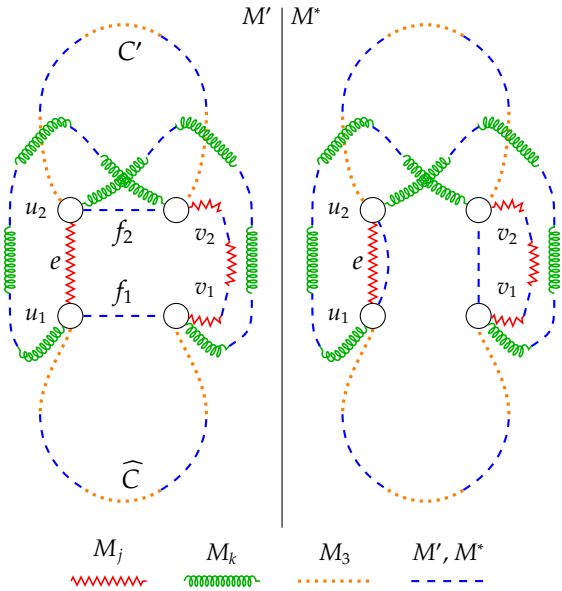


Figure 3.12: Modifications in Case 1 in the proof of Theorem 3.7.6.

Case 1: $f_1 \notin M_k$ holds and u_2 comes before v_2 on walk W.
Let $M^* := (M' \setminus \{f_1, f_2\}) \cup \{\{u_1, u_2\}, \{v_1, v_2\}\}$ (see Figure 3.12). We now prove Property (A) for M^*. The symmetric difference $M_k \Delta M^*$ consists of a single cycle that arises from the cycle $M_k \Delta M'$ by removing edges f_1, f_2

and adding edges $\{u_2, u_1\}$ and $\{v_2, v_1\}$. For M_j, the situation is different, since there is a new component $e \in M_j \cap M^*$. The new M_j-M^*-cycle is now 2 edges shorter than the M_j-M'-cycle was before since it visits the edge $\{v_1, v_2\}$ instead of the path $v_1 - u_1 - u_2 - v_2$. But because we ensured $|M_j \Delta M'| \geq 6$ before, we have $|M_j \Delta M^*| \geq 4$, that is, M^* is also adjacent to M_j.

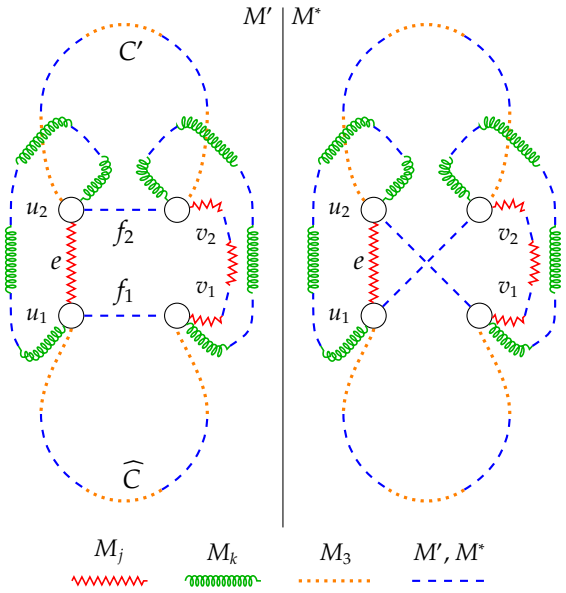


Figure 3.13: Modifications in Case 2 in the proof of Theorem 3.7.6.

Case 2: $f_1 \in M_k$ holds or $f_1 \notin M_k$ and u_2 comes after v_2 on walk W.
Let $M^* := (M' \setminus \{f_1, f_2\}) \cup \{\{u_1, v_2\}, \{u_2, v_1\}\}$ (see Figure 3.13). We now prove Property (A) for M^*. The symmetric difference $M_k \Delta M^*$ consists of a single cycle that arises from the cycle $M_k \Delta M'$ by removing edges f_1, f_2 and adding edges $\{v_2, u_1\}$ and $\{u_2, v_1\}$. There is also only one M_j-M^*-cycle, which is essentially equal to the M_j-M'-cycle, except that the path $v_1 - u_1 - u_2 - v_2$ was replaced by the path $v_1 - u_2 - u_1 - v_2$.

In Case 1 as well as in Case 2, M^* is again a perfect matching since M' was a perfect matching and they differ only in the way the nodes u_1, u_2, v_1, v_2 are matched. Furthermore, M^* connects the two components $\widehat{C}$ and C', that is, $c(M_3, M^*) = c(M_3, M') - 1$. In order to create the desired contradiction to the minimality of $c(M_3, M')$, it remains to prove that M^* satisfies Properties (B),(C), (D) and (E).

Property (B) is satisfied for M^* because $V(\widehat{C})$ is contained in an M_3-M^*-component and all edges in $M^* \setminus M'$ are contained in the same component.

Property (C) is also satisfied for M^* since all M^*-edges that were not M'-edges before, are contained in cycle $\widehat{C}$ (which was by definition the only M_3-M'-cycle containing edges in $M' \cap (M_1 \Delta M_2)$).

We now prove that Property (D) is satisfied for M^*. We have

$$\begin{aligned} c(M_3, M^*) + 3 = c(M_3, M') + 3 - 1 &\leq \frac{1}{2}|M_1 \Delta M'| + \frac{1}{2}|M_2 \Delta M'| - 1 \\ &= \frac{1}{2}|M_k \Delta M'| + \frac{1}{2}|M_j \Delta M'| - 1 \\ &\leq \frac{1}{2}|M_k \Delta M^*| + \frac{1}{2}\Big(|M_j \Delta M^*| + 2\Big) - 1 \\ &= \frac{1}{2}|M_1 \Delta M^*| + \frac{1}{2}|M_2 \Delta M^*|, \end{aligned}$$

where the first inequality is due to Property (D) for M' and the last inequality comes from the fact that in Case 1, $M_j \Delta M^*$ has two fewer edges than $M_j \Delta M'$ and in Case 2, the cardinalities agree.

By similar arguments, $c(M_3, M^*) \leq \frac{1}{2}|M_i \Delta M^*|$ holds for $i = 1, 2$. In order to prove that Property (E) is satisfied for M^*, assume that $c(M_3, M^*) = \frac{1}{2}|M_i \Delta M^*|$ holds for some $i \in \{1, 2\}$. Due to $c(M_3, M^*) = c(M_3, M') - 1$, this implies $i = j$ and we are in Case 1 since only there $|M_i \Delta M^*|$ is less than $|M_i \Delta M'|$, and also have $c(M_3, M') = \frac{1}{2}|M_j \Delta M'|$. Property (E) of M' guarantees that $V(M_k \Delta M') \supseteq V^*$ holds. But since the node sets of $M_k \Delta M'$ and $M_k \Delta M^*$ are the same, we also have $V(M_k \Delta M^*) \supseteq V^*$.

We proved that M^* is a good perfect matching, yielding the required contradiction to the minimality assumption of $c(M_3, M')$ which completes the proof. □

ANALYZING SIMPLE EXTENSIONS OF POLYTOPES:

3.8 A Question Relating Simple Extensions with Diameters

Let us make a brief digression on the potential relevance of simple extensions with respect to questions related to the diameter of a polytope, i.e., the maximum distance (minimum number of edges on a path) between any pair of vertices in the graph of the polytope. We denote by $\Delta(d, m)$ the maximum diameter of any d-dimensional polytope with m facets. It is well-known that $\Delta(d, m)$ is attained by simple polytopes. A necessary condition for a polynomial time variant of the simplex-algorithm to exist is that $\Delta(d, m)$ is bounded by a polynomial in d and m (thus by a polynomial in m). In fact, in 1957 Hirsch even conjectured (see [21]) that $\Delta(d, m) \leq m - d$ holds, which has only rather recently been disproved by Santos [60]. However, still it is even unknown whether $\Delta(d, m) \leq 2m$ holds true, and the question, whether $\Delta(d, m)$ is bounded polynomially (i.e., whether the *polynomial Hirsch-conjecture* is true) is a major open problem in Discrete Geometry.

In view of the fact that linear optimization over a polytope can be performed by linear optimization over any of its extensions, a reasonable relaxed version of this question might be to ask whether every d-dimensional polytope P with m facets admits an extension whose size and diameter both are bounded polynomially in m. Stating the relaxed question in this naive way, the answer clearly is positive, as one may construct an extension by forming a pyramid over P (after embedding P into $\mathbb{R}^{\dim(P)+1}$), which has diameter two. However, in some accordance

with the way the simplex algorithm works by pivoting between bases rather than only by proceeding along edges, it seems to make sense to require the extension to be simple (which a pyramid, of course, in general is not). But still, this is not yet a useful variation, since our result on flow polytopes (see Theorem 3.6.1) shows that there are polytopes that even do not admit a polynomial (in the number of facets) size simple extension at all. Therefore, we propose to investigate the following question, whose positive answer would be implied by establishing the polynomial Hirsch-conjecture (as every polytope is an extension of itself).

Question 3.8.1. *Does there exist a polynomial q such that every* simple *polytope P with m facets has a* simple *extension Q with at most $q(m)$ many facets and diameter at most $q(m)$?*

In order to make progress on this question, one might want to investigate the face-lattice embeddings for projections of simple extensions of simple polytopes. On the one hand, all lower bounds we obtained in this chapter are for a non-simple polytope. On the other hand, all constructive results are for simple polytopes, e.g., the completion-time polytopes mentioned in the introduction (see Corollary 4.5 in [57] for a proof of this fact).

Bibliography

[1] Tobias Achterberg. SCIP: Solving constraint integer programs. *Math. Program. Ser. C*, 1(1):1–41, 2009.

[2] Hatem Ben Amor and Jacques Desrosiers. A proximal trust-region algorithm for column generation stabilization. *Computers & Operations Research*, 33(4):910 – 927, 2006.

[3] David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*, chapter TSP Cuts Which Do Not Conform to the Template Paradigm, pages 261–303. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[4] David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. Concorde TSP solver, `http://www.math.uwaterloo.ca/tsp/concorde.html`, 2006.

[5] David L. Applegate, William Cook, Sanjeeb Dash, and Daniel G. Espinoza. *QSopt_ex*, `http://www.dii.uchile.cl/daespino/ESolver_doc/main.html`, 2007.

[6] David L. Applegate, William Cook, Sanjeeb Dash, and Daniel G. Espinoza. Exact solutions to linear programming problems. *Operations Research Letters*, 35(6):693 – 699, 2007.

[7] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[8] Benjamin Assarf, Ewgenij Gawrilow, Katrin Herr, Michael Joswig, Benjamin Lorenz, Andreas Paffenholz, and Thomas Rehn. polymake in linear and integer programming. *arXiv preprint arXiv:1408.4653*, 2014.

[9] David Avis and Hans Raj Tiwary. On the extension complexity of combinatorial polytopes. In Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg, editors, *Automata, Languages, and Programming*, volume 7965 of *Lecture Notes in Computer Science*, pages 57–68. Springer Berlin Heidelberg, 2013.

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

[11] Egon Balas. Disjunctive programming. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 3 – 51. Elsevier, 1979.

[12] Abbas Bazzi, Samuel Fiorini, Sebastian Pokutta, and Ola Svensson. No small linear program approximates vertex cover within a factor $2 - \varepsilon$. arXiv:1503.00753, 2015.

[13] Daniel Bienstock. Approximate formulations for 0-1 knapsack sets. *Operations Research Letters*, 36(3):317 – 320, 2008.

[14] Robert E. Bixby, E. Andrew Boyd, and Ronni R. Indovina. Miplib: A test set of mixed-integer programming problems. *SIAM News*, 25, 1992.

[15] Christoph Buchheim, Frauke Liers, and Marcus Oswald. Local cuts revisited. *Operations Research Letters*, 36(4):430 – 433, 2008.

[16] Robert D. Carr and Goran Konjevod. Polyhedral combinatorics. In H. J. Greenberg, editor, *Tutorials on Emerging Methodologies and Applications in Operations Research*, volume 76 of *International Series in Operations Research & Management Science*, chapter 2, pages 1–46. Springer, 2005.

[17] Vasek Chvátal. On certain polytopes associated with graphs. *J. Combin. Theory Ser. B*, 18(2):138 – 154, 1975.

[18] Michele Conforti, Volker Kaibel, Matthias Walter, and Stefan Weltge. Subgraph polytopes and independence polytopes of count matroids. *Operations Research Letters*, 43(5):457 – 460, Sep 2015.

[19] William Cook, Thorsten Koch, Daniel E Steffy, and Kati Wolter. A hybrid branch-and-bound approach for exact rational mixed-integer programming. *Mathematical Programming Computation*, 5(3):305–344, 2013.

[20] William Cook, Thorsten Koch, DanielE. Steffy, and Kati Wolter. An exact rational mixed-integer programming solver. In Oktay Günlük and GerhardJ. Woeginger, editors, *Integer Programming and Combinatoral Optimization*, volume 6655 of *Lecture Notes in Computer Science*, pages 104–116. Springer Berlin Heidelberg, 2011.

[21] George B. Dantzig. *Linear Programming and Extensions*. Princeton landmarks in mathematics and physics. Princeton University Press, 1963.

[22] Olivier du Merle, Daniel Villeneuve, Jacques Desrosiers, and Pierre Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1–3):229 – 237, 1999.

[23] Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B*, 69:125–130, 1965.

[24] Friedrich Eisenbrand and Sören Laue. A faster algorithm for two-variable integer programming. In *Algorithms and Computation*, pages 290–299. Springer, 2003.

[25] Paul Erdös, Siemion Fajtlowicz, and Alan J Hoffman. Maximum degree in graphs of diameter 2. *Networks*, 10(1):87–90, 1980.

[26] Samuel Fiorini, Volker Kaibel, Kanstantsin Pashkovich, and Dirk Oliver Theis. Combinatorial bounds on nonnegative rank and extended formulations. *Discrete Math.*, 313(1):67–83, 2013.

[27] Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary, and Ronald de Wolf. Linear vs. semidefinite extended formulations: exponential separation and strong lower bounds. In Howard J. Karloff and Toniann Pitassi, editors, *STOC*, pages 95–106. ACM, 2012.

[28] Giorgio Gallo and Claudio Sodini. Extreme points and adjacency relationship in the flow polytope. *Calcolo*, 15:277–288, 1978.

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

[30] Rafael Gillmann. *0/1-Polytopes Typical and Extremal Properties*. PhD thesis, Technische Universität Berlin, 2007.

[31] Michel Goemans. Smallest compact formulation for the permutahedron. `http://www-math.mit.edu/~goemans/publ.html`, 2009.

[32] Michel X. Goemans and Young-Soo Myung. A Catalog of Steiner Tree Formulations. *Networks*, 23(1):19–28, 1993.

[33] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[34] Martin Grötschel and Manfred W. Padberg. Polyhedral theory. In E. L. Lawler, Jan Karel Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors, *The traveling salesman problem: a guided tour of combinatorial optimization*, Wiley Interscience Series in Discrete Mathematics, page 476. John Wiley & Sons, Inc., 1985.

[35] Ishay Haviv and Oded Regev. Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 469–477. ACM, 2007.

[36] Emilie V. Haynsworth. On the schur complement. *Basel Mathematical Notes*, 20, 1968.

[37] Lena Hupp, Laura Klein, and Frauke Liers. An exact solution method for quadratic matching: The one-quadratic-term technique and generalisations. *Discrete Optimization*, 18:193 – 216, 2015.

[38] Cor A. J. Hurkens. Blowing up convex sets in the plane. *Linear Algebra Appl.*, 134:121–128, 1990.

[39] Volker Kaibel and Kanstantsin Pashkovich. Constructing extended formulations from reflection relations. In O. Günlük and G. Woeginger, editors, *Integer Programming and Combinatorial Optimization. Proceedings of IPCO XV, New York, NY*, volume 6655 of *Lecture Notes in Computer Science*, pages 287–300. Springer, 2011.

[40] Volker Kaibel, Kanstantsin Pashkovich, and Dirk Oliver Theis. Symmetry matters for sizes of extended formulations. *SIAM J. Disc. Math.*, 26(3):1361–1382, 2012.

[41] Volker Kaibel and Matthias Walter. Simple extensions of polytopes. *Math. Program. Ser. B*, 154(1-2):381–406, 2015.

[42] Volker Kaibel and Stefan Weltge. Lower bounds on the sizes of integer programs without additional variables. *Math. Program. Ser. B*, 154(1-2):407–425, 2015.

[43] Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the smith and hermite normal forms of an integer matrix. *siam Journal on Computing*, 8(4):499–507, 1979.

[44] Ravindran Kannan and László Lovász. Covering minima and lattice-point-free convex bodies. *Annals of Mathematics*, 128:577–602, 1988.

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

[46] Thorsten Koch and Alexander Martin. Solving Steiner Tree Problems in Graphs to Optimality. *Networks*, 32(3):207–232, 1998.

[47] Bernhard Korte and Jens Vygen. *Combinatorial Optimization*, volume 21. Springer, fifth edition, 2012.

[48] Hendrik W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[49] François Margot, Alain Prodon, and Thomas M. Liebling. Tree polytope on 2-trees. *Mathematical Programming*, 63(1-3):183–191, 1994.

[50] R. Kipp Martin. Using separation algorithms to generate mixed integer model reformulations. *Oper. Res. Lett.*, 10(3):119–128, 1991.

[51] Garth P. McCormick. Computability of global solutions to factorable nonconvex programs: Part i — convex underestimating problems. *Mathematical Programming*, 10(1):147–175, 1976.

[52] Manfred W. Padberg and Mendu Rammohan Rao. The travelling salesman problem and a class of polyhedra of diameter two. *Math. Program.*, 7:32–45, 1974. 10.1007/BF01585502.

[53] Christos H. Papadimitriou. The adjacency relation on the traveling salesman polytope is np-complete. *Mathematical Programming*, 14(1):312–324, 1978.

[54] Kanstantsin Pashkovich. Tight lower bounds on the sizes of symmetric extensions of permutahedra and similar results. *Mathematics of Operations Research*, 39(4):1330–1339, 2014.

[55] Sebastian Pokutta and Mathieu Van Vyve. A note on the extension complexity of the knapsack polytope. *Oper. Res. Lett.*, 41(4):347–350, 2013.

[56] The polymake team. *polymake*, `http://www.polymake.org`, 2015.

[57] Maurice Queyranne. Structure of a simple scheduling polyhedron. *Math. Program.*, 58(1-3):263–285, 1993.

[58] Thomas Rothvoss. Some 0/1 polytopes need exponential size extended formulations. *Mathematical Programming*, 142(1-2):255–268, 2013.

[59] Thomas Rothvoss. The matching polytope has exponential extension complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 263–272, New York, NY, USA, 2014. ACM.

[60] Francisco Santos. A counterexample to the hirsch conjecture. *Annals of Mathematics. Second Series*, 176(1):383–412, 2012.

[61] Rolf Schneider. *Convex bodies: the Brunn–Minkowski theory*. Number 151. Cambridge University Press, 2013.

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

[63] Alexander Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer, 2003.

[64] Daniel E. Steffy and Kati Wolter. Valid linear programming bounds for exact mixed-integer programming. *INFORMS Journal on Computing*, 25(2):271–284, 2013.

[65] Leena M. Suhl and Uwe H. Suhl. A fast LU update for linear programming. *Annals of Operations Research*, 43(1):33–47, 1993.

[66] Uwe H. Suhl and Leena M. Suhl. Computing Sparse LU Factorizations for Large-Scale Linear Programming Bases. *ORSA Journal on Computing*, 2(4):325–335, 1990.

[67] Matthias Walter. *IPO – Investigating Polyhedra by Oracles*, `http://polyhedra-oracles.bitbucket.org/`, 2016.

[68] Roland Wunderling. *Paralleler und objektorientierter Simplex-Algorithmus*. PhD thesis, ZIB, 1997.

[69] Mihalis Yannakakis. Expressing combinatorial optimization problems by linear programs. *J. Comput. Syst. Sci.*, 43(3):441–466, 1991.

[70] Günter M. Ziegler. *Lectures on Polytopes (Graduate Texts in Mathematics)*. Springer, 2001.

APPENDIX:

A.1 Computational Studies: Dimensions

As in Section 2.9, the linear relaxations of the instances are denoted by P, the linear relaxations of the presolved instances are denoted by Q, and the respective mixed-integer hulls by P_I and Q_I. Tables A.1–A.8 display the following measurements of our implementation of Algorithm 2.4.3 for these four polyhedra:

n	Ambient dimension of the polyhedron.
$\max^{\text{bit}}$	Maximum encoding length of a direction vector.
t	Overall time.
k^{cach} / t^{cach}	Number of / time spent for the cache searches.
k^{heur} / t^{heur}	Number of / time spent for the heuristic calls.
k^{orac} / t^{orac}	Number of / time spent for the oracle calls.
k^{apx} / t^{apx}	Number of / time spent for approximate directions.
k^{exct} / t^{exct}	Number of / time spent for exact directions.
k^{fact} / t^{fact}	Number of / time spent for factorization updates.

As before, all times are given in seconds. Note that for the linear relaxations, since the heuristic always returns an optimal solution, the values k^{orac} and t^{orac} were zero for all instances and hence are omitted in the tables. Also note that the affine hull of `p0548`'s mixed-integer hull could not be computed due to a timeout in one of the oracle calls.

Table A.1: Statistics for relaxations of MIPLIB 2.0 instances.

Instance	n	$\max_P^{\text{bit}}$	k_P^{cach}	k_P^{heur}	k_P^{apx}	k_P^{exct}	k_P^{fact}
air01	771	544,192	750	769	296,827	768	733
bell3b	133	511,962	133	149	8,912	133	134
bell5	104	287,557	104	117	5,461	104	105
bm23	27	40,650	27	28	379	27	28
cracpb1	572	2,749,540	484	484	159,963	557	485
dcmulti	548	4,434,537	470	479	147,346	547	471
diamond	2	2	2	1	4	2	3
egout	141	12,458	68	103	7,311	131	69
enigma	100	181,576	79	80	4,820	90	80
flugpl	18	1,365	12	13	151	18	13
gen	870	19,784,443	720	726	367,561	869	721
lseu	89	25,093	89	99	4,006	89	90
misc01	83	30,118	68	79	3,211	82	61
misc02	59	2,882	47	58	1,600	54	42
misc03	160	106,686	136	152	12,101	159	122
misc05	136	289,569	108	117	8,651	131	101
misc07	260	291,163	228	256	32,760	255	208
mod008	319	2	319	638	51,041	319	320
mod013	96	1,646	83	131	4,566	91	84
p0033	33	7,699	33	48	562	33	34
p0040	40	9,200	40	50	776	40	31
p0201	201	49,114	201	259	18,706	201	146
p0282	282	3,611	282	323	39,904	282	283
p0291	291	1,104	291	337	42,487	291	292
p0548	548	1,484,208	548	612	150,421	548	546
pipex	48	22,575	32	28	1,041	47	33
rgn	180	265,958	160	168	16,081	179	161
sample2	67	347	44	50	2,003	62	45
sentoy	60	2	60	60	1,831	60	61
stein15	15	2	15	30	121	15	16
stein27	27	2	27	54	379	27	28
stein45	45	2	45	90	1,036	45	46
stein9	9	2	9	18	46	9	10
vpm1	378	106,491	336	411	67,537	377	289

Table A.2: Timings for relaxations of MIPLIB 2.0 instances.

Instance	t_P	t_P^{cach}	t_P^{heur}	t_P^{apx}	t_P^{exct}	t_P^{fact}
air01	64	0.1	46.3	2.6	12.1	2.4
bell3b	25	0.1	12.1	0.1	8.3	3.9
bell5	13	0.1	6.9	0.1	4.2	2.0
bm23	1	0.0	1.1	0.0	0.1	0.0
cracpb1	1,919	0.4	288.8	9.8	1,182.2	437.2
dcmulti	2,969	1.9	261.7	13.6	1,924.3	767.6
diamond	0	0.0	0.0	0.0	0.0	0.0
egout	2	0.0	1.9	0.1	0.0	0.0
enigma	3	0.0	2.6	0.0	0.5	0.2
flugpl	0	0.0	0.1	0.0	0.0	0.0
gen	28,870	17.9	1,573.9	72.6	19,715.8	7,489.9
lseu	1	0.0	0.9	0.0	0.0	0.0
misc01	3	0.0	2.7	0.0	0.2	0.1
misc02	1	0.0	0.6	0.0	0.0	0.0
misc03	13	0.0	9.5	0.1	2.5	1.0
misc05	25	0.0	18.1	0.0	4.8	1.8
misc07	67	0.0	38.9	0.4	20.0	7.8
mod008	22	0.0	20.0	0.5	0.2	1.3
mod013	1	0.0	1.0	0.0	0.0	0.0
p0033	0	0.0	0.3	0.0	0.0	0.0
p0040	0	0.0	0.4	0.0	0.0	0.0
p0201	7	0.0	6.2	0.1	0.5	0.1
p0282	8	0.0	6.5	0.3	0.3	0.6
p0291	9	0.0	6.9	0.5	0.4	0.7
p0548	62	0.2	48.8	3.3	6.3	3.8
pipex	0	0.0	0.3	0.0	0.0	0.0
rgn	14	0.1	7.2	0.1	5.1	1.5
sample2	0	0.0	0.3	0.0	0.0	0.0
sentoy	0	0.0	0.4	0.0	0.0	0.0
stein15	0	0.0	0.1	0.0	0.0	0.0
stein27	0	0.0	0.2	0.0	0.0	0.0
stein45	1	0.0	0.7	0.0	0.0	0.0
stein9	0	0.0	0.1	0.0	0.0	0.0
vpm1	17	0.1	13.0	1.9	1.0	1.1

Table A.3: Statistics for MIPLIB 2.0 instances.

Instance	n	$\max_{P_I}^{\text{bit}}$	$k_{P_I}^{\text{cach}}$	$k_{P_I}^{\text{heur}}$	$k_{P_I}^{\text{orac}}$	$k_{P_I}^{\text{apx}}$	$k_{P_I}^{\text{exct}}$	$k_{P_I}^{\text{fact}}$
air01	771	9,859	750	876	134	285,672	770	618
bell3b	133	145,456	133	123	19	8,741	133	116
bell5	104	47,613	104	99	8	5,537	104	98
bm23	27	221	27	16	0	379	27	28
cracpb1	572	103,921	484	435	7	159,414	553	479
dcmulti	548	2,790,396	470	251	4	147,106	547	468
diamond	2	2	1	1	1	3	1	0
egout	141	7,029	68	120	28	5,103	129	42
enigma	100	2	79	155	77	398	83	4
flugpl	18	14	12	23	4	127	18	10
gen	870	49,509	720	962	181	324,271	869	541
lseu	89	4,393	89	77	0	4,006	89	90
misc01	83	241	68	88	25	2,707	82	45
misc02	59	97	47	62	11	1,518	55	38
misc03	160	8,876	136	128	21	11,891	159	117
misc05	136	26,439	108	67	11	8,576	134	99
misc07	260	40,670	228	197	25	32,335	257	205
mod008	319	2	319	638	0	51,041	319	320
mod013	96	936	83	129	0	4,566	91	84
p0033	33	30	33	51	7	547	33	28
p0040	40	111	40	48	11	776	40	31
p0201	201	27,841	201	253	63	18,411	201	140
p0282	282	86	282	318	0	39,904	282	283
p0291	291	78	291	332	0	42,487	291	292
p0548	548							
pipex	48	526	32	17	2	1,024	47	32
rgn	180	86,119	160	158	0	16,081	179	161
sample2	67	164	44	58	13	1,684	66	33
sentoy	60	2	60	60	0	1,831	60	61
stein15	15	2	15	30	0	121	15	16
stein27	27	2	27	54	0	379	27	28
stein45	45	2	45	90	0	1,036	45	46
stein9	9	2	9	18	0	46	9	10
vpm1	378	6,172	336	547	49	67,537	376	289

Table A.4: Timings for MIPLIB 2.0 instances.

Instance	t_{P_I}	$t_{P_I}^{\text{cach}}$	$t_{P_I}^{\text{heur}}$	$t_{P_I}^{\text{orac}}$	$t_{P_I}^{\text{apx}}$	$t_{P_I}^{\text{exct}}$	$t_{P_I}^{\text{fact}}$
air01	50	0.0	21.2	25.0	1.6	1.5	0.1
bell3b	5	0.0	2.2	1.7	0.1	0.7	0.2
bell5	3	0.0	2.1	0.7	0.0	0.1	0.0
bm23	2	0.0	1.7	0.0	0.0	0.0	0.0
cracpb1	122	0.0	117.1	1.4	1.1	1.6	0.7
dcmulti	1,421	1.3	64.3	0.7	9.7	945.9	398.5
diamond	0	0.0	0.0	0.0	0.0	0.0	0.0
egout	3	0.0	1.8	0.6	0.0	0.0	0.0
enigma	110	0.0	15.5	94.4	0.0	0.0	0.0
flugpl	3	0.0	0.2	3.1	0.0	0.0	0.0
gen	621	1.6	107.4	476.6	25.1	2.8	6.1
lseu	1	0.0	1.1	0.0	0.0	0.0	0.0
misc01	10	0.0	7.2	2.5	0.0	0.0	0.0
misc02	1	0.0	1.2	0.2	0.0	0.0	0.0
misc03	33	0.0	24.6	8.5	0.0	0.2	0.0
misc05	32	0.0	28.9	3.0	0.1	0.2	0.1
misc07	141	0.0	103.2	36.2	0.2	1.3	0.5
mod008	19	0.0	17.0	0.0	0.3	0.2	1.4
mod013	1	0.0	0.8	0.0	0.0	0.0	0.0
p0033	1	0.0	0.8	0.3	0.0	0.0	0.0
p0040	1	0.0	0.4	0.1	0.0	0.0	0.0
p0201	21	0.0	11.6	8.6	0.1	0.3	0.1
p0282	6	0.0	4.9	0.0	0.3	0.2	0.4
p0291	7	0.0	5.7	0.0	0.3	0.2	0.5
p0548							
pipex	1	0.0	0.9	0.5	0.0	0.0	0.0
rgn	4	0.0	2.8	0.0	0.1	0.6	0.2
sample2	1	0.0	0.6	0.4	0.0	0.0	0.0
sentoy	0	0.0	0.3	0.0	0.0	0.0	0.0
stein15	0	0.0	0.1	0.0	0.0	0.0	0.0
stein27	0	0.0	0.3	0.0	0.0	0.0	0.0
stein45	1	0.0	0.7	0.0	0.0	0.0	0.0
stein9	0	0.0	0.1	0.0	0.0	0.0	0.0
vpm1	17	0.1	12.7	1.9	0.9	0.3	1.0

Table A.5: Statistics for Relaxations of presolved MIPLIB 2.0 instances.

Instance	n	$\max_Q^{\text{bit}}$	k_Q^{cach}	k_Q^{heur}	k_Q^{apx}	k_Q^{exct}	k_Q^{fact}
air01	760	40,156	363	364	210,178	758	364
bell3b	113	375,315	91	101	6,189	113	92
bell5	87	1,399	56	71	3,333	88	57
bm23	27	40,453	27	29	379	27	28
cracpb1	518	2,559,840	478	479	133,602	506	479
dcmulti	548	4,508,942	469	489	147,267	547	470
diamond	2	0	0	1	3	0	0
egout	118	1,161	41	53	4,019	116	42
enigma	100	192,392	79	79	4,820	94	80
flugpl	16	936	10	11	116	14	11
gen	699	6,148	509	674	203,323	699	412
lseu	89	61,305	85	88	3,996	89	86
misc01	82	23,813	56	57	3,053	82	57
misc02	58	553	37	44	1,481	55	38
misc03	159	78,142	115	116	11,731	159	116
misc05	128	273,917	100	99	7,851	126	101
misc07	259	225,017	201	218	31,960	254	202
mod008	319	2	319	637	51,041	319	320
mod013	96	1,277	83	95	4,566	93	84
p0033	29	4,879	26	32	430	29	27
p0040	40	14	20	21	611	39	21
p0201	201	82,948	163	202	17,527	201	128
p0282	281	58	200	241	36,301	281	201
p0291	264	2	67	96	15,478	264	68
p0548	527	2,704,204	362	381	125,434	527	363
pipex	48	25,695	32	33	1,041	47	33
rgn	175	208,257	160	178	15,281	174	161
sample2	55	945	32	37	1,265	52	33
sentoy	60	94	60	60	1,831	60	61
stein15	15	2	15	30	121	15	16
stein27	27	2	27	54	379	27	28
stein45	45	2	45	90	1,036	45	46
stein9	9	2	9	18	46	9	10
vpm1	362	47,101	168	197	46,789	361	169

Table A.6: Timings for relaxations of presolved MIPLIB 2.0 instances.

Instance	t_Q	t_Q^{cach}	t_Q^{heur}	t_Q^{apx}	t_Q^{exct}	t_Q^{fact}
air01	16	0.0	13.1	0.9	1.1	0.1
bell3b	9	0.0	5.3	0.0	2.9	1.2
bell5	1	0.0	0.9	0.0	0.0	0.0
bm23	1	0.0	1.3	0.0	0.1	0.0
cracpb1	1,346	0.3	210.1	5.3	818.3	312.3
dcmulti	3,162	2.2	272.2	13.6	2,057.8	815.6
diamond	0	0.0	0.0	0.0	0.0	0.0
egout	1	0.0	0.8	0.0	0.0	0.0
enigma	3	0.0	2.6	0.0	0.6	0.2
flugpl	0	0.0	0.1	0.0	0.0	0.0
gen	60	0.1	47.2	9.2	1.1	2.3
lseu	2	0.0	1.4	0.0	0.1	0.1
misc01	2	0.0	1.8	0.0	0.1	0.1
misc02	0	0.0	0.3	0.0	0.0	0.0
misc03	9	0.0	6.7	0.1	1.6	0.6
misc05	22	0.0	15.5	0.0	4.8	1.9
misc07	44	0.0	23.8	0.4	14.2	5.4
mod008	20	0.1	17.4	1.3	0.4	1.0
mod013	1	0.0	0.7	0.0	0.0	0.0
p0033	0	0.0	0.3	0.0	0.0	0.0
p0040	0	0.0	0.1	0.0	0.0	0.0
p0201	7	0.0	6.0	0.2	1.0	0.3
p0282	5	0.0	4.3	0.3	0.1	0.2
p0291	1	0.0	1.1	0.0	0.1	0.0
p0548	64	0.3	35.3	1.5	20.6	5.7
pipex	1	0.0	0.5	0.0	0.0	0.0
rgn	8	0.0	5.1	0.1	1.8	0.5
sample2	0	0.0	0.2	0.0	0.0	0.0
sentoy	0	0.0	0.4	0.0	0.0	0.0
stein15	0	0.0	0.1	0.0	0.0	0.0
stein27	0	0.0	0.3	0.0	0.0	0.0
stein45	1	0.0	0.8	0.0	0.0	0.0
stein9	0	0.0	0.1	0.0	0.0	0.0
vpm1	6	0.0	4.8	0.5	0.4	0.2

Table A.7: Statistics for presolved MIPLIB 2.0 instances.

Instance	n	$\max_{Q_I}^{\text{bit}}$	$k_{Q_I}^{\text{cach}}$	$k_{Q_I}^{\text{heur}}$	$k_{Q_I}^{\text{orac}}$	$k_{Q_I}^{\text{apx}}$	$k_{Q_I}^{\text{exct}}$	$k_{Q_I}^{\text{fact}}$
air01	760	2,970	363	361	3	209,381	759	362
bell3b	113	28,918	91	67	6	6,064	113	87
bell5	87	506	56	71	0	3,333	87	57
bm23	27	221	27	16	0	379	27	28
cracpb1	518	132,385	478	432	0	133,602	511	479
dcmulti	548	2,152,153	469	133	3	147,106	548	468
diamond	2	0	0	1	1	3	0	0
egout	118	2,444	41	59	0	4,019	115	42
enigma	100	2	79	155	77	495	80	4
flugpl	16	22	10	15	4	92	15	8
gen	699	106,918	509	701	99	203,035	699	412
lseu	89	3,187	85	69	0	3,996	89	86
misc01	82	352	56	65	13	2,745	82	45
misc02	58	120	37	48	5	1,387	56	34
misc03	159	8,249	115	93	6	11,496	159	111
misc05	128	32,645	100	56	3	7,792	127	99
misc07	259	44,300	201	157	4	31,841	255	199
mod008	319	2	319	638	0	51,041	319	320
mod013	96	1,227	83	126	0	4,566	93	84
p0033	29	30	26	38	7	400	29	21
p0040	40	65	20	13	0	611	39	21
p0201	201	16,590	163	192	37	17,601	200	128
p0282	281	1,501	200	235	0	36,301	281	201
p0291	264	110	67	99	0	15,478	264	68
p0548	527	147,885	362	265	6	124,594	527	358
pipex	48	586	32	21	2	1,041	47	32
rgn	175	10,314	160	131	0	15,281	174	161
sample2	55	56	32	35	0	1,265	50	33
sentoy	60	2	60	60	0	1,831	60	61
stein15	15	2	15	30	0	121	15	16
stein27	27	2	27	54	0	379	27	28
stein45	45	2	45	90	0	1,036	45	46
stein9	9	2	9	18	0	46	9	10
vpm1	362	4,974	168	224	0	46,789	362	169

Table A.8: Timings for presolved MIPLIB 2.0 instances.

Instance	t_{Q_I}	$t_{Q_I}^{\text{cach}}$	$t_{Q_I}^{\text{heur}}$	$t_{Q_I}^{\text{orac}}$	$t_{Q_I}^{\text{apx}}$	$t_{Q_I}^{\text{exct}}$	$t_{Q_I}^{\text{fact}}$
air01	14	0.0	10.9	0.3	0.7	1.0	0.1
bell3b	2	0.0	1.4	0.2	0.0	0.1	0.1
bell5	1	0.0	0.8	0.0	0.0	0.0	0.0
bm23	2	0.0	1.7	0.0	0.0	0.0	0.0
cracpb1	108	0.1	105.3	0.0	0.8	1.4	0.5
dcmulti	369	0.6	14.0	0.3	6.1	276.4	71.3
diamond	0	0.0	0.0	0.0	0.0	0.0	0.0
egout	1	0.0	0.9	0.0	0.0	0.0	0.0
enigma	153	0.0	17.5	135.6	0.0	0.0	0.0
flugpl	2	0.0	0.2	1.6	0.0	0.0	0.0
gen	58	0.3	39.7	6.9	6.8	1.5	2.1
lseu	1	0.0	1.0	0.0	0.0	0.0	0.0
misc01	7	0.0	5.6	1.6	0.0	0.0	0.0
misc02	1	0.0	0.8	0.1	0.0	0.0	0.0
misc03	31	0.0	24.6	6.3	0.0	0.1	0.0
misc05	32	0.0	30.7	0.7	0.0	0.3	0.0
misc07	111	0.0	87.2	22.6	0.2	1.0	0.4
mod008	20	0.1	17.4	0.0	0.4	0.2	1.3
mod013	1	0.0	1.0	0.0	0.0	0.0	0.0
p0033	1	0.0	0.5	0.2	0.0	0.0	0.0
p0040	0	0.0	0.1	0.0	0.0	0.0	0.0
p0201	12	0.0	9.1	2.7	0.1	0.2	0.0
p0282	4	0.0	3.6	0.0	0.1	0.2	0.1
p0291	1	0.0	0.9	0.0	0.0	0.1	0.0
p0548	175	0.1	34.1	124.0	2.9	9.2	4.6
pipex	2	0.0	1.1	0.7	0.0	0.0	0.0
rgn	3	0.0	2.5	0.0	0.1	0.1	0.1
sample2	0	0.0	0.3	0.0	0.0	0.0	0.0
sentoy	0	0.0	0.4	0.0	0.0	0.0	0.0
stein15	0	0.0	0.1	0.0	0.0	0.0	0.0
stein27	0	0.0	0.3	0.0	0.0	0.0	0.0
stein45	1	0.0	0.8	0.0	0.0	0.0	0.0
stein9	0	0.0	0.1	0.0	0.0	0.0	0.0
vpm1	6	0.1	4.5	0.0	0.3	0.3	0.2