Representations of lattice point sets

Theory, Algorithms, Applications

Habilitationsschrift

zur Erlangung des akademischen Grades

doctor rerum naturalium habilitatus (Dr. rer. nat. habil.)

genehmigt

durch die Fakultät für Mathematik

der Otto-von-Guericke-Universität Magdeburg

von	**Dr. Raymond Hemmecke**	
geb. am	**16.07.1972**	in **Kölleda, Thüringen**

Gutachter:	Prof. Dr. Robert Weismantel	Universität Magdeburg
	Prof. Dr. Peter Kleinschmidt	Universität Passau
	Prof. Dr. Bernd Sturmfels	University of California, Berkeley

Magdeburg, den 4. Oktober 2006

Zusammenfassung

In der vorliegenden Arbeit beschäftigen wir uns mit Systemen linearer diophantischer Gleichungen und Ungleichungen der Form

$$\begin{array}{rcl} a_i^\intercal z & = & \alpha_i, \quad i = 1, \ldots, m_1, \\ b_j^\intercal z & \leq & \beta_j, \quad j = 1, \ldots, m_2, \\ c_k^\intercal z & \equiv & \gamma_k \pmod{p_k}, \quad k = 1, \ldots, m_3. \end{array}$$

Solche Systeme treten in vielen interessanten Anwendungen auf, z.B.:

- lineare und nichtlineare ganzzahlige Optimierung [2],
- Sampling auf Kontingenztabellen in der Statistik [50],
- Zählen von Gitterpunkten in der Kombinatorik und der Statistik [44],
- Dekomposition chemischer Reaktionen in Elementarreaktionen [89],
- Erreichbarkeit in Petri-Netzen [22],
- Fundamentalkurven und -flächen in der Topologie [124].

Abhängig vom Problem ergeben sich unterschiedliche mathematische Fragestellungen, z.B.:

- *Lösbarkeit* des linearen Systems über $\mathbb{Z}$,
- Finden *einer* ganzzahligen Lösung,
- Effiziente Kodierung *aller* ganzzahlige Lösungen,
- *Zählen* aller ganzzahligen Lösungen,
- Finden einer Kosten-*optimalen* ganzzahligen Lösung.

Dabei bildet die effiziente Kodierung aller, möglicherweise unendlich vielen, ganzzahligen Lösungen eines linearen Systems eines der grundlegenden Probleme. Wir betrachten im folgenden drei Möglichkeiten:

- Darstellung über ganzzahlige Basen,
- Darstellung über Hilbertbasen,
- Darstellung über kurze rationale Erzeugendenfunktionen.

Darstellungen von Gitterpunktmengen

Darstellung über ganzzahlige Basen. Wir nennen $T \subseteq S$ eine *ganzzahlige Erzeugendenmenge* von $S \subseteq \mathbb{Z}^n$, falls für alle $s \in S$ endliche viele Elemente $t_i \in T$ und nichtnegative ganze Zahlen α_i mit

$$s = \sum \alpha_i t_i$$

existieren. Eine inklusions-minimale ganzzahlige Erzeugendenmenge nennt man auch *ganzzahlige Basis*. Mit Hilfe einer ganzzahligen Basis lassen sich also alle Elemente einer, möglicherweise unendlichen, Gitterpunktmenge S ganzzahlig erzeugen. Obwohl diese Basen in einem primalen Algorithmus zur Lösung linearer ganzzahliger Optimierungsprobleme Einsatz finden [69], haben sie für viele andere Anwendungen zwei wichtige Nachteile. Zum einen existiert nicht für jede Gitterpunktmenge S eine *endliche* ganzzahlige Basis, selbst dann nicht, wenn S die Gitterpunktmenge eines Polyeders ist. Zum anderen werden durch T meist auch Elemente erzeugt, die nicht zu S gehören, d.h., im allgmeinen ist

$$S \subsetneq \left\{ \sum \alpha_i t_i : t_i \in T, \alpha_i \in \mathbb{Z}_+ \right\}.$$

Es sei noch bemerkt, daß für die Gitterpunktmenge in rationalen polyedrischen Kegeln stets eine endliche ganzzahlige Erzeugendenmenge, eine sogenannte *Hilbertbasis*, existiert.

Darstellung über Hilbertbasen. Die beiden Nachteile von ganzzahligen Basen lassen sich zumindest für Polyeder dadurch beheben, daß wir eine ganzzahlige Version von Weyls Theorem über die Zerlegung von Polyedern in die Summe eines Polytops und eines Kegels verwenden:

Giles & Pulleyblank [64]: *Für jedes nichtleere Polyeder $P \subseteq \mathbb{R}^n$ existiert ein rationales Polytop $Q \subseteq \mathbb{R}^n$ und ein rationaler Kegel $C \subseteq \mathbb{R}^n$ mit $P \cap \mathbb{Z}^n = (Q \cap \mathbb{Z}^n) + (C \cap \mathbb{Z}^n)$.*

Mit anderen Worten, jeder Gitterpunkt aus P ist die Summe einer *inhomogenen* ganzzahligen Lösung aus Q und einer nichtnegativen ganzzahligen Linearkombination der *homogenen* Elemente einer Hilbertbasis von C.

Die Vorteile dieser endlichen Darstellung liegen auf der Hand: Für jedes nichtleere Polyeder existiert eine solche Darstellung seiner Gitterpunkte und diese Darstellung kodiert wirklich nur die Gitterpunkte des Polyeders. Als Nachteil erweist sich jedoch die Größe der beteiligten Mengen. Sowohl $Q \cap \mathbb{Z}^n$ als auch die Hilbertbasis von C können selbst in fester Dimension exponentiell viele Elemente besitzen.

Darstellung über kurze rationale Erzeugendenfunktionen. Eine weitere Kodierungsmöglichkeit ist die implizite Kodierung der Gitterpunkte einer Menge S in einer Erzeugendenfunktion:

$$f_S(z) = \sum_{\alpha \in S} z^\alpha,$$

wobei $z^\alpha = z_1^{\alpha_1} \ldots z_n^{\alpha_n}$. Wenn S unendlich ist, so ist $f_S(z)$ eine (Laurent-) Reihe und für endliches S ist $f_S(z)$ ein (Laurent-) Polynom. Desweiteren hat $f_S(z)$ die schöne Eigenschaft, daß $|S| = f_S(\mathbf{1})$. Allerdings besitzt $f_S(z)$ auch $|S|$ Summanden. Interessanterweise kann man aber im Fall, daß S die Gitterpunktmenge eines Polyeders ist, $f_S(z)$ auch äquivalent als Summe rationaler Funktionen

$$f_S(z) = \sum_{i\in I} E_i \frac{z^{u_i}}{\prod_{j=1}^{d}(1 - z^{v_{ij}})}$$

mit endlicher Indexmenge I und mit $E_i \in \{0, 1\}$ schreiben, wobei die Kodierungslänge dieser Repräsentation in fester Dimension polynomiell in der Kodierungslänge der Ungleichungsbeschreibung für P ist [13]. Dies rechtfertigt die Bezeichnung "*kurze* rationale Erzeugendenfunktion". Wie schon oben angedeutet, ermöglicht diese Darstellung das effiziente Zählen von Gitterpunkten in Polytopen fester Dimension.

Obwohl diese Darstellung in fester Dimension schöne Komplexitätsaussagen erlaubt, hat auch sie einen Nachteil. Ist S lediglich über $f_S(z)$, d.h. als rationale Erzeugendenfunktion gegeben, erweist sich bereits die Angabe eines einzigen Punktes aus S als *praktisch* schwierig, obwohl sich dies in fester Dimension *theoretisch* natürlich in polynomieller Zeit bewerkstelligen läßt.

Themen der vorliegenden Arbeit

Ganzzahlige Basen, Hilbertbasen, Graverbasen

In Kapitel 1 zeigen wir, daß eine Gitterpunktmenge S genau dann eine endliche ganzzahlige Basis besitzt, wenn cone(S) ein rationaler polyedrischer Kegel ist.

In Kapitel 2 betrachten wir spezielle ganzzahlige Basen, nämlich Hilbertbasen und Graverbasen. Wir setzen die Ausführungen aus [71] über die positive Summeneigenschaft fort, kombinieren sie aber mit einem Project-and-Lift Ansatz, der zuerst Hilbert- und Graverbasen in Projektionen berechnet und diese dann in den Originalraum liftet. Dieser Algorithmus zur Berechnung von Hilbertbasen und Graverbasen scheint das ganzzahlige Pendant zur Double-Description Methode zur Berechnung von Extremstrahlen von Kegeln und von Zirkuiten von Matrizen [60, 74] zu sein.

Kapitel 3 enthält keine eigenen neuen Resultate, sondern erklärt Schritt für Schritt, wie Graverbasen sich als universelle Testmengen in einem primalen Augmentierungsalgorithmus zur Lösung von linearen ganzzahligen Optimierungsproblemen einsetzen lassen. Hierbei liefert die Graverbasis Richtungen, in die nichtoptimale ganzzahlige Lösungen verbessert werden können. Schulz und Weismantel [108] haben gezeigt, daß man sogar die verbessernden Richtungen so geschickt wählen kann, daß der Augmentierungsalgorithmus in polynomieller Zeit in der Kodierungsgröße des Problems und der Kodierungsgröße der Graverbasis läuft.

In Kapitel 4 wenden wir diese Komplexitätsaussage auf lineare ganzzahlige Optimierungsprobleme an, deren Struktur eine polynomiell große Kodierung der Graverbasis zuläßt und damit zu einem polynomiellen Algorithmus zur Lösung dieser Problemklasse führt.

In Kapitel 5 präsentieren wir dann einen Algorithmus zur effizienteren Berechnung von Graverbasen im Fall, daß der gegebenen Matrix eine gewisse *symmetrische* Struktur zugrundeliegt. Dies ermöglicht zum Beispiel die Berechnung von Graverbasen für größere Kontingenztabellen in der Statistik.

In Kapitel 6 zeigen wir die Existenz *endlicher* Testmengen für gewisse ganzzahlige konvexe Optimierungsprobleme über Polyedern. Unser Ansatz über Graverbasen führt zur gleichen endlichen Testmenge wie in [97] durch Verfeinerung von Kegeln und Vereinigung ihrer inklusions-minimalen Hilbertbasen.

Am Ende des ersten Teil, in Kapitel 7, präsentieren wir Malkins Project-and-Lift Algorithmus zur Berechnung von Erzeugendenmengen und Gröbnerbasen von Gitteridealen [75]. Diese Mengen finden zum Beispiel beim Sampling in der Statistik, als Testmengen in der ganzzahligen Optimierung (für feste Kostfunktion) Kapitel 3 oder beim Zählen von Gitterpunkten in Polytopen (Kapitel 8) Anwendung. Interessanterweise ist Malkins Algorithmus angewandt auf die Berechnung von Graverbasen mit Hilfe von Gröbnerbasen äquivalent zum Project-and-Lift Algorithmus aus Kapitel 2 zur direkten Berechnung von Graverbasen, [114, Kapitel 14].

Kurze rationale Erzeugendenfunktionen

Nachdem wir in Kapitel 8 zeigen, wie man Gitterpunkte in Polytopen mit Hilfe von Hilbertbasen und Gröbnerbasen torischer Ideale berechnen kann, gehen wir in Kapitel 9 auf die einzelnen Schritte von Barvinoks Algorithmus [13] und dessen Implementierung in der Software `LattE` [41] ein. Obwohl diese Implementierung bereits recht erfolgreich neue Zählformeln berechnen konnte, konnte erst unsere vorgeschlagene Homogenisierung des Polytops mit anschließender Verwendung von Barvinoks Algorithmus die Zählformel für magische 5×5 Quadrate bestimmen.

In Kapitel 10 nutzen wir Barvinoks rationale Erzeugendenfunktionen zur Maximierung einer linearen Funktionen über den Gitterpunkten eines Polytops. Schon Barvinok merkte an, daß ein in fester Dimension polynomieller Algorithmus zum Gitterpunktzählen mit Hilfe binärer Suche auch einen polynomiellen Optimierungsalgorithmus in fester Dimension impliziert. Lasserre gab einen weiteren heuristischen Ansatz an, den wir in Kapitel 10 zu einem deterministischen Algorithmus, dem Digging-Algorithmus, ausbauen. Beide Algorithmen wurden implementiert und verglichen, wobei der Digging-Algorithmus in unseren Beispielen besser abschnitt.

In den Kapiteln 11 und 12 konstruieren wir für feste Dimension einen FPTAS (vollpolynomielles Approximationsschema) für die ganzzahlige und gemischt-ganzzahlige Optimierung eines nichtnegativen Polynoms über einem Polytop. Dies liefert die bestmögliche zu erhoffende Komplexitätsklasse, da beide Probleme selbst in fester Dimension NP-schwer sind.

In 2003 zeigten Barvinok und Woods [15], daß man mit kurzen rationalen Erzeugendenfunktionen auch effizient z.B. Durchschnitte oder Projektionen von Gitterpunktmengen berechnen kann. Insbesondere zeigten sie, daß sich Hilbertbasen und Graverbasen in kurze rationale Erzeugendenfunktionen kodieren lassen. In Kapitel 13 nutzen wir Barvinoks und Woods' Resultate, um auch Gröbnerbasen torischer Ideale effizient zu kodieren.

Acknowledgments

There are many people that I wish to thank. Without them, this thesis would not exist or would not be half as exciting.

Over the last 5 years, I have had the privilege and the pleasure to work in the research groups of David Woodruff and Jesus DeLoera at the University of California in Davis, and of Robert Weismantel at the Otto-von-Guericke University of Magdeburg. My greatest thanks go to David, Jesus, and Robert, who all three provided me with perfect working conditions, who supported me not only financially but encouraged research in a variety of different exciting fields such as Combinatorics, Computational Geometry, Statistics or (Stochastic) Integer Programming, who gave me constant support and advice whenever needed, and who were a continuous source for many inspiring and fruitful discussions leading to a whole series of exciting results and research articles.

It was great fun to collaborate with my many co-authors on quite a few interesting research projects. They did not only introduce me to great mathematical problems and ideas, but also shared with me their knowledge and experience on a variety of mathematical topics. My dearest thanks go to Maya Ahmed, Matthias Aschenbrenner, Jesus DeLoera, David Haws, Harald Held, Peter Huggins, Matthias Köppe, Peter Malkin, Shmuel Onn, Rüdiger Schultz, Bernd Sturmfels, Jeremy Tauzer, Robert Weismantel, David Woodruff, and Ruriko Yoshida.

I wish to thank Peter Malkin and Matthias Walter for their great contributions to `4ti2`. I am especially grateful to Peter for letting me include his project-and-lift algorithm to compute generating sets and Gröbner bases of lattices ideals [75] in Chapter 7.

In this respect, I also want to thank all users of `4ti2` and of `LattE` for applying these codes to many hard problems, for demonstrating their practical usefulness and for posing new challenges that finally led and will further lead to improvements of algorithms and implementations.

I would like to thank Matthias Köppe, Matthias Walter, and Ruriko Yoshida for letting me use some of their very nice pictures.

And finally, I wish to thank my wife Susi for being the shining star in my life, for keeping me smiling happily each and every day.

Contents

Notation

$\mathbb{Z}$	integer numbers
$\mathbb{Q}$	rational numbers
$\mathbb{R}$	real numbers
$\mathbb{C}$	complex numbers
$\mathbb{X}_+$	$\{x \in \mathbb{X} : x \geq 0\}$
$\mathbb{X}_{>0}$	$\{x \in \mathbb{X} : x > 0\}$
$\ker_{\mathbb{X}}(A)$	$\{x \in \mathbb{X} : Ax = 0\}$
$(\mathrm{IP})_{c,b}$	$\min\{c^\intercal z : Az = b,\ z \in \mathbb{Z}_+\}$
$(\mathcal{L})_{c,\beta}$	$\min\{c^\intercal z : z \equiv \beta \pmod{\mathcal{L}}, z \in \mathbb{Z}_+^n\}$
$v^{(i)}$	i^{th} component of v
$\|v\|_1$	l_1 – norm $\sum_{i=1}^d \|v^{(i)}\|$
$\mathrm{supp}(v)$	support $\{i : v^{(i)} \neq 0\}$
$(v^+)^{(i)}$	$\max(v^{(i)}, 0)$
$(v^-)^{(i)}$	$\max(-v^{(i)}, 0)$
x^v	$x_1^{v^{(1)}} \cdot \ldots \cdot x_d^{v^{(d)}}$
$e_1, \ldots, e_d$	unit vectors in $\mathbb{R}^d$
$v \leq w$, where $v, w \in \mathbb{R}_+^d$	$v^{(i)} \leq w^{(i)}$ for $i = 1, \ldots, d$
$v \sqsubseteq w$, where $v, w \in \mathbb{R}^d$	$(v^+, v^-) \leq (w^+, w^-)$
$x_I \geq 0$	$x_i \geq 0$ for all $i \in I$

Introduction

In this thesis, we are concerned with systems of linear diophantine equations and inequalities of the form

$$\begin{array}{rcll} a_i^\intercal z & = & \alpha_i, & i = 1, \ldots, m_1, \\ b_j^\intercal z & \leq & \beta_j, & j = 1, \ldots, m_2, \\ c_k^\intercal z & \equiv & \gamma_k \pmod{p_k}, & k = 1, \ldots, m_3. \end{array}$$

Since we can turn any inequality into an equation by introducing a non-negative integer slack variable, and since we can also turn any modulo constraint into an equation by introducing an integer slack variable that is unconstraint in sign, we may assume that the above system is given in the form $Az = b, z_I \geq 0$ for some index set $I \subseteq \{1, \ldots, n\}$.

Such systems appear in many interesting applications, such as

- linear and nonlinear integer programming [2],
- sampling in statistics [50],
- counting lattice points in combinatorics [44],
- decomposition of chemical reactions into elementary reactions [89],
- reachability in Petri nets [22],
- fundamental curves and surfaces in topology [124].

Depending on the application, we are faced with different mathematical questions, for example:

- *feasibility* of the above linear system over $\mathbb{Z}$,
- finding *one* integer solution,
- efficient encoding of *all* integer solutions,
- *counting* all integer solutions,
- finding a cost-*optimal* integer solution.

Among these questions, the efficient encoding of all, potentially infinitely many, integer solutions to a linear system of the above type is of fundamental importance. In this thesis, we consider the following three possibilities:

- representation via integral bases,
- representation via Hilbert bases,
- representation via short rational generating functions.

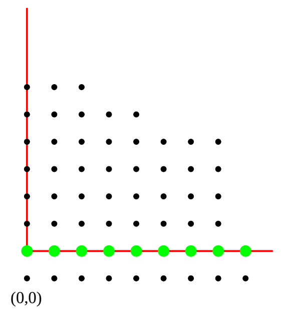
Representations of lattice point sets

Representation via integral bases. We call $T \subseteq S$ an *integral generating set* of $S \subseteq \mathbb{Z}^n$, if for all $s \in S$ there exist finitely many elements $t_i \in T$ and non-negative integers α_i with

$$s = \sum \alpha_i t_i.$$

An inclusion-minimal integral generating set is also called an *integral basis*. Once an integral generating set T of S is available, every lattice point in S can be written as a finite non-negative integer linear representation of elements from T, even if S is not finite. However, there are two drawbacks. First, there might be many more lattice points outside of S that can be represented as such a linear combination, and second, not every set $S \subseteq \mathbb{Z}^n$ possesses a *finite* integral generating set.

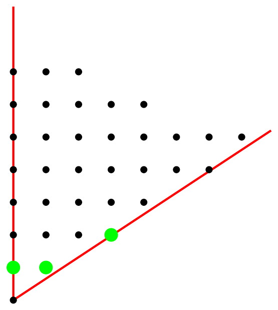
Example. The set $S = \{(z_1, 1) \in \mathbb{Z}^2 : z_1 \in \mathbb{Z}_+\}$ does not have a finite integral generating set.



In fact, S itself is the smallest integral generating set of S. This example also demonstrates that $\{\sum \alpha_i t_i : t_i \in T, \alpha_i \in \mathbb{Z}_+\} = \{z \in \mathbb{Z}^2_+ : z_2 \geq 1\}$ can be much bigger than S. □

Despite these negative properties, integral generating sets find application in the design of integer simplex type methods based on reformulation techniques to solve linear integer optimization problems, see for example the *Integral Basis Algorithm* in [69]. Herein, the authors exploit the fact that a set S of the form $S = \{z \in \mathbb{Z}^n : Az \leq b, z \geq 0\}$ always has a finite integral generating set. Moreover, as a side-effect of their approach, unwanted additional solutions from $\{\sum \alpha_i t_i : t_i \in T, \alpha_i \in \mathbb{Z}_+\}$ are excluded automatically.

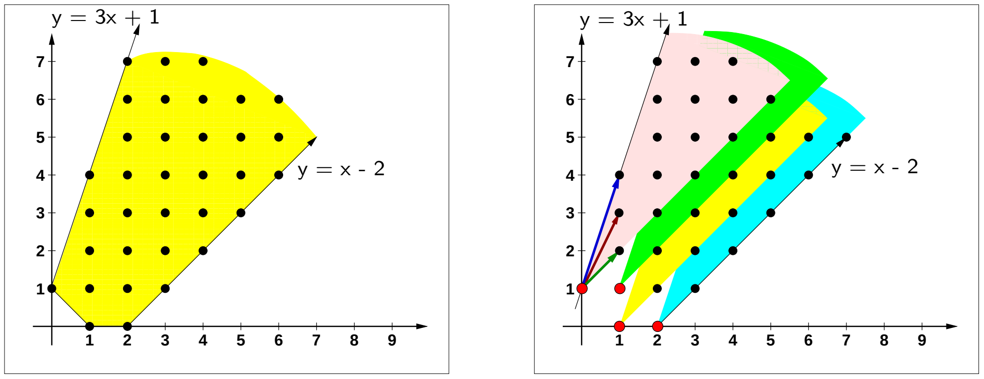
In [19], Bertsimas and Weismantel characterize those polyhedra $P \subseteq \mathbb{R}^n_+$ whose sets of lattice points $S = P \cap \mathbb{Z}^n$ possess a finite integral generating set. In Chapter 1, we extend this criterion to arbitrary sets $S \subseteq \mathbb{Z}^n$. It turns out that S has a finite integral generating set if and only if cone(S) is a rational polyhedral cone. The latter includes the well-known fact that the lattice points in a rational polyhedral cone always possess a finite integral generating set, called a *Hilbert basis* of the cone. Moreover, it implies that a monoid $M \subseteq \mathbb{Z}^n$ is finitely generated if and only if cone(M) is a rational polyhedral cone, see Jeroslow [84].



Representation via Hilbert bases. Both disadvantages of integral bases can be avoided for polyhedra by using an integer version of Weyl's theorem [64]: For every non-empty rational polyhedron $P \subseteq \mathbb{R}^n$ there exist a rational polytope Q and a rational cone C such that

$$P \cap \mathbb{Z}^n = (Q \cap \mathbb{Z}^n) + (C \cap \mathbb{Z}^n).$$

In other words, if we set $I = Q \cap \mathbb{Z}^n$ and denote by H a finite Hilbert basis of C, then *any* integer point in P can be written as the sum of *one* of the finitely many (inhomogeneous) solutions in I and a *non-negative integer linear combination* of the finitely many (homogeneous) solutions in H.



$\{(x,y) \in \mathbb{Z}^2 : x - y \leq 2, -3x + y \leq 1, x + y \geq 1, y \geq 0\} = \{(0,1),(1,0),(1,1),(3,0)\} + \text{cone}((1,1),(1,3)) \cap \mathbb{Z}^2$

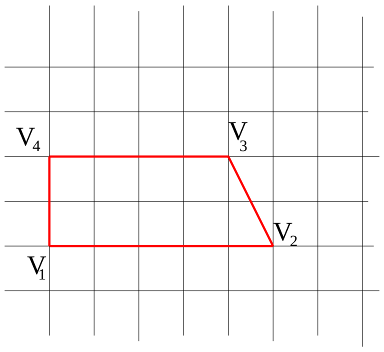
The advantages of this representation are immediate: There exists such a nice representation for *every* nonempty polyhedron and it encodes *exactly* the lattice points in the polyhedron. A disadvantage, however, are the sizes of the sets I and H. Even in fixed dimension, both sets can be of exponential size in the encoding length of the inequalities defining the polyhedron.

Representation via short rational generating functions. Another possible representation is the implicit encoding of the lattice points of a set S in a generating function

$$f_S(z) = \sum_{\alpha \in S} z^\alpha,$$

where $z^\alpha = z_1^{\alpha_1} \dots z_n^{\alpha_n}$. For infinite S, this generating function is a Laurent series. If S is finite, $f_S(z)$ is a (Laurent-) polynomial and it holds that $|S| = f_S(\mathbf{1})$. However, $f_S(z)$ contains also $|S|$ many summands.

$$V_1 = (0,0), V_2 = (5,0), V_3 = (4,2), V_4 = (0,2)$$



$$f_S(z) = z_1^5 + z_1^4z_2^2 + z_1^4z_2 + z_1^4 + z_1^3z_2^2 + z_1^3z_2 + z_1^3 + z_1^2z_2^2 + z_1^2z_2 + z_1^2 + z_1z_2^2 + z_1z_2 + z_1 + z_2^2 + z_2 + 1.$$

If S is the set of lattice points in a polyhedron, $f_S(z)$ can be written in a shorter form, namely as the sum of rational functions

$$f_S(z) = \sum_{i \in I} E_i \frac{z^{u_i}}{\prod_{j=1}^d (1 - z^{v_{ij}})},$$

where I is a finite index set and where $E_i \in \{0,1\}$. In fact, Barvinok [13] showed that in fixed dimension, one can even find such a representation that is of polynomial size in the encoding length of the inequality description of the polyhedron. This justifies the notion "*short* rational generating function". One possible application for these generating functions is the efficient counting of lattice points in polytopes in fixed dimension.

Although this representation allows us to show various nice complexity results, it also has a disadvantage. If S is given only encoded as a rational generating function, it is a hard task (in practice) to find a single element in S, although it is (in theory) a polynomially solvable problem in fixed dimension.

Topics and structure of this thesis

This theses presents the results from several single-authored [72, 73, 76] and co-authored [3, 40, 41, 42, 43, 44, 75] research articles.

Integral bases, Hilbert bases, Graver bases

In Chapter 1, we show that a lattice point set $S \subseteq \mathbb{Z}^n$ possesses a finite integral generating set if and only if $\operatorname{cone}(S)$ is a rational polyhedral cone. This result generalizes the characterizations by Bertsimas and Weismantel [19] of finite integral bases of polyhedra and by Jeroslow [84] of finitely generated monoids.

In Chapter 2, we deal with special integral bases, namely with (inclusion-minimal) Hilbert bases and Graver bases. Note that the Graver basis of a matrix A is the union

$$\bigcup_j \mathcal{H}_j \setminus \{0\}$$

of the inclusion-minimal Hilbert bases $\mathcal{H}_j$ of the pointed rational polyhedral cones

$$\ker(A) \cap \mathbb{O}_j = \{x \in \mathbb{O}_j : Ax = 0\}$$

taken over all 2^n orthants $\mathbb{O}_j$ of $\mathbb{R}^n$.

We base our treatment on the *positive sum property* [72] and combine it with a project-and-lift approach [73], which first computes Hilbert bases and Graver bases in projected spaces and then lifts them into the original space. This algorithm to compute Hilbert bases and Graver bases seems to be the integer equivalent to the double-description method to compute extreme rays of cones and circuits of matrices [60, 74].

One application of Graver bases is as *universal test sets* for the family of linear integer programming problems

$$\min\{c^\intercal z : Az = b, z \geq 0, z \in \mathbb{Z}^n\},$$

as $b \in \mathbb{Z}^d$ and $c \in \mathbb{R}^n$ vary. A universal test set for this family is a set of (integral) directions that contains an improving direction t for *any non-optimal* solution z_0 of *any* given problem from the problem family above. This means, that $z_0 - t$ is also feasible and $c^\intercal(z_0 - t) < c^\intercal z_0$. Repeating such augmentation steps, we obtain a simple primal optimization algorithm. In Chapter 3, which contains no new research results, we present all the necessary pieces of this augmentation algorithm. We wish to point out here, that Schulz and Weismantel [108] have shown that the augmenting steps can be chosen in such a way that only polynomially many augmentation steps are needed in order to reach an optimal solution. Thus, it is only the exponential size of the Graver basis that prevents the augmentation algorithm from running in polynomial time in the input data.

In Chapter 4, we consider a class of optimization problems with a very structured problem matrix

$$[A, B]^{(n)} := \begin{pmatrix} B & B & B & \cdots & B \\ A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A \end{pmatrix}.$$

If we fix the matrices A and B, then by combining a finiteness result by Hoşten and Sullivant [79] with the above mentioned polynomiality result by Schulz and Weismantel [108], we show

- that the size of the Graver basis of $[A,B]^{(n)}$ is polynomially bounded in n and in the size of the input data, and
- that consequently, the augmentation algorithm to solve this class of optimization problems runs in polynomial time in n and in the size of the input data.

This algorithm settles the complexity of optimizing a linear function over the lattice points of 3-dimensional $r \times s \times n$ transportation polytopes (also called *contingency tables* by statisticians) when two side lengths r and s are kept fix and the third side length n is allowed to vary. Three-dimensional transportation polytopes are especially appealing, since DeLoera and Onn [46] have shown recently that *every* polytope can be transformed into a polynomial-size three-dimensional transportation polytope, for which there exists a (coordinate-erasing) bijection between the lattice points of both polytopes.

We continue our treatment of Graver bases for special matrices in Chapter 5. Here, we present an algorithm that computes Graver bases of "symmetric matrices", or better of "symmetric lattices", more efficiently. We call a lattice *symmetric*, if there is a group of permutations that permute the components of a vector in $\mathbb{Z}^n$ in such a way that lattice points are mapped to lattice points. A matrix A is called symmetric, if the lattice $\ker(A)$ is symmetric. The algorithm we present is an adaptation of the project-and-lift algorithm from Chapter 2 and allows the computation of much bigger examples if the lattice is symmetric. The biggest algorithmic challenge here was the combination of *symmetry* with the *symmetry-breaking* project-and-lift algorithm algorithm from Chapter 2.

In the next chapter, Chapter 6, we generalize the notion of a test set to a certain class of convex integer optimization problems over the lattice points of a polyhedron, for which we show that a *finite* test set always exists. Our construction via linearization and Graver bases leads to the same set of vectors that was independently discovered by Murota, Saito, and Weismantel [97] via a refinement of cones and a union of their inclusion-minimal Hilbert bases.

Finally, in Chapter 7, we present Malkin's project-and-lift algorithm to compute generating sets and Gröbner bases of lattice ideals [75]. This algorithm completes the algorithmic framework started with the project-and-lift approach to compute Hilbert bases and Graver bases from Chapter 2. It turns out that Malkin's algorithm outperforms the saturation approach by Hoşten and Sturmfels [78] and its improvement by Bigatti, LaScala, and Robbiano [21]. Generating sets and Gröbner bases of lattice ideals have a variety of interesting applications such as for sampling in statistics or as test sets for integer linear programs, where the objective function is kept fix and only the right-hand side vector $b \in \mathbb{Z}^d$ is allowed to vary, see Chapter 3.

Gröbner bases of special lattice ideals, namely of toric ideals, also appear in one algorithm to count lattice points in polytopes via Hilbert bases, see Chapter 8. This counting algorithm is of course restricted by the sizes of the Hilbert basis and the Gröbner basis that has to be computed. As these sets might be exponentially large even in fixed dimension, this counting approach is applicable only for smaller examples. Nonetheless, we were able to recover known counting formulas and to establish new counting formulas such as for magic and semi-magic $3 \times 3 \times 3$ cubes.

Short rational generating functions

In the following chapters we concentrate on problems in fixed dimension. This leads to some nice complexity results. For example, in the 1980's, H. Lenstra created an algorithm to *detect* integer points in polyhedra, based on the LLL-algorithm and the idea of short vectors [67, 93]. As a consequence, solving the feasibility problem with a *fixed number of variables* can be done in time polynomial in the size of the input. In the 1990's, based on work by the geometers Brion, Khovanski, Lawrence, and Pukhlikov, Barvinok created an algorithm to *count* integer points inside polyhedra that runs in polynomial time for fixed dimension [13, 14]. Shortly after Barvinok's breakthrough, Dyer and Kannan [52] modified the original algorithm of Barvinok, which originally relied on Lenstra's result, giving a new proof that integer linear programming problems with a fixed number of variables can be solved in polynomial time.

In Chapter 9 we present each step of Barvinok's algorithm and give also details on how we implemented it into a software package we named `LattE` [41]. Although this implementation could already produce new counting formulas, only our algorithmic improvement via a homogenization of the polytope was able to compute the counting formula for magic 5×5 squares.

In Chapter 10, we compare two algorithms based on Barvinok's short rational generating functions for maximizing a linear function over the lattice points of a rational polytope. For the first algorithm, note that one can turn Barvinok's counting oracle into an algorithm that solves integer linear programming problems with a fixed number of variables in polynomial time (i.e. by counting the number of lattice points in P that satisfy $c^\intercal x \geq \alpha$, we can narrow the range for the maximum value of $c^\intercal x$, then we iteratively look for the largest α where the count is non-zero). This idea was already proposed by Barvinok in [14]. In the second algorithm we replace z_i by t^{c_i} in the generating function. Hence z^α becomes $t^{c^\intercal \alpha}$ and we obtain

$$f_S(z) = \sum_{\alpha \in S} z^\alpha \longrightarrow f_S(t) = \sum_{\alpha \in S} t^{c^\intercal \alpha}.$$

If we perform this substitution in the short rational function representation of $f_S(z)$, we are faced with the problem of recovering the highest term in the series expansion of $f_S(t)$. Recently, Lasserre gave a very easy heuristics to find this highest appearing term [91]. Lasserre suggests to expand every single summand, a rational function in t, to extract the highest term appearing in these expansions, and to add up the coefficients of this term. If the coefficient is non-zero, we have identified the highest appearing term in the expansion of $f_S(t)$ and have found the optimal value. Unfortunately, his heuristics often fails in practice and we need to "dig" further for the highest term with a *nonzero* coefficient. In Chapter 10, we present the details of this *digging algorithm* and compare it with Barvinok's binary search idea.

In Chapters 11 and 12, we construct fully polynomial time approximation schemes (FPTAS) for maximizing a non-negative polynomial over the (mixed-) integer points of a polytope based on short rational generating functions. For the pure integer case, in order to find lower and upper bounds for the optimal value f^*, we use the simple fact that for tuples of non-negative real numbers

$f(x_1), \ldots, f(x_s)$ and for all integers $k \geq 1$ we have

$$\sqrt[k]{\sum_{i=1}^{s} f(x_s)^k / s} \leq f^* = \max\{f(x_1), \ldots, f(x_s)\} \leq \sqrt[k]{\sum_{i=1}^{s} f(x_s)^k}.$$

Increasing k, these bounds become more and more tight. Then we employ a binary search to produce a lattice point whose objective value is bigger than $(1-\epsilon)f^*$ for given ϵ. This algorithm runs in time polynomial in the size of the input data and in $1/\epsilon$. In the mixed-integer situation, we construct a suitable sequence of grid refinements and use an FPTAS for the pure integer case on these grids in order to find a rational point whose objective value is bigger than $(1-\epsilon)f^*$.

In 2003, Barvinok and Woods [15] showed that one can perform set operations such as intersection, union, or even projection of certain lattice point sets in polynomial time when the dimension is fixed, using again short rational generating functions. They also showed that in fixed dimension, Hilbert bases and Graver bases can be encoded in polynomial time into a short rational generating function. In Chapter 13, using the results of Barvinok and Woods, we present an algorithm to construct a similar encoding also for Gröbner bases of toric ideals.

Part I

Integral bases, Hilbert bases, and Graver bases

Chapter 1

Integral bases

At the core of this chapter are integral generating sets of lattice point sets.

Definition 1.0.1 *Let $S \subseteq \mathbb{Z}^n$. Then we call $T \subseteq S$ an* **integral generating set** *of S, if for every $s \in S$ there exists a finite (integer) linear combination $s = \sum \alpha_i t_i$ with $t_i \in T$ and $\alpha_i \in \mathbb{Z}_+$. We call T an* **integral basis** *of S if T is an inclusion-minimal integral generating set.*

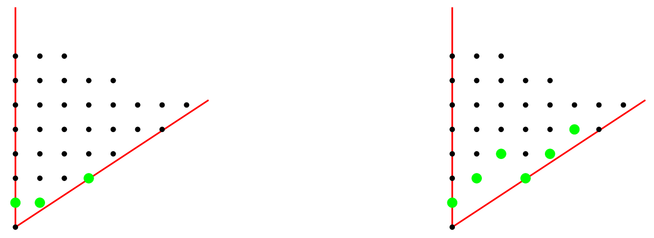


Figure 1.1: Minimal integral generating sets of two sets of lattice points.

Note that an integral generating set of S is allowed to contain elements only from S itself! The main result in this chapter is the following characterization of which sets $S \subseteq \mathbb{Z}^n$ admit a *finite* linear integer representation, i.e., every point of S can be expressed as a non-negative linear integer combination of a finite subset of S. For the proof of this theorem see Section 1.3.

Theorem 1.0.2 *Let $S \subseteq \mathbb{Z}^n$ be any set of lattice points in $\mathbb{Z}^n$.*

(a) *S has a finite integral generating set if and only if $C = \operatorname{cone}(S)$ is a rational polyhedral cone.*

(b) *If the cone $C = \operatorname{cone}(S)$ is rational and pointed, then there is a unique integral basis of S.*

Example 1.0.3 For $S = \left\{(x,y) \in \mathbb{Z}^2 : (x,y) = (s^2, s+t), s,t \in \mathbb{R}_+\right\} = \{(x,y) \in \mathbb{Z}^2_+ : x - y^2 \geq 0\}$, and for $S = \{(x,y) \in \mathbb{Z}^2_+ : y \geq 1\}$ we easily see that cone(S) is not rational, see Figure 1.2. Consequently, there do not exist finite integral generating sets for these sets S. $\square$

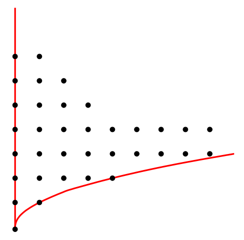
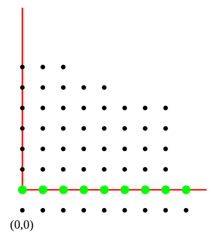



Figure 1.2: cone(S) is not rational.

Theorem 1.0.2 generalizes a characterization by Jeroslow of finitely generated monoids to arbitrary sets of lattice points.

Corollary 1.0.4 (Jeroslow [84]) *A monoid $M \subseteq \mathbb{Z}^n$ has a finite (integral) generating set if and only if $C = \operatorname{cone}(M)$ is a rational polyhedral cone.*

If S is the set of lattice points in a rational polyhedron, Bertsimas and Weismantel [19] already characterized those rational polyhedra that have a *finite* integral generating set. We give a proof of this non-trivial consequence of Theorem 1.0.2 in Section 1.4.

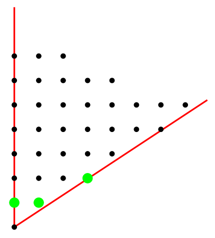
Corollary 1.0.5 (Bertsimas and Weismantel [19]) *For $A \in \mathbb{Z}^{d\times n}$ and $b \in \mathbb{Z}^d$, define the sets $P = \left\{x \in \mathbb{R}^n_+ : Ax \leq b\right\}$, $S = P \cap \mathbb{Z}^n$, and $C = \{x \in \mathbb{R}^n_+ : Ax \leq 0\}$.*

- **(a)** *There exists a finite integral generating set of S if and only if S contains all but finitely many integer points in $C \cap \mathbb{Z}^n_+$.*
- **(b)** *If a finite integral generating set of S exists, then there is a unique integral basis of S.*

As a special case of this corollary we may state a well-known fact about rational cones.

Corollary 1.0.6 *For every rational polyhedral cone $C = \{z \in \mathbb{R}^n : Bz \geq 0\}$, the set $C \cap \mathbb{Z}^n$ possesses a finite integral generating set, a so-called* Hilbert basis.

If C is pointed, there is a unique inclusion-minimal integral generating set.



This fundamental fact has applications in many scenarios for linear integer programming. In particular, it is important for

- proving finiteness results in cutting plane theory [106],
- showing the existence of totally dual integral systems of linear diophantine systems [64],
- deriving optimality conditions for linear integer optimization problems [65],
- designing integer simplex type methods based on reformulation techniques [69].

Note that Corollary 1.0.6 is also an immediate consequence of a more general statement implied by Theorem 1.0.2.

Corollary 1.0.7 *For every rational polyhedral cone $C = \{z \in \mathbb{R}^n : Bz \geq 0\}$ and for every sublattice Λ of $\mathbb{Z}^n$, the set $C \cap \Lambda$ forms a finitely generated monoid under vector addition.*

If C is pointed, there is a unique inclusion-minimal integral generating set.

Before proving Theorem 1.0.2 and the non-trivial Corollary 1.0.5, we present two major ingredients needed in the proofs. The first ingredient is the well-known Gordan-Dickson Lemma, whereas the second ingredient is a simple idea to prove Corollary 1.0.6. The combination of both enables us to show our main theorem of this chapter, Theorem 1.0.2.

1.1 Gordan-Dickson Lemma

The Gordan-Dickson Lemma (see for example Section 4.2 in [36]) is not only important for our proof of Theorem 1.0.2, but it will also be crucial for the termination proofs of the algorithms presented in later chapters. We state and prove here two equivalent versions of it. Whereas the *set version* is suitable to show finiteness of integral generating sets, the *sequence version* is suitable to show termination of many algorithms. In the statements the relation $\leq$ denotes the partial ordering on $\mathbb{Z}_+^n$, where $u \leq v$ if $u^{(i)} \leq v^{(i)}$ for $i = 1, \ldots, n$.

Lemma 1.1.1 (Gordan-Dickson Lemma, sequence version) *Let $\{p_1, p_2, \ldots\}$ be a sequence of points in $\mathbb{Z}_+^n$ such that $p_i \not\leq p_j$ whenever $i < j$. Then this sequence is finite.*

Proof. We prove this lemma inductively on the number n of variables. For $n = 1$, the sequence $\{p_1, p_2, \ldots\}$ is a strictly decreasing sequence of non-negative integers and is thus finite.

Let the lemma be true for $n = 1, \ldots, k$ and consider the case $n = k+1$. Assume that there exists an infinite sequence in $\mathbb{Z}_+^n$ such that $p_i \not\leq p_j$ whenever $i < j$. From this infinite sequence we construct a similar infinite sequence in $\mathbb{Z}_+^{n-1}$, a contradiction.

Consider the set of non-negative integer numbers $\{p_1^{(n)}, p_2^{(n)}, \ldots\}$ and let j_1 be the first index such that $p_{j_1}^{(n)}$ is minimal in this set. Now repeat the same step with the set of non-negative integer numbers $\{p_{j_1+1}^{(n)}, p_{j_1+2}^{(n)}, \ldots\}$ and find the first index j_2 such that $p_{j_2}^{(n)}$ is minimal in this set. Clearly, $p_{j_1}^{(n)} \leq p_{j_2}^{(n)}$. As the sequence $\{p_1, p_2, \ldots\}$ is not finite, we can repeat this step infinitely often, constructing a sequence $\{p_{j_1}, p_{j_2}, \ldots\}$ with the property that $p_{j_1}^{(n)} \leq p_{j_2}^{(n)} \leq \ldots$, that is, with last components that form an increasing sequence. Moreover, we still have for this infinite subsequence that $p_i \not\leq p_j$ whenever $i < j$. For $i = 1, 2, \ldots$, let $\pi_{n-1}(p_{j_i})$ denote the projection of p_{j_i} onto the first $n-1$ components. By construction, $p_i \not\leq p_j$ whenever $i < j$ implies that also $\pi_{n-1}(p_i) \not\leq \pi_{n-1}(p_j)$ whenever $i < j$. Therefore, $\{\pi_{n-1}(p_{j_1}), \pi_{n-1}(p_{j_2}), \ldots\}$ forms an infinite sequence in $\mathbb{Z}_+^{n-1}$ that contradicts our induction hypothesis for $n-1$. Thus, the sequence $\{p_1, p_2, \ldots\}$ must be finite. $\square$

Lemma 1.1.2 (Gordan-Dickson Lemma, set version) *Every infinite set $S \subseteq \mathbb{Z}_+^n$ contains only finitely many $\leq$-minimal points.*

Proof. Assume on the contrary that there were infinitely many such $\leq$-minimal points. They can be put into an infinite sequence $\{p_1, p_2, \ldots\}$ with $p_i \not\leq p_j$ for all i, j. By Lemma 1.1.1, however, this infinite sequence must be finite. A contradiction. $\square$

Consequently, Lemma 1.1.1 implies Lemma 1.1.2. Note that the converse also holds true.

Lemma 1.1.3 *Lemma 1.1.2 implies Lemma 1.1.1.*

Proof. Let $\{p_1, p_2, \ldots\}$ be a sequence of points in $\mathbb{Z}_+^n$ such that $p_i \not\leq p_j$ whenever $i < j$. Then the set $\{p_1, p_2, \ldots\} \subseteq \mathbb{Z}^n$ contains at most finitely many $\leq$-minimal points. Let k denote the maximal index among these $\leq$-minimal p_i. Thus, for $j > k$ we have $p_i \leq p_j$ for at least one ($\leq$-minimal) p_i, $i \leq k < j$. This contradicts the properties of the sequence $\{p_1, p_2, \ldots\}$, which is therefore finite. $\square$

Thus, both presented versions of the Gordan-Dickson Lemma are equivalent.

1.2 Hilbert bases

One notion that plays a fundamental role in the subsequent chapters is that of a *Hilbert basis* of a polyhedral cone. Corollary 1.0.6 states that rational polyhedral cones always possess a (by definition *finite*) Hilbert basis. Pointed rational polyhedral cones even contain a *unique* inclusion-minimal Hilbert basis. Let us give a simple proof of these facts.

Proof of Corollary 1.0.6. It suffices to prove the claim for full-dimensional simplicial cones C. If C is not full-dimensional, we may perform a linear integer transformation (that maps lattice points to lattice points) to make C full-dimensional in a space of smaller dimension. If C is not simplicial, we may triangulate it into simplicial cones. (Let us remind the reader that a *triangulation* of a cone C in dimension d is a collection of d-dimensional simplicial cones such that their union is C, their interiors are disjoint, and any pair of them intersect in a (possibly empty) common face.)

Let $v_1, \ldots, v_n \in \mathbb{Z}^n$ and consider the (half-open) parallelepiped

$$F = \left\{ \sum_{j=1}^{n} \alpha_j v_j : 0 \leq \alpha_1, \ldots, \alpha_n < 1 \right\}.$$

As $F \subseteq C$ is bounded, F contains only finitely many lattice points $\{f_1, \ldots, f_t\}$ in $\mathbb{Z}^n$. Moreover, every lattice point $v \in C \cap \mathbb{Z}^n$ can be written as

$$v = \sum_{j=1}^{n} \lambda_j v_j = \sum_{j=1}^{n} \lfloor \lambda_j \rfloor v_j + \left[\sum_{j=1}^{n} (\lambda_j - \lfloor \lambda_j \rfloor) v_j \right] \in \mathbb{Z}_+ v_1 + \ldots + \mathbb{Z}_+ v_n + (F \cap \mathbb{Z}^n)$$

with $\lambda_j \geq 0$ and $0 \leq \lambda_j - \lfloor \lambda_j \rfloor < 1$ for all $j = 1, \ldots, n$. Thus, the finitely many lattice points $(F \cap \mathbb{Z}^n) \cup \{v_1, \ldots, v_n\}$ form an integral generating sets of $C \cap \mathbb{Z}^n$. This shows the existence of a (finite) Hilbert basis of C.

Let us now show uniqueness of an inclusion-minimal Hilbert basis if C is pointed. As C is pointed, there is some vector $c \in \mathbb{R}^n$ such that $C \subseteq \{x \in \mathbb{R}^n : c^\intercal x \geq 0\}$ and $C \cap \{x \in \mathbb{R}^n : c^\intercal x = 0\} = \{0\}$. Assume that $U = \{u_1, \ldots, u_r\}$ and $V = \{v_1, \ldots, v_t\}$ are two different inclusion-minimal Hilbert bases of C. Moreover, assume that w.l.o.g. $u_1 \notin V$. Minimality of U implies that u_1 cannot be written as a positive integer linear combination of elements in $U \setminus \{u_1\}$. However, as V is a Hilbert basis of C and as $u_1 \in C \cap \mathbb{Z}^n$, there is a non-negative integer linear combination $u_1 = \sum_{j=1}^{t} \alpha_j v_j$. Consequently, we also have $c^\intercal u_1 = \sum_{j=1}^{t} \alpha_j c^\intercal v_j$. Clearly, as $u_1 \notin V$, $\alpha_j \in \mathbb{Z}_+$, and $\{x \in \mathbb{R}^n : c^\intercal x = 0\} \cap C = \{0\}$, we have $c^\intercal v_j < c^\intercal u_1$ for all j with $\alpha_j > 0$. As also U is a Hilbert basis of C and as all $v_j \in C \cap \mathbb{Z}^n$, there are non-negative integer linear combinations $v_j = \sum_{i=1}^{r} \beta_{i,j} u_i$, $j = 1, \ldots, t$. Moreover, we have again $c^\intercal u_i \leq c^\intercal v_j$ whenever $\beta_{i,j} > 0$. Plugging these representations into $u_1 = \sum_{j=1}^{t} \alpha_j v_j$, we get a representation of u_1 as a non-negative integer linear combination of elements in U. However, by construction, they all have a scalar product with c that is strictly less than $c^\intercal u_1$. Thus, we have written u_1 as a non-negative integer linear combination of elements in $U \setminus \{u_1\}$, a contradiction to our assumption that U is an inclusion-minimal Hilbert basis, and the claim is proved. □

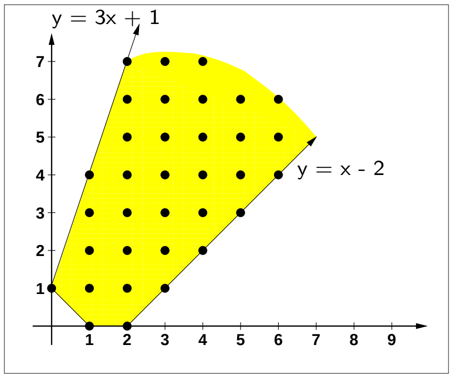
Finiteness of Hilbert bases already implies a nice and important finite description for all integer solutions of a system of linear inequalities, that is, for the lattice points in polyhedra, Lemma 1.2.1. If one has an algorithm to compute Hilbert bases at hand, one can even turn its proof into a construction to produce such a description. Moreover this lemma implies an integer version of Weyl's theorem, see Corollary 1.2.2.

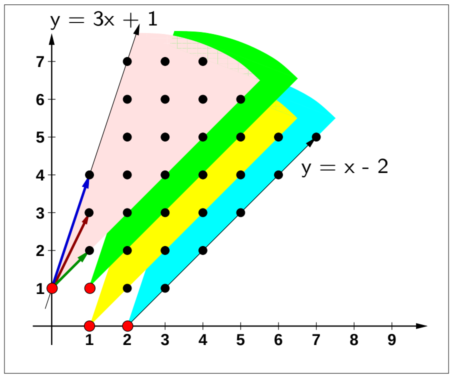
Lemma 1.2.1 *Let $P = \{z \in \mathbb{R}^n : Az \leq b\}$ be a polyhedron and $C = \{z \in \mathbb{R}^n : Az \leq 0\}$ its recession cone. If P contains a lattice point, that is $P \cap \mathbb{Z}^n \neq \emptyset$, then there exist finitely many points $z_1, \ldots, z_k \in P \cap \mathbb{Z}^n$ and $h_1, \ldots, h_s \in C \cap \mathbb{Z}^n$ such that every solution $z \in P \cap \mathbb{Z}^n$ can be written as*

$$z = z_i + \sum_{j=1}^{s} \alpha_j h_j,$$

for some $i \in \{1, \ldots, k\}$ and with $\alpha_j \in \mathbb{Z}_+$ for all $j = 1, \ldots, s$.

Moreover, the vectors $h_1, \ldots, h_s$ form a Hilbert basis of C.





Proof. Consider the rational polyhedral cone $\overline{P} := \{(z,u) \in \mathbb{R}^{n+1} : Az - bu \leq 0 : z \in P\}$. Note that $z \in P$ if and only if $(z,1) \in \overline{P}$ and $z \in C$ if and only if $(z,0) \in \overline{P}$. Consider a (finite) Hilbert basis H of $\overline{P}$ and let $\{z_1,\ldots,z_k\} := \{z : (z,1) \in H\}$ and $\{h_1,\ldots,h_s\} := \{h : (h,0) \in H\}$. We show now that these vectors z_i and h_j have the required property.

Let $z \in P \cap \mathbb{Z}^n$. Then $(z,1) \in \overline{P} \cap \mathbb{Z}^{n+1}$ and consequently,

$$(z,1) = \sum \alpha_j g_j, \tag{1.2.1}$$

with $\alpha_j \in \mathbb{Z}_+$ and $g_j \in H$ for all j. As the last component of each element in $\overline{P}$ is non-negative, the last component of each g_j with $\alpha_j > 0$ in Equation (1.2.1) must be 0 or 1. Moreover, exactly *one* such g_i has last component equal to 1 and necessarily $\alpha_i = 1$ and all the other vectors g_j, $j \neq i$, must have a zero last component. Thus,

$$(z,1) = (z_i,1) + \sum_{j=1}^{s} \alpha_j (h_j, 0),$$

for some $i \in \{1,\ldots,k\}$. Deleting the last component, we get the desired representation of z.

The vectors $h_1,\ldots,h_s$ form a Hilbert basis of C, since with $z \in C \cap \mathbb{Z}^n$ we have $(z,0) \in \overline{P}$ and consequently, all g_j with $\alpha_j > 0$ in the representation $(z,0) = \sum \alpha_j g_j$ must have zero last component and therefore are among the vectors $h_1,\ldots,h_s$. □

The representation from Lemma 1.2.1 immediately proves true an integer analogue of Weyl's theorem, which states that each polyhedron P can be written as the sum of a polytope Q plus the recession cone C of P.

Corollary 1.2.2 (Giles and Pulleyblank [64]) *Let $P = \{z : Az \leq b, z \in \mathbb{R}^n\}$ with $P \cap \mathbb{Z}^n \neq \emptyset$ and let $C = \{z : Az \leq 0, z \in \mathbb{R}^n\}$. Then there exists a polytope $Q \subseteq \mathbb{R}^n$ such that*

$$(P \cap \mathbb{Z}^n) = (Q \cap \mathbb{Z}^n) + (C \cap \mathbb{Z}^n).$$

Proof. Clearly, $Q = \operatorname{conv}(z_1,\ldots,z_k)$ with $z_1,\ldots,z_k$ as defined in the proof of Lemma 1.2.1 has the desired property. □

It should be noted that in general, the polytope Q from Weyl's theorem and the polytope Q from its integer analogue can be different.

Example 1.2.3 Let $P = \{z : 2z \leq 1, z \in \mathbb{R}\}$. Then we have $Q = \{1/2\}$ in the continuous and $Q = \{0\}$ in the integer setting. □

1.3 Proof of Theorem 1.0.2

For the proof of Theorem 1.0.2, we adapt the proof of Corollary 1.0.6. However, as we do not impose any structural assumption on S, we have to be cautious to check whether the integral generating set that we construct consists of lattice points from S only. The tricky part in the proof of Theorem 1.0.2 is to show that all constructed points lie indeed in S. The second example in Figure 1.1 on page 11 already illustrates this difficulty.

Proof of Theorem 1.0.2. Let us start showing part (a). If $C = \text{cone}(S)$ is not a rational polyhedral cone, S cannot have a finite integral generating set $G \subseteq S$, since $C = \text{cone}(S) = \text{cone}(G)$ would be a rational cone, contradicting our initial assumption on C.

Now we show the remaining claim that S has a finite integral generating set if $C = \text{cone}(S)$ is rational by explicitly constructing such a finite set. It should be noted that the constructed integral generating set is not necessarily minimal. As in the proof of Corollary 1.0.6, it suffices to prove the claim for full-dimensional cones C.

Next, let us triangulate C into (finitely many!) rational simplicial cones $C_1, \ldots, C_l$. Note that we can and do choose such a triangulation for which the generators of the cones C_i are also among the generators $v_1, \ldots, v_m$ of C. Then

$$S \subseteq \bigcup_{q \in Q \cap \mathbb{Z}^n} \bigcup_{i=1}^{l} (q + C_i).$$

We now construct a finite integral generating set for each of the sets $(q + C_i) \cap S$. W.l.o.g., we assume $C_i = \text{cone}(v_1, \ldots, v_r)$ (otherwise relabel the v_i). Consider the parallelepiped

$$F_i = \left\{ \sum_{j=1}^{r} \alpha_j v_j : 0 \leq \alpha_1, \ldots, \alpha_r < 1 \right\}.$$

As F_i is bounded, F_i contains only finitely many lattice points $\{f_1, \ldots, f_t\}$ in $\mathbb{Z}^n$. Moreover, $(q + C_i) \cap \mathbb{Z}^n$ is the disjoint union of the following t sets $F_{i,1}, \ldots, F_{i,t}$ with

$$F_{i,j} = \left\{ q + f_{i,j} + \sum_{k=1}^{r} \alpha_k v_k : \alpha_1, \ldots, \alpha_r \in \mathbb{Z}_+ \right\}.$$

Thus, it suffices to construct a finite integral generating set for each of the sets $F_{i,j} \cap S$. If we had $q + f_{i,j} \in S$, we had have already found a desired representation of the points $F_{i,j} \cap S$ by vectors from $\{v_1, \ldots, v_k, q + f_{i,j}\} \subseteq S$. Thus assume on the contrary that $q + f_{i,j} \notin S$. We construct now a finite set $H_{i,j} \subseteq S$ such that every $s \in F_{i,j} \cap S$ can be written as

$$s = h + \sum_{j=1}^{r} \alpha_j v_j$$

for some $h \in H_{i,j}$ and with $\alpha_j \in \mathbb{Z}_+$, $j = 1\ldots,m$, as desired. Consequently, $H_{i,j} \cup \{v_1,\ldots,v_k\} \subseteq S$ is a finite integral generating set for $F_{i,j} \cap S$.

As C_i is a simplicial cone, each point in $F_{i,j}$ has a unique representation as $q + f_{i,j} + \sum_{k=1}^r \alpha_k v_k$ implying that there is a one-to-one correspondence $\phi_{i,j}$ between $F_{i,j}$ and $\mathbb{Z}_+^r$ given by

$$\phi_{i,j}\left(q + f_{i,j} + \sum_{k=1}^r \alpha_k v_k\right) = (\alpha_1,\ldots,\alpha_r).$$

Consider the set $\phi_{i,j}(F_{i,j} \cap S) \subseteq \mathbb{Z}_+^r$. By the Gordan-Dickson Lemma, Lemma 1.1.2, there are only finitely many points $\{g_1,\ldots,g_p\}$ that are minimal with respect to the partial ordering $\leq$ defined on $\mathbb{Z}_+^r$. Let $H_{i,j} = \{\phi_{i,j}^{(-1)}(g_1),\ldots,\phi_{i,j}^{(-1)}(g_p)\} \subseteq S$. Thus, for every element $s \in F_{i,j} \cap S$ there exists some $g \in \{g_1,\ldots,g_p\}$ with $g \leq \phi_{i,j}(s)$, implying that $s = \phi_{i,j}^{(-1)}(g) + \sum_{k=1}^r \alpha_k v_k$ for $\phi_{i,j}^{(-1)}(g) \in H_{i,j}$ and $\alpha_k = (\phi_{i,j}(s) - g)^{(k)} \in \mathbb{Z}_+$, $k = 1,\ldots,r$. □

1.4 Proof of Corollary 1.0.5

Let us show part (a) first. If S is finite, nothing is left to show. Thus, assume that S is not finite and therefore also $C \neq \{0\}$. Assume that S contains all but finitely many integer points in $C \cap \mathbb{Z}_+^n$. In particular, S contains an (integer) point of every extreme ray of C, which implies that $C \subseteq \text{cone}(S)$. By Corollary 1.2.2, we have $S = Q + C$ with $Q = \text{conv}(G)$ for some $G \subseteq S$. Thus, also $\text{cone}(G) \subseteq \text{cone}(S)$. Thus,

$$\text{cone}(S) \subseteq \text{cone}(G) + \text{cone}(C) = \text{cone}(G) + C \subseteq \text{cone}(S),$$

implying $\text{cone}(S) = \text{cone}(G) + C$. Thus, $\text{cone}(S)$ is the Minkowski sum of two rational polyhedral cones and as such also a rational polyhedral cone. Consequently, S has a finite integral generating set by Theorem 1.0.2.

Now assume that there are infinitely many integer points in C that do not belong to S. In particular, $0 \notin P$ as otherwise $0 = A0 \leq b$ implying $C \subseteq P$ and thus $C \cap \mathbb{Z}^n \subseteq S$. Assume for the moment that each extreme ray R of C contains a (nonzero!) rational point v_R of P. Then $\{\lambda v_R : \lambda \geq 1\} \subseteq P \cap C$, since $\lambda v_R \leq v_R \leq 0$ and since $\lambda v_R \leq v_R \leq b$. By convexity of P, P must contain the convex hull H of all these half-lines $\{\lambda v_R : \lambda \geq 1, \lambda \in \mathbb{R}\}$. As

$$C \setminus H = \left\{x : x = \sum_R \lambda_R v_R, 0 \leq \sum_R \lambda_R < 1, \lambda_R \geq 0 \text{ for all rays } R\right\}$$

is bounded, only a finite number of integer points in C can lie in $C \setminus H$. As $S = P \cap \mathbb{Z}^n$, this implies that only finitely many integer points C can lie outside of S, contradicting our initial assumption on C. This implies that there exists an extreme ray R of C that does not contain any point of P.

We now show that $\text{cone}(S)$ cannot be a rational cone, and by Theorem 1.0.2, S does not have a finite integral generating set. Assume on the contrary that $\text{cone}(S)$ is rational. By convexity of S, every ray in $\text{cone}(S)$ has a nontrivial intersection with S and thus also with P. This implies that the extreme ray $R = \text{cone}(v_R)$ does not belong to $\text{cone}(S)$. As $\text{cone}(S)$ is rational, there exists a finite (rational) description

$$\text{cone}(S) = \{x \in \mathbb{R}^n : c_i^\intercal x \leq 0, i = 1, \ldots, p\}.$$

Then $v_R \notin \text{cone}(S)$ implies that there is some index j such that $c_j^\intercal v_R > 0$. Now consider any integer point $w \in S$. As $\text{conv}(S) = \text{conv}(G) + C$, all integer points on the half-line $\{w + \alpha v_R : \alpha \geq 0\}$ belong to S. Moreover, as v_R is a rational vector, there are *infinitely* many integer points on this half-line. However, as $c_j^\intercal v_R > 0$, we have $c_j^\intercal (w + \alpha v_R) > 0$ for sufficiently large α, implying that there are integer points of S that lie outside of $\text{cone}(S)$. This contradiction shows that $\text{cone}(S)$ cannot be rational and part (a) is proved.

Finally, part (b) of our claim follows again literally the lines of the uniqueness part in the proof of Lemma 1.0.6, where the notion of a Hilbert basis is replaced by the notion of an integral generating set. □

Chapter 2

Positive sum property

In this chapter we consider special integral generating sets, namely those that have the *positive sum property*. Informally, $G \subseteq S$ has the positive sum property with respect to $S \subseteq \mathbb{Z}^n$ if and only if $G \cap \mathbb{O}_j$ is an integral generating set of $S \cap \mathbb{O}_j$ for *every* orthant $\mathbb{O}_j$ of $\mathbb{R}^n$.

2.1 The relation $\sqsubseteq$

Let us start by generalizing the partial ordering $\leq$ on $\mathbb{R}^n_+$ to a partial ordering $\sqsubseteq$ on $\mathbb{R}^n$, and by exhibiting some useful facts about it.

Definition 2.1.1 *We call $u, v \in \mathbb{R}^n$* **sign-compatible** *if $u^{(j)}v^{(j)} \geq 0$ for all $j = 1, \ldots, n$.*

For $u, v \in \mathbb{R}^n$ we say that $u \sqsubseteq v$ if u and v are sign-compatible and if $|u^{(j)}| \leq |v^{(j)}|$ for all components $j = 1, \ldots, n$, that is, u belongs to the same orthant as v and its components are not greater in absolute value than the corresponding components of v.

Example 2.1.2 Checking the definition, we see that $(1, -1, 0) \sqsubseteq (2, -1, 4)$. On the other hand, we have $(1, -1, 0) \not\sqsubseteq (2, 1, 4)$, since the signs of the second components disagree. $\square$

For $v \in \mathbb{Z}^n$ we component-wise define $v^+ := \max(0, v)$ and $v^- := (-v)^+ = \max(0, -v)$. With this, it can easily been checked that $u \sqsubseteq v$ if and only if $(u^+, u^-) \leq (v^+, v^-)$. This simple correspondence readily implies an extension of both versions of the Gordan-Dickson Lemma to the $\sqsubseteq$-situation.

Lemma 2.1.3 (Gordan-Dickson Lemma, $\sqsubseteq$ version)

- *Every sequence $\{p_1, p_2, \ldots\}$ of points in $\mathbb{Z}^n$ such that $p_i \not\sqsubseteq p_j$ whenever $i < j$, is finite.*
- *Every infinite set $S \subseteq \mathbb{Z}^n$ contains only finitely many $\sqsubseteq$-minimal points.*

The following fact about the relation $\sqsubseteq$ will turn out very useful in the subsequent proofs.

Lemma 2.1.4 *Let $u, v, w \in \mathbb{R}^n$ with $u \sqsubseteq v - w$. Then for each of the n components of $v - u$ and of $w + u$ we have*

- $w^{(i)} \leq (v-u)^{(i)} \leq v^{(i)}$ *and* $w^{(i)} \leq (w+u)^{(i)} \leq v^{(i)}$, *if* $(v-w)^{(i)} \geq 0$,
- $w^{(i)} \geq (v-u)^{(i)} \geq v^{(i)}$ *and* $w^{(i)} \geq (w+u)^{(i)} \geq v^{(i)}$, *if* $(v-w)^{(i)} \leq 0$.

In other words, each component $(v-u)^{(i)}$ of $v-u$ and each component $(w+u)^{(i)}$ of $w+u$ lies in the interval defined by the components $v^{(i)}$ and $w^{(i)}$ of v and w.

For $w = 0$, the inequalities involving $(v-u)^{(i)}$ imply $v - u \sqsubseteq v$.

Proof. First observe that $u \sqsubseteq v - w$ is equivalent to $-u \sqsubseteq -(v-w)$. Let us write down the definition of $-u \sqsubseteq -(v-w)$ component-wise:

- $0 \leq -u^{(i)} \leq -(v-w)^{(i)}$, if $-(v-w)^{(i)} \geq 0$,
- $0 \geq -u^{(i)} \geq -(v-w)^{(i)}$, if $-(v-w)^{(i)} \leq 0$.

Now we simply add $v^{(i)}$ to the inequalities on the left-hand sides and we are done:

- $v^{(i)} \leq (v-u)^{(i)} \leq w^{(i)}$, if $(v-w)^{(i)} \leq 0$,
- $v^{(i)} \geq (v-u)^{(i)} \geq w^{(i)}$, if $(v-w)^{(i)} \geq 0$.

Analogously, to obtain the inequalities for $(w+u)^{(i)}$ write down the defining inequalities for $u \sqsubseteq v - w$ and add $w^{(i)}$ to all parts of these inequalities. □

2.2 Positive sum property

Now we are ready to introduce the basic notion of this chapter.

Definition 2.2.1 (Positive Sum Property) *We say that $v \in \mathbb{Z}^n$ is* **$\sqsubseteq$-representable** *with respect to $G \subseteq \mathbb{Z}^n$, if v can be written as a finite integer linear combination $v = \sum_{i \in I} \alpha_i g_i$ with the property that for all $i \in I$ we have*

- $g_i \in G$, $\alpha_i > 0$, $\alpha_i \in \mathbb{Z}_{>0}$, *and*
- $g_i \sqsubseteq v$.

A set G has the **positive sum property** *with respect to $S \subseteq \mathbb{Z}^n$ if $G \subseteq S$ and if every non-zero $v \in S$ is $\sqsubseteq$-representable with respect to G.*

Example 2.2.2 Trivially, the set $\{\pm e_1, \ldots, \pm e_n\}$ has the positive sum property with respect to $\mathbb{Z}^n$, whereas the set $\{e_1, \ldots, e_n\}$ has the positive sum property with respect to $\mathbb{Z}^n_+$. □

Finding for given $S \subseteq \mathbb{Z}^n$ a set G that has the positive sum property with respect to S is not easy in general. The following lemma gives a characterization for the case that S is a sublattice of $\mathbb{Z}^n$.

Lemma 2.2.3 *Let $\mathcal{L}$ be a sublattice of $\mathbb{Z}^n$ and let $G \subseteq \mathcal{L}$. The set G has the positive sum property with respect to $\mathcal{L}$ if and only if G contains the set M of all $\sqsubseteq$-minimal elements of $\mathcal{L} \setminus \{0\}$.*

Proof. Assume that G has the positive sum property with respect to $\mathcal{L}$ and let $v \in M$. Then v must belong to G, which can be seen as follows. If $v \notin G$, then the positive sum property of G with respect to $\mathcal{L}$ implies that there exists a positive integer linear combination $v = \sum \alpha_i g_i$ with nonzero vectors $g_i \in G$ and with $g_i \sqsubseteq v$. The condition $g_i \sqsubseteq v$, however, contradicts the $\sqsubseteq$-minimality of v unless $v = g_i$. Therefore, $v \in G$ and consequently, $M \subseteq G$.

On the other hand, the relation $M \subseteq G$ already implies that G has the positive sum property with respect to $\mathcal{L}$. Suppose on the contrary that there is some $z \in \mathcal{L} \setminus \{0\}$ that cannot be written as a positive integer linear combination of elements in M which are all sign-compatible with z. Among all such vectors z choose one that has a smallest l_1-norm (which is of course a positive integer). Clearly, $z \notin M$, that is, z is not $\sqsubseteq$-minimal. Therefore, there is some $v \in M$ with $v \sqsubseteq z$. By our minimality assumption on $\|z\|_1$, there exists a positive integer linear combination $z - v = \sum \alpha_i g_i$ with $g_i \in M$ and $g_i \sqsubseteq z - v$, since $\|z - v\|_1 < \|z\|_1$. But now we have $z = v + \sum \alpha_i g_i$ with $v, g_i \in M$, $v \sqsubseteq z$, and $g_i \sqsubseteq z - v \sqsubseteq z$. This contradicts our non-representability assumption on z. $\square$

A simple consequence of this lemma is the following.

Corollary 2.2.4 *For any sublattice $\mathcal{L}$ of $\mathbb{Z}^n$, the set of $\sqsubseteq$-minimal elements of $\mathcal{L} \setminus \{0\}$ is the unique inclusion-minimal set that has the positive sum property with respect to $\mathcal{L}$.*

A natural question to ask is how this corollary looks like if we fix our attention to just one orthant $\mathbb{O}_j$ of $\mathbb{R}^n$. Here is the straight-forward answer from the last corollary.

Corollary 2.2.5 *For any sublattice $\mathcal{L}$ of $\mathbb{Z}^n$, the set of $\sqsubseteq$-minimal elements of $(\mathcal{L} \cap \mathbb{O}_j) \setminus \{0\}$ is the unique inclusion-minimal set that has the positive sum property with respect to $\mathcal{L} \cap \mathbb{O}_j$.*

In fact, this result holds for any union of orthants. As indicated already in the introduction, we are sometimes interested in sets having the positive sum property with respect to $\{x \in \mathcal{L} : x_I \geq 0\}$, where $I \subseteq \{1, \ldots, n\}$. Herein, we take the union over $2^{|I|}$ orthants.

Corollary 2.2.6 *For any sublattice $\mathcal{L}$ of $\mathbb{Z}^n$ and for any index set $I \subseteq \{1, \ldots, n\}$, the set of $\sqsubseteq$-minimal elements of $\{x \in \mathcal{L} : x_I \geq 0\} \setminus \{0\}$ is the unique inclusion-minimal set that has the positive sum property with respect to $\{x \in \mathcal{L} : x_I \geq 0\}$.*

2.3 How to check the positive sum property

So far, checking whether a given set $G \subseteq \mathbb{Z}^n$ has the positive sum property with respect to a given lattice $\mathcal{L}$ would require infinitely many $\sqsubseteq$-representability tests. Although Lemma 2.2.3

characterizes the elements of $\mathcal{L}$ that need to be computed in order to complete a given generating set of lattice vectors with respect to the positive sum property, it does not say how to do this step algorithmically. The following criterion, however, is different. It states that only finitely many "critical vectors" need be tested for representability. It is this lemma that leads to a finite algorithm to compute a set of vectors that has the positive sum property with respect to the given lattice, see Section 2.4.

Lemma 2.3.1 (Criterion for positive sum property with respect to integer lattice) *Let $\mathcal{L}$ be an integer sublattice of $\mathbb{Z}^n$. A symmetric set $G \subseteq \mathcal{L}$, that is $v \in G$ implies $-v \in G$, has the positive sum property with respect to $\mathcal{L}$ if and only if the following two conditions hold:*

- *G finitely generates $\mathcal{L}$ over $\mathbb{Z}$, and*
- *for every pair $v, w \in G$, the vector $v + w$ is $\sqsubseteq$-representable with respect to G.*

Proof. We have to show that any non-zero $z \in \mathcal{L}$ can be written as a finite positive linear integer combination of elements from G where each vector in this combination is sign-compatible with z. Since G generates $\mathcal{L}$ over $\mathbb{Z}$ and since G is symmetric, the vector z can be written as a linear combination $z = \sum \alpha_i v_i$ for finitely many $\alpha_i \in \mathbb{Z}_{>0}$ and $v_i \in G$. Note that $\sum \alpha_i \|v_i\|_1 \geq \|z\|_1$ with equality if and only if $v_i \sqsubseteq z$ for all i.

From the set of all such linear integer combinations $\sum \alpha_i v_i$ choose one such that $\sum \alpha_i \|v_i\|_1$ is minimal and assume that $\sum \alpha_i \|v_i\|_1 > \|z\|_1$. Otherwise $v_i \sqsubseteq z$ for all i and we are done. Therefore, there have to exist vectors v_{i_1}, v_{i_2} in this representation which have some component $k = k_0$ of different signs.

By the assumptions on G, the vector $v_{i_1} + v_{i_2}$ can be written as $v_{i_1} + v_{i_2} = \sum \beta_j v'_j$ for finitely many $\beta_j \in \mathbb{Z}_{>0}$, $v'_j \in G$, and $\beta_j v'_j \sqsubseteq v_{i_1} + v_{i_2}$ for all j. The latter implies that we have for each component $k = 1, \ldots, n$,

$$\sum_j \beta_j |{v'_j}^{(k)}| = |\sum_j \beta_j {v'_j}^{(k)}| = |(v_{i_1} + v_{i_2})^{(k)}| \leq |v_{i_1}^{(k)}| + |v_{i_2}^{(k)}|,$$

where the last inequality is strict for $k = k_0$ by construction. Summing up over $k = 1, \ldots, n$, yields $\sum \beta_j \|v'_j\|_1 = \|v_{i_1} + v_{i_2}\|_1 < \|v_{i_1}\|_1 + \|v_{i_2}\|_1$. But now z can be represented as

$$\begin{aligned} z &= \alpha_{i_1} v_{i_1} + \alpha_{i_2} v_{i_2} + \sum_{i \neq i_1, i_2} \alpha_i v_i \\ &= \sum \beta_j v'_j + (\alpha_{i_1} - 1) v_{i_1} + (\alpha_{i_2} - 1) v_{i_2} + \sum_{i \neq i_1, i_2} \alpha_i v_i \end{aligned}$$

and it holds that

$$\sum \beta_j \|v'_j\|_1 + (\alpha_{i_1} - 1)\|v_{i_1}\|_1 + (\alpha_{i_2} - 1)\|v_{i_2}\|_1 + \sum_{i \neq i_1, i_2} \alpha_i \|v_i\|_1 < \sum \alpha_i \|v_i\|_1$$

in contradiction to the minimality required on the sum $\sum \alpha_i \|v_i\|_1$. Altogether we obtain that $\|z\|_1 = \sum \alpha_i \|v_i\|_1$, that is, G has the positive sum property with respect to $\mathcal{L}$. $\square$

Sometimes we can exploit an already known or pre-computed structure of G. This structure allows us to reduce the number of critical vectors $v+w$ that need to be checked in the second condition above and leads to a tremendous speed-up of the algorithm, see Section 2.5.

Example. The lattice $\mathcal{L} = \ker_{\mathbb{Z}^{n+d}}(A|I_d)$ is generated over $\mathbb{Z}$ by the vectors $(e_1, -Ae_1)$, $\ldots$, $(e_n, -Ae_n)$. Obviously, the set $G = \{\pm(e_1, -Ae_1), \ldots, \pm(e_n, -Ae_n)\}$ has already the positive sum property with respect to $\mathcal{L}$ on the first n components. This nice property is exploited in the following criterion. $\square$

Lemma 2.3.2 (Another criterion for positive sum property with respect to a lattice) *Let $\mathcal{L}$ be an integer sublattice of $\mathbb{Z}^n$ and let π_s be the projection of vectors in $\mathbb{R}^n$ onto their first s components. Moreover, let $G \subseteq \mathcal{L}$ be a set such that $\pi_s(G)$ has the positive sum property with respect to $\pi_s(\mathcal{L})$. In addition, we require that the set $\{v \in G : \pi_s(v) = 0\}$ is symmetric.*

Under these assumptions, the set G has the positive sum property with respect to $\mathcal{L}$ if and only if the following two conditions hold:

- *$\{v \in G : \pi_s(v) = 0\}$ finitely generates $\{v \in \mathcal{L} : \pi_s(v) = 0\}$ over $\mathbb{Z}$, and*
- *for every pair $v, w \in G$ for which $\pi_s(v)$ and $\pi_s(w)$ are sign-compatible, the vector $v + w$ is $\sqsubseteq$-representable with respect to G.*

Proof. Again, we have to show that any $z \in \mathcal{L} \setminus \{0\}$ can be written as a finite positive integer linear combination of elements from G where each vector in this combination belongs to the same orthant as z. Since $\pi_s(G)$ has the positive sum property with respect to $\pi_s(\mathcal{L})$, we can write $\pi_s(z) = \sum \alpha_i \pi_s(v_i)$ for finitely many $\alpha_i \in \mathbb{Z}_{>0}$ and $v_i \in G$, $\pi_s(v_i) \sqsubseteq \pi_s(z)$.

Thus, we know that $\pi_s(z - \sum \alpha_i v_i) = 0$ and therefore, $z - \sum \alpha_i v_i$ can be written as a positive integer linear combination of elements from $\{v \in G : \pi_s(v) = 0\}$. Combining these two representations, we get a representation $z = \sum \alpha_i v_i$ with $\alpha_i \in \mathbb{Z}_{>0}$, $v_i \in G$, and $\pi_s(v_i) \sqsubseteq \pi_s(z)$, that is, in the first s components all v_i have the same signs as z.

If we consider only such representations for z as just constructed, the remainder of the proof follows literally the proof of Criterion 2.3.1. $\square$

2.4 How to complete a set to have the positive sum property

Lemma 2.3.1 reduces the decision whether a finite symmetric generating set F of $\mathcal{L}$ has the positive sum property with respect to $\mathcal{L}$ to a finite number of representability tests which may be treated algorithmically. In order to complete a set of vectors with respect to the positive sum property, this lemma suggests the following procedure which adds further elements to the set as long as it does not have the desired positive sum property, see Pottier [102]. Such procedures are known as completion procedures [27].

Algorithm 2.4.1 (Completion Procedure)

Input: a finite symmetric set $F \subseteq \ker_{\mathbb{Z}^n}(A)$ generating $\mathcal{L}$ over $\mathbb{Z}$

Output: a set $G \supseteq F$ that has the positive sum property with respect to $\mathcal{L}$

```
$G := F$
$C := \bigcup_{f,g \in G} \text{S-vectors}(f,g)$
while $C \neq \emptyset$ do
    $s :=$ an element in $C$
    $C := C \setminus \{s\}$
    $f := \text{normalForm}(s, G)$
    if $f \neq 0$ then
        $C := C \cup \bigcup_{g \in G} \text{S-vectors}(f,g)$
        $G := G \cup \{f\}$
return $G$.
```

In this algorithm, the set S-vectors(f, g) corresponds to the “critical” vectors that are described in Lemma 2.3.1, that is, S-vectors$(f, g) = \{f + g\}$.

Behind the function normalForm(s, G) there is the following algorithm which returns 0 if a $\sqsubseteq$-representation $s = \sum \alpha_i g_i$ with finitely many $\alpha_i \in \mathbb{Z}_{>0}$ and $g_i \in G$, $g_i \sqsubseteq z$ is found, or it returns a vector t such that a $\sqsubseteq$-representation of s is possible with respect to $G \cup \{t\}$.

The normalForm algorithm aims at finding such a $\sqsubseteq$-representation $s = \sum \alpha_i g_i$ by reducing s by elements of G in such a way that, if at some point of this reduction the zero vector is reached, a $\sqsubseteq$-representation $s = \sum \alpha_i g_i$ has been found. If the reduction process terminates with a vector $t \neq 0$ then a $\sqsubseteq$-representation $s = \sum \alpha_i g_i$ with respect to $G \cup \{t\}$ has been constructed. The vector t is called a *normal form* of s with respect to the set G.

Algorithm 2.4.2 (Normal form algorithm)

Input: a vector s, a set G of vectors

Output: a normal form of s with respect to G

```
while there is some $g \in G$ such that $s$ is reducible by $g$ do
    $s :=$ reduce $s$ by $g$
return $s$
```

The reduction involved in the normalForm algorithm remains to be defined. We say that $s \in \mathbb{Z}^n$ can be reduced by $g \in \mathbb{Z}^n$ to $s - g$ if $g \sqsubseteq s$. Thus, in case of reducibility, we have $s = g + (s - g)$ with $g \sqsubseteq s$ and $s - g \sqsubseteq s$. Since $\|s - g\|_1 < \|s\|_1$, we conclude that normalForm(s, G) always terminates.

Lemma 2.4.3 (Pottier [102]) *With the above definitions of* S-vectors *and of* normalForm *the Completion Procedure 2.4.1 terminates and satisfies its specifications.*

Proof. Termination of the above algorithm follows immediately by application of the $\sqsubseteq$-version of the Gordan-Dickson Lemma, Lemma 2.1.3, to the sequence of vectors $\{v \in G \setminus F\}$ as they were added to G by the algorithm. To see this, note that $f = \text{normalForm}(s, G)$ implies that there is no $g \in G$ with $g \sqsubseteq f$. Thus, the algorithm produces a sequence of vectors $\{v \in G \setminus F\} = \{f_1, f_2, \ldots\}$, where $f_i \not\sqsubseteq f_j$ for $i < j$. Such a sequence is always finite by Lemma 2.1.3. Correctness of the algorithm follows immediately by Lemma 2.3.1, since upon termination $\text{normalForm}(v+w, G) = 0$ for all $v, w \in G$, giving a representation $v + w = \sum \alpha_i g_i$ with $\alpha_i \in \mathbb{Z}_{>0}$, $g_i \in G$, and $g_i \sqsubseteq v + w$. □

Remark. Let us state here a simple criterion to reduce the number of critical vectors $v + w$ that need to be checked. If v and w belong to the same orthant, the relation $v, w \sqsubseteq (v + w)$ implies that $(v + w) = v + w$ is already a desired $\sqsubseteq$-representation of $(v + w)$as in the second condition of Lemma 2.3.1 and we do not need to waste time searching the (possibly huge) set G for suitable vectors g_i. □

2.5 Speeding up the completion procedure

In this section we exploit the criterion given in Lemma 2.3.2 instead of the one given in Lemma 2.3.1 to complete a generating set of $\mathcal{L}$ with respect to the positive sum property. In fact, we wish to solve a more general problem:

"Compute the $\sqsubseteq$-minimal elements in $\mathcal{L}_I := \{x \in \mathcal{L} : x_I \geq 0\} \setminus \{0\}$ for given $I \subseteq \{1, \ldots, n\}$."

For this, we first introduce a special generating set of $\mathcal{L}$. This generating set not only decreases the number of sums $f + g$ that need to be added to C, but more importantly, it reduces the amount of work to compute $\text{normalForm}(s, G)$ tremendously. As the latter computation is the most expensive part of Algorithm 2.4.1, this heavily speeds up the computation. Moreover, we employ a so-called critical-pair selection strategy that chooses the next element s from C according to a certain rule. This, together with the special input set, will imply that the set G returned by our algorithm is *exactly* the set of $\sqsubseteq$-minimal elements (in the case that no non-negativity constraint is present). Not a single unnecessary vector is computed!

In Algorithm 2.4.1, one has to wait until the very end to extract the $\sqsubseteq$-minimal elements from the returned set G. In contrast to this, our algorithm provides a *certificate* of $\sqsubseteq$-minimality for each vector that is added to G.

Let us assume for the rest of this chapter that $\mathcal{L}$ is generated by r linearly independent vectors and that the first r components of the vectors of $\mathcal{L}$ are linearly independent, that is, the only r-dimensional vector that is zero on the first r components is 0. Consequently, the first r components of a vector $v \in \mathcal{L}$ uniquely define v. In order to compute this lifting quickly whenever needed, we may assume that the generators $\{p_1, \ldots, p_r\} \subseteq \mathbb{Z}^n$ of $\mathcal{L}$ have the following structure, which is clearly achievable by suitable elementary integral row operations and permutations of columns:

$$\begin{array}{rcllllllll}
p_1 & = & (& p_{1,1}, & p_{1,2}, & \ldots, & \ldots, & p_{1,r}, & \ldots, & p_{1,n} \quad), \\
p_2 & = & (& 0, & p_{2,2}, & \ldots, & \ldots, & p_{2,r}, & \ldots, & p_{2,n} \quad), \\
p_3 & = & (& 0, & 0, & p_{3,3} & \ldots, & p_{3,r}, & \ldots, & p_{3,n} \quad), \\
\vdots & & (& \vdots, & \vdots, & \ldots, & \ddots, & \vdots, & \ldots, & \vdots \quad), \\
p_r & = & (& 0, & 0, & \ldots, & 0, & p_{r,r}, & \ldots, & p_{r,n} \quad),
\end{array}$$

with $p_{i,i} > 0$, $i = 1, \ldots, r$.

Let π_d denote the projection of an n-dimensional vector onto its first $d \geq r$ components. Next let us define a norm $\|.\|$ on vectors $v \in \mathcal{L}$ as follows: $\|v\| := \|\pi_d(v)\|_1$, where $\|.\|_1$ denotes the l_1-norm on $\mathbb{R}^d$. It can easily be checked that, under our assumptions on $\mathcal{L}$, this defines indeed a norm on $\mathcal{L}$. It is this norm definition that breaks existing symmetry of the given problem.

Putting $J := I \cap \{1, \ldots, d\}$, we clearly have $\mathcal{L}_I \subseteq \mathcal{L}_J$. Let $\pi_d(\overline{F})$ denote the set of all $\sqsubseteq$-minimal nonzero vectors in $\pi_d(\mathcal{L}_J)$. By Corollary 2.2.6, $\pi_d(\overline{F})$ has the positive sum property with respect to $\pi_d(\mathcal{L}_J)$, that is, every vector $\pi_d(v) \in \pi_d(\mathcal{L}_J)$ is $\sqsubseteq$-representable with respect to $\pi_d(\overline{F})$. Such a representation $\pi_d(v) = \sum \alpha_i \pi_d(f_i)$ with $\alpha_i \in \mathbb{Z}_{>0}$ and $\pi_d(f_i) \sqsubseteq \pi_d(v)$ can be uniquely lifted to $v = \sum \alpha_i f_i$ showing in particular that $\overline{F}$ generates $\mathcal{L}_J$ (and thus also $\mathcal{L}_I$) over $\mathbb{Z}_+$.

Moreover, this set $\pi_d(\overline{F})$ can be computed for example via Algorithm 2.4.1 once a lattice basis F of $\pi_d(\mathcal{L}) \subseteq \mathbb{Z}^d$ over $\mathbb{Z}$ is given as input. Note that all these $\sqsubseteq$-minimal vectors in $\pi_d(\overline{F})$ must lift to $\sqsubseteq$-minimal elements in $\mathcal{L}$, since they are already indecomposable/minimal on the first d components. This approach via Algorithm 2.4.1 is not very efficient in practice, although in our computational experiments, it often happened that the set $\pi_r(\overline{F})$, in case that $d = r$, simply contained the unit vectors in $\mathbb{Z}^r$ and possibly their negatives. However, the reference to Algorithm 2.4.1 shall suffice here to convince the reader that $\pi_d(\overline{F})$ can be found without computing the $\sqsubseteq$-minimal elements in $\mathcal{L}$ first.

Let us now define the pieces for the project-and-lift algorithm that computes the $\sqsubseteq$-minimal elements in $\mathcal{L}_J \setminus \{0\}$ and thus also in $\mathcal{L}_I \setminus \{0\}$.

Algorithm 2.5.1 (Faster Algorithm to Compute $\sqsubseteq$-minimal elements in $\mathcal{L}_J \setminus \{0\}$)

Input: set $\overline{F} \subseteq \mathcal{L}_J$ such that $\pi_d(\overline{F})$ is the set of $\sqsubseteq$-minimal nonzero vectors in $\pi_d(\mathcal{L}_J)$

Output: a set G which contains all $\sqsubseteq$-minimal vectors in $\mathcal{L}_J \setminus \{0\}$

```
$G := \overline{F}$
$C := \bigcup_{f,g \in G : \pi_d(f) \text{ and } \pi_d(g) \text{ are sign-compatible}} \{f + g\}$
while $C \neq \emptyset$ do
    $s :=$ an element in $C$ with smallest $\|.\|$-norm
```

```
        $C := C \setminus \{s\}$
        $f :=$ normalForm$(s, G)$
        if $f \neq 0$ then
                $G := G \cup \{f\}$
                $C := C \cup \bigcup_{g \in G : \pi_d(f) \text{ and } \pi_d(g) \text{ lie in the same orthant of } \mathbb{R}^d} \{f + g\}$
return $G$.
```

Due to our special input set, the function normalForm(s, G) can be sped up as follows.

Algorithm 2.5.2 (Faster Normal Form Algorithm)

```
Input: a vector $s$, a set $G$ of vectors
Output: a normal form of $s$ with respect to $G$

if there is some $g \in G$ such that $g \sqsubseteq s$ return 0
return $s$
```

Once the normal form algorithm finds a vector $g \in G$ with $g \sqsubseteq s$, it would reduce s to $s - g$ with $\|s - g\|_1 < \|s\|_1$. As we will show below, the algorithm can conclude now that a representation $s = g_1 + \ldots + g_r$, $g_i \in G$, $g_i \sqsubseteq s$, $i = 1, \ldots, r$, will exist upon termination of the algorithm. Therefore, the function normalForm(s, G) can return 0 immediately without explicitly constructing such a relation.

Since we started with a very special input set that has the positive sum property already on the first d components, only those pairs of vectors $f, g \in G$ lead to a critical vector in C, for which the projections $\pi_d(f)$ and $\pi_d(g)$ are sign-compatible.

Finally, let us prove our claims. There are in fact two claims. First, the special input set, the restricted set of critical pairs, and the new definition for the normal form algorithm guarantee that a superset of all $\sqsubseteq$-minimal elements in $\mathcal{L}_J$ is computed. Second, the selection strategy that chooses the next s by increasing norm $\|.\|$ guarantees that no element is computed that is not $\sqsubseteq$-minimal.

Lemma 2.5.3 *Algorithm 2.5.1 always terminates and returns a set G containing all $\sqsubseteq$-minimal elements in $\mathcal{L}_J \setminus \{0\}$.*

Proof. To prove termination, consider the sequence of vectors in $G \setminus \overline{F} = \{f_1, f_2, \ldots\}$ as they are added to G during the run of the algorithm. By construction, we have $f_i \not\sqsubseteq f_j$, that is $(f_i^+, f_i^-) \not\leq (f_j^+, f_j^-)$ whenever $i < j$. Thus, by the Gordan-Dickson Lemma, this sequence must be finite and the algorithm terminates.

To prove correctness, let us assume that z is $\sqsubseteq$-minimal in $\mathcal{L}_J \setminus \{0\}$ but is not contained in the final output set G of Algorithm 2.5.1. Without loss of generality we may assume that z has a smallest norm $\|z\|$ among all such $\sqsubseteq$-minimal vectors in $\mathcal{L}_J \setminus \{0\}$. Therefore, we can assume that all $\sqsubseteq$-minimal elements g in $\mathcal{L}_J \setminus \{0\}$ with $\|g\| < \|z\|$ are contained in G. In the following, we construct a contradiction to the assumption $z \notin G$ and correctness of Algorithm 2.5.1 is proved. Note that we essentially follow the construction of the proof of Lemma 2.3.1. We only need to show that each step is still correct with our new definitions.

As $\overline{F}$ is contained in G, there is a representation $z = \sum \alpha_i v_i$ with positive integers α_i and vectors $v_i \in G$ with $\pi_d(v_i) \sqsubseteq \pi_d(z)$. From the set of all such linear integer combinations choose one such that $\sum \alpha_i \|v_i\|_1$ is minimal.

Let us assume first that $\sum \alpha_i \|v_i\|_1 > \|z\|_1$. Therefore, there have to exist vectors v_{i_1}, v_{i_2} in this representation which have some component $k = k_0$ of different signs. By construction, $k_0 > d$, as the v_i all have the same sign as z on the first d components.

The vector $v_{i_1} + v_{i_2}$ was added to C during the run of Algorithm 2.5.1 (as $\pi_d(v_{i_1})$ and $\pi_d(v_{i_2})$ are sign-compatible by construction). If $\|v_{i_1} + v_{i_2}\| = \|z\|$, then all other v_i, $i \neq i_1, i_2$, must satisfy $\|v_i\| = 0$ as $\pi_d(v_i) \sqsubseteq \pi_d(z)$ for all i. But $\pi_d(v_i) = 0$ implies $v_i = 0$ and thus $v_{i_1} + v_{i_2} = z$. Since $v_{i_1} + v_{i_2} (= z)$ is a vector that was added to C during the run of the algorithm, the vector z is eventually chosen as $s \in C$. Being $\sqsubseteq$-minimal, we have normalForm$(z, G) = z$ and thus, z must have been added to G, in contradiction to our assumption $z \notin G$.

Therefore, we may assume that $\|v_{i_1} + v_{i_2}\| < \|z\|$. However, since all $\sqsubseteq$-minimal elements v in $\mathcal{L}_J \setminus \{0\}$ with norm $\|v\| < \|z\|$ are assumed to be in G, there must exist a representation $v_{i_1} + v_{i_2} = \sum \beta_j v'_j$ for finitely many $\beta_j \in \mathbb{Z}_{>0}$, $v'_j \in G$, and $\beta_j v'_j \sqsubseteq v_{i_1} + v_{i_2}$ for all j. This implies that we have for each component $k = 1, \ldots, n$,

$$\sum_j \beta_j |{v'_j}^{(k)}| = |\sum_j \beta_j {v'_j}^{(k)}| = |(v_{i_1} + v_{i_2})^{(k)}| \leq |v_{i_1}^{(k)}| + |v_{i_2}^{(k)}|,$$

where the last inequality is strict for $k = k_0$ by construction. Summing up over $k = 1, \ldots, n$, yields $\sum \beta_j \|v'_j\|_1 = \|v_{i_1} + v_{i_2}\|_1 < \|v_{i_1}\|_1 + \|v_{i_2}\|_1$. But now z can be represented as

$$\begin{aligned} z &= \alpha_{i_1} v_{i_1} + \alpha_{i_2} v_{i_2} + \sum_{i \neq i_1, i_2} \alpha_i v_i \\ &= \sum \beta_j v'_j + (\alpha_{i_1} - 1) v_{i_1} + (\alpha_{i_2} - 1) v_{i_2} + \sum_{i \neq i_1, i_2} \alpha_i v_i \end{aligned}$$

and it holds

$$\sum \beta_j \|v'_j\|_1 + (\alpha_{i_1} - 1)\|v_{i_1}\|_1 + (\alpha_{i_2} - 1)\|v_{i_2}\|_1 + \sum_{i \neq i_1, i_2} \alpha_i \|v_i\|_1 < \sum \alpha_i \|v_i\|_1$$

in contradiction to the minimality required on $\sum \alpha_i \|v_i\|_1$. Thus, our assumption $\sum \alpha_i \|v_i\|_1 > \|z\|_1$ was wrong and $\sum \alpha_i \|v_i\|_1 = \|z\|_1$ must hold.

But this last equation implies that $v_i \sqsubseteq z$ for all i, contradicting $\sqsubseteq$-minimality of z unless the representation $z = \sum \alpha_i v_i$ is trivial, that is $z = v_1 \in G$. This, however, again contradicts our initial assumption $z \notin G$ and thus all $\sqsubseteq$-minimal elements in $\mathcal{L}_J \setminus \{0\}$ are in G as claimed. $\square$

It should be noted that we did not make use of our selection strategy to prove termination and correctness of Algorithm 2.5.1. It is the following Lemma that provides a certificate for $\sqsubseteq$-minimality of vectors in G.

Lemma 2.5.4 *The set G returned by Algorithm 2.5.1 is exactly the set of $\sqsubseteq$-minimal elements in $\mathcal{L}_J \setminus \{0\}$.*

Proof. The crucial observation for this proof is that the norms $\|s\|$ of the vectors s that are chosen from C form a non-decreasing sequence. This follows from the definition that normalForm(s, g) only returns either 0 or s and from our condition that only vectors $f + g$ are added to C whose first d components are sign-compatible. The latter implies $\|f + g\| = \|f\| + \|g\|$. Thus, when we remove an element s from C which turns out to be irreducible by the elements of G, that is, if $s = \text{normalForm}(s, G)$, we only add new elements $s+g$ to C that satisfy $\|s+g\| = \|s\|+\|g\| > \|s\|$.

Now assume that some $z \in G$ is not $\sqsubseteq$-minimal in $\mathcal{L}_J \setminus \{0\}$. Thus, there is some $\sqsubseteq$-minimal element g in $\mathcal{L}_J \setminus \{0\}$ with $g \sqsubseteq z$. Under our assumptions on $\mathcal{L}$, $g \sqsubseteq z$ implies $\|g\| < \|z\|$. Since G contains all $\sqsubseteq$-minimal elements of $\mathcal{L}_J \setminus \{0\}$ (and thus in particular the vector g) at the end of the algorithm and since after z only vectors f are added to G that have a norm $\|f\| \geq \|z\|$, the vector g must have been contained in G already at the time when z was added to G. This however, implies that Algorithm 2.5.1 must have computed normalForm$(z, G) = 0$, a contradiction to the assumption that z was added to G. □

2.6 Computing Hilbert bases and Graver bases

Let us clarify in this section how to invoke this project-and-lift approach more effectively in order to compute the nonzero $\sqsubseteq$-minimal elements of a lattice $\mathcal{L} \subseteq \mathbb{Z}^n$ and of $\mathcal{L}_+$. Assume again, that $\mathcal{L}$ is spanned by r linearly independent vectors and that the first r components of $\mathcal{L}$ are linearly independent, too.

2.6.1 $\sqsubseteq$-minimal elements in $\mathcal{L} \setminus \{0\}$

This question is easy to solve. First compute a set $\overline{F}$ such that $\pi_r(\overline{F})$ is the set of $\sqsubseteq$-minimal elements in $\pi_r(\mathcal{L}) \setminus \{0\}$. Then use Algorithm 2.5.1 to compute *exactly* the $\sqsubseteq$-minimal elements in $\mathcal{L} \setminus \{0\}$, that is, the Graver basis of $\mathcal{L}$.

Example 2.6.1 The *homogeneous primitive partition identities* of order n, see [114, Chapter 14] for details, correspond to the Graver basis elements of the matrix

$$\begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & 2 & 3 & \ldots & n \end{pmatrix}.$$

For $n = 20$, there exist $1,254,767$ Graver basis elements (and their negatives). This computation with the implementation in `4ti2` of this project-and-lift algorithm took about 30 days on a Sun Fire V890 Ultra Sparc IV processor with 1200 MHz. A more recent implementation by Matthias Walter reduces the computation time down to 8.25 days on the same machine. Note that both implementations did not exploit the special structure of the problem matrix. □

2.6.2 $\sqsubseteq$-minimal elements in $\mathcal{L}_+ \setminus \{0\}$

Here, the situation is slightly more complicated. Of course, one could extract these $\sqsubseteq$-minimal elements from the $\sqsubseteq$-minimal elements in $\mathcal{L} \setminus \{0\}$. This is clearly inefficient, as the latter set can be much bigger than the set we are looking for. Unfortunately, we cannot avoid such a behavior, but we can at least try to keep the set, from which we extract our desired $\sqsubseteq$-minimal elements, small.

We achieve this as follows. First we compute the set of $\sqsubseteq$-minimal elements in $\pi_r(\mathcal{L}) \setminus \{0\}$ and from it we extract the $\sqsubseteq$-minimal elements in $\pi_r(\mathcal{L})_+ \setminus \{0\}$. Now we use this set as $\pi_r(\overline{F})$ and the set $\pi_{r+1}(\mathcal{L}) \cap \{x : x_{\{1,\ldots,r\}} \geq 0\}$ instead of $\mathcal{L}_J$ with Algorithm 2.5.1. This computes the $\sqsubseteq$-minimal elements in $(\pi_{r+1}(\mathcal{L}) \cap \{x : x_{\{1,\ldots,r\}} \geq 0\}) \setminus \{0\}$ from which we can extract the $\sqsubseteq$-minimal elements in $\pi_{r+1}(\mathcal{L})_+ \setminus \{0\}$. Iterating this process with one new variable at a time, we obtain the $\sqsubseteq$-minimal elements in $\pi_{r+2}(\mathcal{L})_+ \setminus \{0\}$, …, $\pi_n(\mathcal{L})_+ \setminus \{0\} = \mathcal{L}_+ \setminus \{0\}$, and are done.

Clearly, in intermediate steps we still compute only supersets from which we throw away those elements with negative last component, but these differences in sizes are not as big as when we extract the $\sqsubseteq$-minimal elements in $\mathcal{L}_+ \setminus \{0\}$ from those in $\mathcal{L} \setminus \{0\}$.

Example 2.6.2 Let us consider the cone of magic 6×6 squares, see Chapter 8 for its definition. This cone is a 24-dimensional cone in $\mathbb{R}^{36}$. Using an implementation in `4ti2` of the project-and-lift algorithm above, the $522,347$ elements in the Hilbert basis were computed in about 10 days on a 1GHz PC with 4GB RAM running Linux. $\square$

Remark. This project-and-lift approach is of course applicable also for the computation of the $\sqsubseteq$-minimal elements in $\mathcal{L}_I = \{x \in \mathcal{L} : x_I \geq 0\}$, where $I \subseteq \{1, \ldots, n\}$. Moreover, we can use this approach to compute Hilbert bases of general cones $C = \{x : Ax \leq 0\}$. To see this, introduce slack variables u, define $\sqsubseteq_u$ via $(x_1, u_1) \sqsubseteq_u (x_2, u_2)$ if and only if $u_1 \sqsubseteq u_2$, and compute all $\sqsubseteq_u$-minimal elements in $\{(x,u) \in \mathbb{Z}^{n+d} : Ax + u = 0, u \geq 0\}$. Note that $x \in \mathbb{Z}^n$ belongs to the Hilbert basis of C if and only if $(x, -Ax)$ is $\sqsubseteq_u$-minimal in $\{(x,u) \in \mathbb{Z}^{n+d} : Ax + u = 0, u \geq 0\}$.

Finally, we wish to remark here that there are also other algorithms [4, 33, 100, 103] to compute the $\sqsubseteq$-minimal elements in the case that $\mathcal{L} = \ker(A)$ for some integer matrix A. Our approach, however, is faster and more general. $\square$

Chapter 3

Test sets

In this chapter we deal with the solution of integer linear programs via a simple augmentation algorithm. In fact, we consider the optimization problem:

$$(\mathcal{L})_{c,\beta}: \qquad \min\{c^\intercal z : z \equiv \beta \pmod{\mathcal{L}}, z \in \mathbb{Z}_+^n\}.$$

Herein, $\mathcal{L}$ denotes a sublattice of $\mathbb{Z}^n$, and we have $\beta \in \mathbb{Z}^n$ and $c \in \mathbb{R}^n$. For given matrix $A \in \mathbb{Z}^{d\times n}$ and $\mathcal{L} = \{x \in \mathbb{Z}^n : Ax = 0\}$, this general problem simplifies to the "usual" integer linear program

$$(\mathrm{IP})_{c,b}: \qquad \min\{c^\intercal z : Az = b, z \in \mathbb{Z}_+^n\}$$

for a certain vector $b \in \mathbb{Z}^d$.

Although $(\mathcal{L})_{c,\beta}$ denotes a family of problems as c and β vary, we refer to subsets of $(\mathcal{L})_{c,\beta}$ as $(\mathcal{L})_{c,\beta}$ as well, but state which data is kept fixed and which is allowed to vary. Thus, a single instance, that is where β and c are given, is also denoted by $(\mathcal{L})_{c,\beta}$. The family $(\mathrm{IP})_{c,b}$ is handled analogously. Note that the problem $(\mathcal{L})_{c,\beta}$ has a close connection for example to the "group problem in integer programming" [106] and to the integer Fourier-Motzkin elimination [121, 122, 123], respectively. $\square$

Primal methods are based on repeated augmentation of a feasible solutions to $(\mathcal{L})_{c,\beta}$ to optimality. Strongly related to this augmentation scheme is the notion of a test set. For a recent survey on test sets see for example [120].

Definition 3.0.1 (Test Set) *A set $\mathcal{T} \subseteq \mathbb{Z}^n$ is called a* **test set** *for $(\mathcal{L})_{c,\beta}$ if*

1. *$c^\intercal t > 0$ for all $t \in \mathcal{T}$, and*
2. *for every $\beta \in \mathbb{Z}^n$ and for every non-optimal feasible point z_0 of $(\mathcal{L})_{c,\beta}$ there exists a vector $t \in \mathcal{T}$ such that $z_0 - t$ is feasible.*

A vector $t \in \mathcal{T}$ satisfying these two conditions is called an **improving vector** *or an* **improving direction**.

A set is called a **universal test set** *for $(\mathcal{L})_{c,\beta}$ if it contains a test set for $(\mathcal{L})_{c,\beta}$ for every cost vector $c \in \mathbb{R}^n$.*

Clearly, a universal test set can only depend on the given lattice $\mathcal{L}$ as all the other data is allowed to vary. Moreover, it should be noted that a test set is a subset of the lattice $\mathcal{L}$, since subtracting a test set element t from a feasible solution z must lead to another feasible solution $z - t \equiv \beta \pmod{\mathcal{L}}$.

Once a finite (universal) test set $\mathcal{T}$ for and a feasible solution z_0 to $(\mathcal{L})_{c,\beta}$ are available, the following augmentation algorithm can be employed in order to solve the optimization problem $(\mathcal{L})_{c,\beta}$.

Algorithm 3.0.2 (Augmentation Algorithm)

```
Input: $\mathcal{L}$, $c$, $\beta$, a finite test set $\mathcal{T}$ for $(\mathcal{L})_{c,\beta}$, a feasible solution $z_0$ to $(\mathcal{L})_{c,\beta}$
Output: an optimum $z_{\min}$ of $(\mathcal{L})_{c,\beta}$
while there is $t \in \mathcal{T}$ with $c^\intercal t > 0$ such that $z_0 - t$ is feasible do
    $z_0 := z_0 - t$
return $z_0$
```

Thus, the process of optimizing $(\mathcal{L})_{c,\beta}$ via (universal) test sets can be decomposed as follows:

1. Compute a generating set of $\mathcal{L}$ over $\mathbb{Z}$.
2. Compute a (universal) test set for $(\mathcal{L})_{c,\beta}$.
3. Find an initial solution z_0 to $(\mathcal{L})_{c,\beta}$.
4. Augment z_0 until an optimum $z_{\min}$ is reached.

In the following, we deal with each of these algorithmic questions in more detail.

3.1 How do we find lattice generators?

In this section, we deal with the question of how to find lattice generators if the lattice is given only implicitly as $\mathcal{L} = \{z \in \mathbb{Z}^n : Az = 0, Bz \equiv 0 \pmod{p}\}$, where $Bz \equiv 0 \pmod{p}$ is a short-hand notation for relations $B_i^\intercal z \equiv 0 \pmod{p_i}$ for given integers p_i.

Let us first reduce the problem to one where the lattice has the form $\{z' \in \mathbb{Z}^k : A'z' = 0\}$ for some integer matrix A'. For this, we write each relation $B_i^\intercal z \equiv 0 \pmod{p_i}$ in the equivalent form $B_i^\intercal z - \gamma_i p_i = 0$ for some integer γ_i. Then it can be easily seen that once we project the elements of a generating set of

$$\{(z,\gamma) \in \mathbb{Z}^k : Az = 0, Bz - \operatorname{diag}(p)\gamma = 0\} = \left\{ \begin{pmatrix} A & 0 \\ B & -\operatorname{diag}(p) \end{pmatrix} \begin{pmatrix} z \\ \gamma \end{pmatrix} \right\} =: \{z' \in \mathbb{Z}^k : A'z' = 0\}$$

over $\mathbb{Z}$ onto the first n components (that is, we throw away again the γ_i's), we get a generating set of $\mathcal{L}$ over $\mathbb{Z}$, as desired.

Thus, without loss of generality, we can now assume that $\mathcal{L} = \{z \in \mathbb{Z}^n : Az = 0\}$. Moreover, we may assume that the $d \times n$ matrix A has full row rank d, a property that is easily checked or achieved by computing the row-echelon form of A. We now bring the matrix A into a special form, the *Hermite normal form* $\mathrm{HNF}(A)$. This matrix $\mathrm{HNF}(A)$ is the uniquely determined $d \times n$ matrix $(D|0)$ that one gets from A by doing only integer column operations (change of sign, addition of integer multiple of a column to another column, interchanging two columns) such that D is a lower-triangular matrix with strictly positive entries on the diagonal and all entries d_{ij} of D with $j < i$ are non-negative and strictly smaller than the element d_{ii} of the diagonal of D in the same row.

Thus, we can write $\mathrm{HNF}(A) = AU_1 \dots U_s$, where the U_i are $n \times n$ matrices that encode the column operations performed on A. Note that we get the matrix $U := I_n U_1 \dots U_s$ from the identity matrix I_n by performing the same column operations on I_n as on A.

Lemma 3.1.1 *Let* $\mathrm{HNF}(A) = AU$ *and let* $u_1, \dots, u_n$ *denote the columns of* U. *Then the last* $n-d$ *columns of* U, $u_{d+1}, \dots, u_n$, *generate* $\mathcal{L} = \ker_{\mathbb{Z}^n}(A)$ *over* $\mathbb{Z}$.

Proof. Computing $AU = \mathrm{HNF}(A) = (D|0)$, we already see that $Au_i = 0$ for $i = d+1, \dots, n$, and thus $u_{d+1}, \dots, u_n \in \ker_{\mathbb{Z}^n}(A)$.

Let $z \in \ker_{\mathbb{Z}^n}(A)$ and consider $0 = Az = (AU)(U^{-1}z) = \mathrm{HNF}(A)y = (D|0)y$ with $y = U^{-1}z$. By construction of U, $\det(U) = \pm 1$, and thus U^{-1} contains only integer entries. Consequently, $y \in \ker_{\mathbb{Z}^n}(\mathrm{HNF}(A))$. Due to the special structure of $\mathrm{HNF}(A)$, that is, since the columns of D are linearly independent, $\ker_{\mathbb{Z}^n}(\mathrm{HNF}(A))$ is generated over $\mathbb{Z}$ by the unit vectors $e_{d+1}, \dots, e_n$. We conclude that $z = Uy$ is an integer linear combination of the columns $u_{d+1}, \dots, u_n$ of U and the claim is proved. $\square$

Example 3.1.2 Consider the optimization problem

$$\min \left\{ (1,0,1,0)z : \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 10 & 25 \end{pmatrix} z = \begin{pmatrix} 10 \\ 110 \end{pmatrix}, z \in \mathbb{Z}^4_+ \right\}.$$

Herein, we have $A = \left(\begin{smallmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 10 & 25 \end{smallmatrix}\right)$. Let us compute

- the Hermite normal form $\mathrm{HNF}(A) = (D|0)$ of A and
- the transformation matrix U with $\mathrm{HNF}(A) = AU$,

from which we can extract a lattice basis F for $\ker_{\mathbb{Z}^4}(A)$.

$$\left(\begin{smallmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 10 & 25 \end{smallmatrix}\right) \rightarrow \left(\begin{smallmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 9 & 24 \end{smallmatrix}\right) \rightarrow \left(\begin{smallmatrix} 1 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 \end{smallmatrix}\right) \rightarrow \left(\begin{smallmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 4 & 0 \end{smallmatrix}\right) \rightarrow \left(\begin{smallmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{smallmatrix}\right)$$

Thus,

$$\mathrm{HNF}(A) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Applying the same column operations to I_4 we obtain

$$U = \begin{pmatrix} 0 & 1 & -5 & 5 \\ 2 & -2 & 9 & -6 \\ -1 & 1 & -4 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The last two columns of $\mathrm{HNF}(A)$ are zero. Thus, the last two columns of U,

$$F = \{(-5, 9, -4, 0), (5, -6, 0, 1)\},$$

(here written as row vectors) form a lattice basis of $\ker_{\mathbb{Z}^4}(A)$. □

3.2 Graver bases: A universal test set

In this section, we have another look onto a basic object of this book, the *Graver basis*. The Graver basis turns out to be a universal test set for the problem class $(\mathcal{L})_{c,\beta}$. We now give three different equivalent characterizations of Graver bases. Corollaries 2.2.4 and 2.2.5 imply that these characterizations are indeed equivalent.

Definition 3.2.1 *For given sublattice $\mathcal{L}$ of $\mathbb{Z}^n$, we define the* **Graver basis** *$\mathcal{G}_{IP}(\mathcal{L})$ as follows:*

1. *$\mathcal{G}_{IP}(\mathcal{L})$ is the unique inclusion-minimal set having the positive sum property with respect to $\mathcal{L}$.*
2. *$\mathcal{G}_{IP}(\mathcal{L})$ is the set of all $\sqsubseteq$-minimal elements of $\mathcal{L} \setminus \{0\}$.*
3. *For every orthant $\mathbb{O}_j$ of $\mathbb{R}^n$ let H_j denote the unique inclusion-minimal generating set of the monoid $\mathbb{O}_j \cap \mathcal{L}$. Then $\mathcal{G}_{IP}(\mathcal{L})$ is the union of these H_j taken over all 2^n orthants $\mathbb{O}_j$ of $\mathbb{R}^n$.*

Note that Chapter 2 already provided algorithms to compute this Graver basis once generators of the underlying lattice $\mathcal{L}$ are known.

Originally, Graver [65] defined this set via the third condition and only for a special type of lattices, $\mathcal{L} = \ker_{\mathbb{Z}^n}(A) := \{v \in \mathbb{Z}^n : Av = 0\}$, the integer kernel of a given matrix A.

Example 3.1.2 cont. In order to compute the Graver basis of A, perform the completion procedure, Algorithm 2.4.1, or use the software package `4ti2` [70] as follows. Store the problem matrix in a file, say “4coins”:

```
2 4
1 1  1  1
1 5 10 25
```

and then call “graver 4coins” which produces a file named “4coins.gra”:

```
5 4
-5 6  0 -1
-5 9 -4  0
 0 3 -4  1
-5 3  4 -2
-5 0  8 -3
```

containing the desired 5 Graver basis elements. As the Graver basis is symmetric, that is, if v is a Graver basis element so is $-v$, we obtain as the Graver basis of A the 10 vectors:

$$\{\pm(-5,6,0,-1), \pm(-5,9,-4,0), \pm(0,3,-4,1), \pm(-5,3,4,-2), \pm(-5,0,8,-3)\}$$

□

The next lemma shows that it is in fact the positive sum property with respect to $\mathcal{L}$ that turns the Graver basis $\mathcal{G}_{\mathrm{IP}}(\mathcal{L})$ into a universal test set.

Lemma 3.2.2 (Positive Sum Property implies Universal Test Set Property) *If $G \subseteq \mathbb{Z}^n$ has the positive sum property with respect to the sublattice $\mathcal{L}$ of $\mathbb{Z}^n$, then G is a universal test set for $(\mathcal{L})_{c,\beta}$.*

Proof. Given $\beta \in \mathbb{Z}^n$, $c \in \mathbb{R}^n$, and a non-optimal feasible point z_0 of $(\mathcal{L})_{c,\beta}$. We have to prove that there exists a vector $v \in G$ such that $z_0 - v$ is a better feasible solution than z_0.

By z_1 denote another feasible solution with smaller cost function value than z_0. Thus, by the assumptions on G, we can write $z_0 - z_1 \in \mathcal{L}$ as a finite integer linear combination $z_0 - z_1 = \sum \alpha_i g_i$, where $g_i \in G$, $\alpha_i \in \mathbb{Z}_{>0}$, and $\alpha_i g_i \sqsubseteq z_0 - z_1$ for all i. Since $0 < c^\intercal(z_0 - z_1) = \sum \alpha_i c^\intercal g_i$ we have $c^\intercal g_j > 0$ for at least one g_j. We claim that $z_0 - g_j$ is feasible and has a better cost function value than z_0.

Clearly, $z_0 - g_j \equiv \beta - 0 \equiv \beta \pmod{\mathcal{L}}$ and $c^\intercal z_0 > c^\intercal(z_0 - g_j)$, since $g_j \in \mathcal{L}$ and $c^\intercal g_j > 0$. So it remains to prove that $z_0 - g_j \geq 0$. But this follows immediately from Lemma 2.1.4, which states that the components of $z_0 - g_j \geq 0$ lie in the intervals defined by the corresponding (non-negative!) components of z_0 and of z_1. □

Corollary 3.2.3 *The Graver basis $\mathcal{G}_{IP}(\mathcal{L})$ is a universal test set for $(\mathcal{L})_{c,\beta}$.*

Remark. It should be noted that in general a Graver basis does *not* constitute an inclusion-minimal universal test set for $(\mathcal{L})_{c,\beta}$. There is in fact a unique inclusion-minimal universal test set, a *universal Gröbner basis* of the underlying lattice ideal. We refer to [114] for more details. □

3.3 How do we find an initial solution?

Now that we have computed a lattice basis for $\mathcal{L}$, we can apply one of the algorithms presented in Chapter 2 to compute a universal test set for $(\mathcal{L})_{c,\beta}$, the Graver basis. But how can we find an initial solution z_0 for a given problem

$$\min\{c^\intercal z : z \equiv \beta \pmod{\mathcal{L}}, z \in \mathbb{Z}_+^n\}.$$

from which we can start the augmentation algorithm? As it turns out, universal test sets allow a procedure similar to Phase I of the simplex method in linear programming: first we find a solution $z_0 \in \mathbb{Z}^n$ to $z \equiv \beta \pmod{\mathcal{L}}$ and then the negative components of z_0 are iteratively "removed" by subtracting suitable vectors from the universal test set.

An integer solution to $z \equiv \beta \pmod{\mathcal{L}}$ is quickly found: $z_0 = \beta$. However, how do we find it when $z \equiv \beta \pmod{\mathcal{L}}$ is given as $\{z \in \mathbb{Z}^n : Az = a, Bz \equiv b \pmod{p}\}$? Again, as in Section 3.1 we can reduce our attention to the case $\{z \in \mathbb{Z}^n : Az = a\}$. Now compute HNF$(A)$ and U with HNF$(A) = AU$, again as in Section 3.1.

From $a = Az = (AU)(U^{-1}z) = Hy = (D|0)y$, $y = U^{-1}z$, we first compute a solution y_0 to $Hy = a$. Such a solution y_0 is quickly found, as D is a triangular matrix. Moreover, again since D is a triangular matrix, any two solutions to $Hy = a$ differ only in their last $n-d$ components by a linear (integer) combination of $e_{d+1}, \ldots, e_n$. Thus, there is an integer solution z to $Az = a$ if and only if there is an integer solution $y_0 = U^{-1}z_0$ to $Hy = a$. (Remember that U^{-1} is an integer matrix!) Thus, if there is no integer solution y_0 to $Hy = a$, our optimization problem is infeasible. On the other hand, once we have found an integer solution y_0 with $Hy_0 = a$, we easily find the integer solution $z_0 = Uy_0$ with $Az = a$.

It remains to transform this integer solution z_0 of $Az = a$ into a *non-negative* integer solution of $Az = a$. This is done as follows: while there is some t in the universal test set such that $\|(z_0 - t)^-\|_1 < \|z_0^-\|_1$, replace z_0 by $z_0 - t$ and repeat.

Clearly, this process terminates after finitely many steps. Either we have found a feasible (non-negative, integer) solution to $Az = a$, or some components of z_0 are still negative. We claim that in the latter case, the equations $Az = a$ do not have a non-negative integer solution.

Lemma 3.3.1 *If $z_0 \notin \mathbb{Z}_+^n$ is an integer solution of $Az = a$ such that there is no $t \in \mathcal{G}_{IP}(A)$ with $\|(z_0 - t)^-\|_1 < \|z_0^-\|_1$, then $\{z : Az = a, z \in \mathbb{Z}_+^n\}$ is empty.*

Proof. Assume on the contrary that there was a solution $z_1 \in \mathbb{Z}_+^n$ of $Az_1 = a$. Then with $I := \{i : z_0^{(i)} < 0\}$, z_0^+ and $z_1 + z_0^-$ are both solutions to the minimization problem

$$\min\left\{\sum_{i\in I} -z^{(i)} : z \equiv \beta + z_0^- \pmod{\mathcal{L}}, z \in \mathbb{Z}_+^n\right\}.$$

While z_0^+ has objective value $\sum_{i\in I} -(z_0^+)^{(i)} = 0$ by construction of I, the objective value of $z_1 + z_0^-$ is strictly negative, since $(z_1)_I \geq 0$ and since at least one component of $(z_0^-)_I$ is strictly positive by our assumption $z_0 \not\geq 0$. Thus, there is some element t in the universal test set such that $z_0^+ - t$ is a feasible (non-negative) solution with strictly negative objective value

$$\sum_{i \in I} -(z_0^+ - t)^{(i)} = \sum_{i \in I} -(-t)^{(i)} < 0.$$

By adding $\sum_{i \in I} -z_0^{(i)}$ to both sides, we have $\sum_{i \in I} -(z_0 - t)^{(i)} < \sum_{i \in I} -z_0^{(i)}$. However, since $(z_0 - t)^{(i)} \geq 0$ and $z_0^{(i)} \geq 0$ for all $i \notin I$, this is equivalent to $\|(z_0 - t)^-\|_1 < \|z_0^-\|_1$, contradicting our assumption on z_0 that no such t exists. $\square$

Example 3.1.2 cont. To find an integer solution to $Az = \begin{pmatrix} 10 \\ 110 \end{pmatrix}$, we first solve $Hy = \begin{pmatrix} 10 \\ 110 \end{pmatrix}$, which gives for example $y_0 = (10, 110, 0, 0)^\intercal$ as an integer solution. Next we compute $z_0 = Uy_0 = (110, -200, 100, 0)^\intercal$. A quick check verifies $Az_0 = \begin{pmatrix} 10 \\ 110 \end{pmatrix}$.

The vector $(5, -9, 4, 0)^\intercal$ reduces the negative component while keeping the nonnegative components nonnegative. Note that the vector $(5, -1, 0, 1)^\intercal$, however, would also be applicable although new negative components appear. The important fact is that the norm of the negative components becomes smaller.

$$z_0 \rightarrow z_0 := z_0 - 22 \cdot (5, -9, 4, 0)^\intercal = (0, -2, 12, 0)^\intercal$$

Now we can subtract the vector $(0, -3, 4, -1)^\intercal$ and get

$$z_0 \rightarrow z_0 := z_0 - 1 \cdot (0, -3, 4, -1)^\intercal = (0, 1, 8, 1)^\intercal,$$

a feasible solution to the optimization problem. Again, we check that indeed $Az_0 = \begin{pmatrix} 10 \\ 110 \end{pmatrix}$. $\square$

3.4 Does the augmentation process always terminate?

Having found an initial solution z_0 to our minimization problem

$$\min\{c^\intercal z : z \equiv \beta \pmod{\mathcal{L}}, z \in \mathbb{Z}_+^n\}.$$

and having a universal test sets for $(\mathcal{L})_{c,\beta}$ at hand as well, we can apply the augmentation procedure to iteratively improve z_0. There are two natural questions to ask:

- Does the augmentation procedure always terminate if the optimization problem is solvable?
- How can we determine whether the optimization problem is unbounded with respect to the objective function?

To answer the first question, we observe that there are only finitely many different vectors in the universal test set. Thus, in each augmentation step the objective value is decreased by the (well-defined and positive) minimum improvement of all these finitely many vectors. If the augmentation procedure did not terminate, the objective value would drop below any given number.

Let us now assume for the second question that the objective function is not bounded with respect to the objective function. We claim that already a (not necessarily universal) test set provides us with the information that the problem is unbounded.

Lemma 3.4.1 *Let T be a test set for $(\mathcal{L})_{c,\beta}$. The problem $(\mathcal{L})_{c,\beta}$ is unbounded with respect to the objective function if and only if there exists $t \in T$ being feasible for $(\mathcal{L})_{c,0}$ with $c^\intercal t > 0$.*

Proof. Note that the problem $(\mathcal{L})_{c,\beta}$ is unbounded with respect to the objective function if and only if there exists a feasible integer solution z in $(\mathcal{L})_{c,0}$ with $c^\intercal z < 0$. On the other hand, 0 is not optimal for the problem $(\mathcal{L})_{c,0}$ if and only if there exists $t \in T$ with $c^\intercal t > 0$ such that $0 - t$ is a better feasible solution than 0. □

Example 3.1.2 cont. Finally, we want to augment the feasible solution $z_0 = (0, 1, 8, 1)$ (written as a row vector for better readability) to optimality. In a first step, however, we extract those vectors from the Graver basis that have a strictly positive objective value, as only those could possibly be applied as improving directions:

$$\begin{array}{rccc}
(5,-6,0,1): & (5,0,0,1) & \rightarrow & (0,6,0,0)\\
(5,-9,4,0): & (5,0,4,0) & \rightarrow & (0,9,0,0)\\
(0,-3,4,-1): & (0,0,4,0) & \rightarrow & (0,3,0,1)\\
(5,-3,-4,2): & (5,0,0,2) & \rightarrow & (0,3,4,0)\\
(-5,0,8,-3): & (0,0,8,0) & \rightarrow & (5,0,0,3)
\end{array}$$

We would like to point out here, that not all 5 improving directions are needed as a test set for the given cost vector. Already the two vectors

$$\begin{array}{rccc}
(5,-6,0,1): & (5,0,0,1) & \rightarrow & (0,6,0,0)\\
(0,-3,4,-1): & (0,0,4,0) & \rightarrow & (0,3,0,1)
\end{array}$$

would suffice. To convince ourselves of this fact, note that for example if $(5, 0, 4, 0) \rightarrow (0, 9, 0, 0)$ is applicable, then so is $(0, 0, 4, 0) \rightarrow (0, 3, 0, 1)$. The Graver basis, being a universal test set, usually contains many vectors that are superfluous once only a fixed objective function is considered.

Now we can augment z_0 (for example) as follows:

$$\begin{array}{lcl}
(0,1,8,1) & \rightarrow & (0,1,8,1) - (-5,0,8,-3) = (5,1,0,4)\\
(5,1,0,4) & \rightarrow & (5,1,0,4) - (5,-6,0,1) = (0,7,0,3)
\end{array}$$

The solution $(0, 7, 0, 3)$ is optimal with objective value 0. One reason to convince us is the fact that no test set vector can improve it, another is that the objective function is bounded from below by 0 and this value is reached. Thus, $z_{\min} = (0, 7, 0, 3)$ is an optimal solution. □

3.5 How many augmentation steps are needed?

Let us assume that we have found a feasible solution z_0 to the problem $(\mathcal{L})_{c,\beta}$ and assume that we were able to decide that the optimal value of $(\mathcal{L})_{c,\beta}$ is finite. The iterative augmentation procedure, Algorithm 3.0.2 using test sets can now be applied, since test sets allow a solution to the Augmentation problem:

Augmentation problem. Given $\mathcal{L} \subseteq \mathbb{Z}^n$, $z_0 \in \mathbb{Z}^n_+$ and $c \in \mathbb{Z}^n$, either find an improving direction $g \in \mathbb{Z}^n$, namely one with $z - g \in \{y \in \mathbb{Z}^n_+ : y \equiv z_0 \pmod{\mathcal{L}}\}$ and $c^\intercal g > 0$, or assert that no such g exists.

A major remaining question is how many augmentation steps are needed to reach an optimal solution. In the following, we give an answer to this last remaining question.

An *augmentation oracle for a lattice* $\mathcal{L}$ is one that solves the augmentation problem, that is, when queried on $z_0 \in \mathbb{Z}^n_+$ and $c \in \mathbb{Z}^n$, it either returns an improving direction g or asserts that none exists. Recently, in [108], a directed version of the augmentation problem was introduced; quite remarkably, it was shown that the number of *directed* augmentation steps needed to reach optimality is polynomial. We discuss this next. First, we define the directed augmentation problem.

Directed Augmentation Problem. Given $c_1, c_2 \in \mathbb{Q}^n$ and $z_0 \in \{y \in \mathbb{Z}^n_+ : y \equiv \beta \pmod{\mathcal{L}}\}$, find a vector $g = g^+ - g^- \in \mathbb{Z}^n$ such that $z_0 - g \in \{y \in \mathbb{Z}^n_+ : y \equiv \beta \pmod{\mathcal{L}}\}$ and $c_1^\intercal g^+ - c_2^\intercal g^- > 0$, or decide that no such vector g exists.

Thus, the directed augmentation problem involves *two* objective function vectors: c_1 controls the cost of the positive part of g and c_2 controls the cost of the negative part of g. The usual augmentation problem occurs as the special case $c_1 = c_2 = c$. A *directed augmentation oracle for* $\mathcal{L}$ is one that solves the directed augmentation problem, i.e. when queried on $z_0 \in \mathbb{Z}^n_+$, $c_1, c_2 \in \mathbb{Z}^n$, it either returns an improving direction g or asserts that none exists.

In [108], it was assumed that the input includes an upper bound vector $u \in \mathbb{Z}^n_+$ on the variables, so that the actual feasible set is $\{z \in \mathbb{Z}^n_+ : z \equiv \beta \pmod{\mathcal{L}}, z \leq u\}$. Under this assumption, the feasible region is always bounded and there is always an optimal solution. Further, the complexity estimates in [108] depended on the bit size of u and β. However, this is not really needed. If the integer program $(\mathcal{L})_{c,\beta}$ with $\beta = z_0$ is solvable, it has an optimal solution whose bit size is polynomially bounded in the size of (given generators of) $\mathcal{L}$ and $\beta = z_0$ Moreover, such bounds u are polynomial time computable. This basically follows from Cramer's rule, see e.g. [106, Section 17.1]. Therefore, it is possible to compute an upper bound u in terms of (given generators of) $\mathcal{L}$ and β only, and plug it into the analysis of [108].

With this, the results of [108] imply the following.

Theorem 3.5.1 (Schulz, Weismantel [108]) *There is a polynomial oracle time algorithm that, given (generators of)* $\mathcal{L} \subseteq \mathbb{Z}^n$, $z_0 \in \mathbb{Z}^n_+$, $c \in \mathbb{Z}^n$, *solves the integer program* $(\mathcal{L})_{c,\beta}$ *with* $\beta = z_0$ *by querying a directed augmentation oracle for* $\mathcal{L}$.

Here, as usual, *solving* the (feasible) integer program means that the algorithm either returns an optimal solution or asserts that the objective function is unbounded; and *polynomial oracle time* means that the number of arithmetic operations, the number of calls to the oracle, and the size of the numbers occurring throughout the algorithm are polynomially bounded in the size of the input: (generators of) $\mathcal{L}$, $\beta = z_0$, and c.

As we show now, the Graver basis of a lattice $\mathcal{L}$ yields not only an augmentation oracle for $\mathcal{L}$, but it enables the realization of a *directed* augmentation oracle for $\mathcal{L}$ as well.

Lemma 3.5.2 *Let $\mathcal{G}_{IP}(\mathcal{L})$ be the Graver basis of $\mathcal{L} \subseteq \mathbb{Z}^n$. For any $z_0 \in \mathbb{Z}_+^n$ and $c_1, c_2 \in \mathbb{Z}^n$, there is a $g \in \mathbb{Z}^n$ with $z_0 - g \in \{y \in \mathbb{Z}_+^n : y \equiv z_0 \pmod{\mathcal{L}}\}$ and $c_1^\intercal g^+ - c_2^\intercal g^- > 0$ if and only if there is such $g \in \mathcal{G}_{IP}(\mathcal{L})$.*

Proof. Let v be a suitable vector for the Directed Augmentation Problem. Then v can be written as $v = \sum_{i=1}^s g_i$ with (not necessarily distinct) vectors $g_i \in \mathcal{G}_{\mathrm{IP}}(\mathcal{L})$ and $g_i \sqsubseteq v$ for all i. Thus, $(v^+, v^-) = \sum_{i=1}^s (g_i^+, g_i^-)$ and consequently

$$0 < c_1^\intercal v^+ - c_2^\intercal v^- = \sum_{i=1}^s c_1^\intercal g_i^+ - c_2^\intercal g_i^- .$$

This implies that there is some g_j in this sum with $c_1^\intercal g_i^+ - c_2^\intercal g_i^- > 0$. We claim that also $z_0 - g_j \in \{y \in \mathbb{Z}_+^n : y \equiv z_0 \pmod{\mathcal{L}}\}$ and thus also the Graver basis element g_j is a suitable vector for the Directed Augmentation Problem.

Clearly, as $g_j \in \mathcal{G}_{\mathrm{IP}}(\mathcal{L})$, we have $g_j \in \mathcal{L}$ and thus $z_0 - g_j \equiv z_0 \pmod{\mathcal{L}}$. It remains to show that $z_0 - g_j \geq 0$. But this follows immediately from Lemma 2.1.4, which states that the components of $z_0 - g_j \geq 0$ lie in the intervals defined by the corresponding (non-negative!) components of z_0 and of $z_0 - g$. $\square$

As an immediate corollary of Theorem 3.5.1 and Lemma 3.5.2, we get the following statement.

Theorem 3.5.3 *There is a polynomial time algorithm that, given generators of a lattice $\mathcal{L} \subseteq \mathbb{Z}^n$ along with its Graver basis $\mathcal{G}_{IP}(\mathcal{L})$, and vectors $z_0 \in \mathbb{Z}_+^n$ and $c \in \mathbb{Z}^n$, solves the integer program $(\mathcal{L})_{c,\beta}$ with $\beta = z_0$.*

While Theorem 3.5.3 holds for sublattice $\mathcal{L}$ of $\mathbb{Z}^n$, its complexity bound depends on the size of the Graver basis which is part of the input. Typically, the Graver basis is very large and its cardinality may be exponential in the dimension n. However, in the next chapter we show that for a broad and useful class of matrices, we can tame the behavior of the Graver basis.

Chapter 4

n-fold integer programming

It is well-known that the integer program $\min\{c^\intercal x : Ax = b, x \in \mathbb{Z}_+^q\}$, where A is an integer matrix and b, c are integer vectors of suitable dimensions, is generally NP-hard but polynomial time solvable in fixed dimension q, see [106]. In this chapter, motivated by applications to high-dimensional transportation problems and contingency tables and by the recently discovered striking universality theorem for rational polytopes [46] (see Section 4.1), we study the following class of integer programming problems in variable dimension.

The n-fold integer programming problem. Fix a $p \times q$ integer matrix A. Given positive integer n and integer vectors $b = (b^0, b^1, \ldots, b^n)$ and $c = (c^1, \ldots, c^n)$, where $b^0 \in \mathbb{Z}^q$, and $b^k \in \mathbb{Z}^p$ and $c^k \in \mathbb{Z}_+^q$ for $k = 1, \ldots, n$, find a non-negative integer vector $x = (x^1, \ldots, x^n)$, where $x^k \in \mathbb{Z}_+^q$ for $k = 1, \ldots, n$, which minimizes $c^\intercal x = \sum_{k=1}^n (c^k)^\intercal x^k$ subject to the equations $\sum_{k=1}^n x^k = b^0$ and $Ax^k = b^k$ for $k = 1, \ldots, n$.

The term "n-fold integer programming" refers to the problem being almost separable into n similar programs $\min\{{c^k}^\intercal x : Ax^k = b^k, x^k \in \mathbb{Z}_+^q\}$ in fixed dimension. The constraint $\sum_{k=1}^n x^k = b^0$, however, binds these programs together, and the result is an integer program in large variable dimension nq.

Let the *n-fold matrix* of A be the following $(q + np) \times nq$ matrix, with I_q the $q \times q$ identity matrix:

$$A^{(n)} := \begin{pmatrix} I_q & I_q & I_q & \cdots & I_q \\ A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A \end{pmatrix}.$$

Then the n-fold integer programming problem can be written conveniently in matrix form as

$$\min\{c^\intercal x : \ A^{(n)}x = b, x \in \mathbb{Z}_+^{nq}\}.$$

In this chapter we establish the following theorem. Naturally, the input size is n plus the bit size of the integer objective vector $c \in \mathbb{Z}^{nq}$ and the integer right-hand side vector $b \in \mathbb{Z}^{q+np}$.

Theorem 4.0.1 *Fix any integer matrix A. Then there is a polynomial time algorithm that, given any n and any integer vectors b and c, solves the corresponding n-fold integer programming problem.*

The proof of this theorem involves two heavy ingredients. First, it makes use of the equivalence of the linear optimization problem and the directed augmentation problem, recently introduced and studied in [108], see Section 3.5 for details. Second, it uses recent results of [79] and [105] on the stabilization of certain Graver bases.

One important consequence of Theorem 4.0.1 is a polynomial time algorithm for the 3-way transportation problem for long tables, settling its computational complexity; see Section 4.1 for details.

Corollary 4.1.2 Fix any r, s. Then there is a polynomial time algorithm that, given l, integer objective vector c, and integer line-sums $(u_{i,j})$, $(v_{i,k})$ and $(w_{j,k})$, solves the integer transportation problem

$$\min\{c^{\intercal}x : x \in \mathbb{Z}_{+}^{r\times s\times l}, \sum_i x_{i,j,k} = w_{j,k}, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\}.$$

The n-fold integer programming problem and theorem can be generalized as follows.

Generalized n-fold integer programming. Fix integer matrices A, B of sizes $r \times q$ and $s \times q$, respectively. Given positive integer n and integer vectors $b = (b^0, b^1, \ldots, b^n)$ and $c = (c^1, \ldots, c^n)$, with $b^0 \in \mathbb{Z}^s$, and $b^k \in \mathbb{Z}^r$ and $c^k \in \mathbb{Z}_+^q$ for $k = 1, \ldots, n$, find $x = (x^1, \ldots, x^n)$ with $x^k \in \mathbb{Z}_+^q$ for $k = 1, \ldots, n$, which minimizes $c^{\intercal}x = \sum_{k=1}^n c^k x^k$ subject to $\sum_{k=1}^n Bx^k = b^0$ and $Ax^k = b^k$ for $k = 1, \ldots, n$. Thus, the problem matrix is of the form

$$\begin{pmatrix} B & B & B & \cdots & B \\ A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A \end{pmatrix}.$$

We have the following more general result, from which Theorem 4.0.1 is deduced in the case $B = I_q$.

Theorem 4.0.2 *Fix any pair of integer matrices A, B of compatible sizes. Then there is a polynomial time algorithm that solves the generalized n-fold integer programming problem on any input n, b, c.*

This chapter is organized as follows. In Section 4.1 we discuss applications of Theorem 4.0.1 to multi-way transportation problems and to some packing problems, as follows. In Subsection 4.1.1 we obtain the aforementioned Corollary 4.1.2 which provides a polynomial time solution to 3-way integer transportation problems for long tables, contrasting the recent universality theorem of [46] for slim tables. We also extend this result to d-way transportation problems for long tables of any dimension (Corollary 4.1.4). In Subsection 4.1.2 we describe applications to a certain shipment problem (Corollary 4.1.5) and to the classical cutting stock problem (Corollary 4.1.6). In Section 4.2 we develop the remaining ingredients for our n-fold integer programming algorithm. We discuss the stabilization of Graver bases discovered recently in [79, 105], and use it to show that Graver bases of n-fold matrices can be computed in polynomial time (Theorem 4.2.2). In Section 4.3, we us this result together with the results from Section 3.5 to prove our main theorem of this chapter, Theorem 4.0.2, and its specialization Theorem 4.0.1 .

4.1 Applications

4.1.1 High dimensional transportation problems

A *d-way transportation polytope* is the set of all $m_1 \times \cdots \times m_d$ non-negative arrays $x = (x_{i_1,\dots,i_d})$ such that the sums of the entries over some of their lower dimensional subarrays (*margins*) are specified. For simplicity of exposition, we shall concentrate here only on d-way *line-sum polytopes*, of the form

$$T = \left\{ x \in \mathbb{R}_+^{m_1 \times \cdots \times m_d} : \sum_{i_1} x_{i_1,\dots,i_d} = u_{i_2,\dots,i_d}, \sum_{i_2} x_{i_1,\dots,i_d} = u_{i_1,i_3,\dots,i_d}, \dots, \sum_{i_d} x_{i_1,\dots,i_d} = u_{i_1,\dots,i_{d-1}} \right\}.$$

Transportation polytopes and their integer points (called contingency tables by statisticians), have been studied and used extensively in the operations research literature and in the context of secure statistical data disclosure by public agencies such as the census bureau and the national center for health statistics. In the operations research literature, one is typically interested in the integer and linear transportation problems, which are the integer and linear programming problems over the transportation polytope, see e.g. [12, 88, 99, 104, 119, 125] and references therein. In the statistics community, one is often interested in the values an entry can attain in all tables with fixed margins, related to the security of the entry under margin disclosure, and in the construction of a Markov basis allowing a random walk on the set of tables with fixed margins, see e.g. [8, 37, 38, 47, 51, 82] and references therein.

It is well known that the system defining a 2-way transportation polytope is totally unimodular. This implies that all the above problems are easy in this case. However, already 3-way transportation problems are much harder. Consider the problem of deciding if a given 3-way line-sum polytope of $r \times s \times l$ arrays (with r rows, s columns and l layers) contains an integer point: the computational complexity of this problem provides useful indication about the difficulty of the problems mentioned above. If r, s, l are all fixed, then the problem is solvable in polynomial time by integer programming in fixed dimension rsl. On the other hand, if r, s, l are all variable part of the input, then the problem is NP-complete [82]. The in-between cases are much more delicate. The case of two parameters r, s variable and one parameter l fixed was recently resolved in [45], where it was shown to be NP-complete, strengthening [82]. Moreover, very recently, in [46], the following striking universality result was shown.

Proposition 4.1.1 *Any rational polytope $P = \{y \in \mathbb{R}_+^n : Ay = b\}$ is polynomial time representable as a 3-way line-sum transportation polytope of size $r \times s \times 3$ for some (polynomially bounded) r*

and s,

$$T = \{x \in \mathbb{R}_{+}^{r\times s\times 3} : \sum_i x_{i,j,k} = w_{j,k}, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\}.$$

Here representable means that there is a coordinate-erasing projection from $\mathbb{R}^{r\times s\times 3}$ onto $\mathbb{R}^n$ providing a bijection between T and P and between the sets of integer points $T\cap\mathbb{Z}^{r\times s\times 3}$ and $P\cap\mathbb{Z}^n$. Thus, *any* rational polytope is an $r\times s\times 3$ line-sum polytope, and *any* integer (respectively, linear) programming problem is equivalent to an integer (respectively, linear) $r\times s\times 3$ line-sum transportation problem. This result led to the solution of several open problems from [119] and [125] and had several implications on the complexity of Markov bases and the entry security problem, see [46] and [47] for more details.

However, the last case, of two parameters r, s fixed and one parameter l variable, has remained open and intriguing. Here, as a consequence of Theorem 4.0.1, we are able to resolve this problem and show that both the decision and optimization problems are polynomial time solvable.

Corollary 4.1.2 *Fix any r, s. Then there is a polynomial time algorithm that, given l, integer objective vector c, and integer line-sums $(u_{i,j})$, $(v_{i,k})$ and $(w_{j,k})$, solves the integer transportation problem*

$$\min\{c^{\mathsf{T}}x : x \in \mathbb{Z}_{+}^{r\times s\times l}, \sum_i x_{i,j,k} = w_{j,k}, \sum_j x_{i,j,k} = v_{i,k}, \sum_k x_{i,j,k} = u_{i,j}\}.$$

Proof. We formulate the 3-way integer transportation problem as an n-fold integer program with $n := l$, $p := r+s$, and $q := r\cdot s$. Reindex the variables as $x^k_{i,j} := x_{i,j,k}$ so that the variables vector is $x = (x^1,\ldots,x^n)$ with $x^k = (x^k_{i,j}) \in \mathbb{Z}_{+}^{r\times s}$ a 2-way $r\times s$ table–the kth layer of the 3-way table x. Similarly write $c = (c^1,\ldots,c^n)$ with $c^k \in \mathbb{Z}^{r\times s}$ for the objective vector. Next, put $b := (b^0, b^1, \ldots, b^n)$, with $b^0 \in \mathbb{Z}_{+}^{rs}$ defined by $b^0 := (u_{i,j})$, and $b^k \in \mathbb{Z}_{+}^{r+s}$ defined by $b^k := ((v_{i,k}),(w_{j,k}))$ for $k = 1,\ldots,n$. Finally, let A be the $p\times q = (r+s)\times r\cdot s$ matrix of equations for the usual 2-way transportation polytope, forcing row-sums and column-sums on each of the $r\times s$ layers x^k by $Ax^k = b^k$, $k = 1,\ldots,n$. Then the equations $Ax^k = b^k$ force the line-sums $v_{i,k}$ and $w_{j,k}$, and the additional n-fold integer program binding constraint $\sum_{k=1}^n x^k = b^0$ forces the "long" line-sums $u_{i,j}$. This completes the encoding. Since r, s are fixed, so are p, q, and A, and therefore, the corollary follows from Theorem 4.0.1. □

Example 4.1.3 Consider the case $r = s = 3$ (the smallest where the problem is genuinely 3-dimensional). Then $p = 6$, $q = 9$, and writing $x^k = (x^k_{1,1}, x^k_{1,2}, x^k_{1,3}, x^k_{2,1}, x^k_{2,2}, x^k_{2,3}, x^k_{3,1}, x^k_{3,2}, x^k_{3,3})$, the matrix A which defines the n-fold program providing the formulation of the $3\times 3\times l$ transportation problem is

$$A = \begin{pmatrix} 1&1&1&0&0&0&0&0&0\\ 0&0&0&1&1&1&0&0&0\\ 0&0&0&0&0&0&1&1&1\\ 1&0&0&1&0&0&1&0&0\\ 0&1&0&0&1&0&0&1&0\\ 0&0&1&0&0&1&0&0&1 \end{pmatrix}.$$

Already for this case, of $3 \times 3 \times l$ tables, the only polynomial time algorithm for the corresponding line-sum integer transportation problem we are aware of is the one guaranteed by Corollary 4.1.2 above. $\square$

Corollary 4.1.2 extends to transportation problems of any dimension d, for *long* tables, namely, of size $m_1 \times \cdots \times m_{d-1} \times l$, where $m_1, \ldots, m_{d-1}$ are fixed and only the length l is variable, as follows.

Corollary 4.1.4 *Fix $d, m_1, \ldots, m_{d-1}$. Then there is a polynomial time algorithm that, given l, integer objective c, and line-sums $(u_{i_2,\ldots,i_d}), \ldots, (u_{i_1,\ldots,i_{d-1}})$, solves the long multi-way transportation problem*

$$\min\{c^\intercal x : x \in \mathbb{Z}_+^{m_1 \times \cdots \times m_{d-1} \times l} : \sum_{i_1} x_{i_1,\ldots,i_d} = u_{i_2,\ldots,i_d}, \ldots, \sum_{i_d} x_{i_1,\ldots,i_d} = u_{i_1,\ldots,i_{d-1}}\}.$$

Proof. The long multi-way transportation problem can be encoded as an n-fold integer program with $n := l$, $p := \sum_{i=1}^{d-1} m_i$, and $q := \prod_{i=1}^{d-1} m_i$, by reindexing the variables as $x^{i_d}_{i_1,\ldots,i_{d-1}} := x_{i_1,\ldots,i_d}$, letting A be the matrix of the equations of the line-sums of $(d-1)$-way transportation polytope of the $m_1 \times \cdots \times m_{d-1}$ arrays, and proceeding in direct analogy to the proof of Corollary 4.1.2. The details are omitted. $\square$

4.1.2 Some packing problems

Minimum cost shipment

The minimum cost shipment problem concerns the shipment of a large stock of items of several types, using various vessels, with minimum possible cost. More precisely, the data is as follows. There are t types of items. The weight of each item of type j is w_j and there are n_j items of type j to be shipped. There are v available vessels, where vessel k has maximum weight capacity u_k. The cost of shipping one item of type j on vessel k is $p_{j,k}$. We now formulate this as an n-fold integer programming problem. We set $n := v$, $p := 1$, $q := t + 1$. The defining matrix is the row vector $A = (A_j) := (w_1, w_2, \ldots, w_t, 1)$. The variables vector is $x = (x^1, \ldots, x^n)$ with $x^k = (x^k_1, \ldots, x^k_t, x^k_q)$, where x^k_j represents the number of items of type j to be shipped on vessel k for $j = 1, \ldots, t$, and x^k_q is an extra slack variable representing the unused weight capacity in vessel k. The cost vector is $c = (c^1, \ldots, c^n)$ with $c^k = (c^k_1, \ldots, c^k_t, c^k_q)$, where $c^k_j := p_{j,k}$ for $j = 1, \ldots, t$, and $c^k_q := 0$. Finally, the demand vector is $b = (b^0, b^1, \ldots, b^n)$ with $b^k := u_k$ for $k = 1, \ldots, n$, and $b^0 := (n_1, \ldots, n_t, \sum_{k=1}^v u_k - \sum_{j=1}^t n_j w_j)$. Then the resulting n-fold integer programming problem, $\min\{c^\intercal x : A^{(n)} x = b, x \in \mathbb{Z}_+^{nq}\}$, can be written in scalar form as follows:

$$\begin{array}{lll}
\min & \sum_{j=1}^{q}\sum_{k=1}^{n} c_j^k x_j^k = \sum_{j=1}^{t}\sum_{k=1}^{v} p_{j,k} x_j^k & \\
\text{s.t.} & \sum_{k=1}^{n} x_j^k = b_j^0 = n_j, & j = 1,\ldots,t, \\
& \sum_{k=1}^{n} x_q^k = b_q^0 = \sum_{k=1}^{v} u_k - \sum_{j=1}^{t} n_j w_j, & \\
& \sum_{j=1}^{q} A_j\, x_j^k = \sum_{j=1}^{t} w_j\, x_j^k + x_q^k = b^k = u_k, & k = 1,\ldots,n, \\
& x_j^k \in \mathbb{Z}_+, & j = 1,\ldots,q, k = 1,\ldots,n.
\end{array}$$

Assume that the number t of types is fixed, but the numbers n_j of items of each type may be very large: this is a reasonable assumption in applications (for instance, several types of automobiles to be shipped overseas, or several types of appliances to be shipped on ground). Then we obtain the following striking corollary of Theorem 4.0.1, showing that the problem is polynomial time solvable, where the input size is v plus the bit size of the integer numbers $n_j, w_j, u_k, p_{j,k}$ constituting the data. Note that this result is *much stronger* than the standard results on the *pseudo-polynomial time* solvability of this kind of packing and knapsack-type problems using dynamic programming: our algorithm can handle *very large* n_j and u_k, possibly exponential in the dimensional parameter v.

Corollary 4.1.5 *For each fixed number t of types, the minimum cost shipment problem is solvable in time which is polynomial in the number v of vessels and in the bit size of the integer numbers n_j of items of each type to be shipped, type weights w_j, vessel capacities u_k, and shipment costs $p_{j,k}$.*

The cutting stock problem

This is a classical manufacturing problem, where the usual setup is as follows: a manufacturer supplies rolls of material (such as scotch-tape or band-aid) in one of t different widths $w_1, \ldots, w_t$. The rolls are all cut out from standard rolls of common large width u, coming out of the production line. Given orders by customers for n_j rolls of width w_j, the problem facing the manufacturer is to meet the orders using the smallest possible number of standard rolls. This is almost a direct special case of the minimum cost shipment problem discussed above, with sufficiently many identical vessels, say $v := \sum_{j=1}^{t}\lceil n_j/\lfloor u/w_j\rfloor\rceil$, of capacity $u_k := u$ each, playing the role of the standard rolls, and with cost $p_{j,k} := w_j$ for each roll of width w_j regardless of the standard roll from which it is being cut out. The only correction needed is that each slack variable $x_q^k = x_{t+1}^k$, measuring the unused width of the kth standard roll, has cost of one unit instead of zero, so that the total cost becomes the number of standard rolls used. Thus the formulation as an n-fold program is with $n := \sum_{j=1}^{t}\lceil n_j/\lfloor u/w_j\rfloor\rceil$, $p := 1$, $q := t+1$, $A := (w_1, w_2, \ldots, w_t, 1)$, variables x_j^k representing the number of rolls of width w_j cut out of the kth roll for $j = 1, \ldots, t$ and x_q^k representing the unused width of the kth standard roll, costs $c_j^k := w_j$ for $j = 1, \ldots, t$ and $c_q^k := 1$, and demands $b^k := u$ for $k = 1, \ldots, n$ and $b^0 := (n_1, \ldots, n_t, nu - \sum_{j=1}^{t} n_j w_j)$.

Again, quite surprisingly, we get the following useful corollary regarding this classical problem.

Corollary 4.1.6 *For fixed number t of widths, the cutting stock problem is solvable in time polynomial in $\sum_{j=1}^{t}\lceil n_j/\lfloor u/w_j\rfloor\rceil$ and in the bit size of the numbers n_j of roll orders and roll widths w_j and u.*

One common approach to the cutting stock problem makes use of so-called *cutting patterns*, which are feasible solutions of the knapsack problem $\{y \in \mathbb{Z}_+^t : \sum_{j=1}^{t} w_j y_j \le u\}$. This is useful when the width u of the standard rolls is of the same order of magnitude as the demand widths w_j. However, when u is much larger than the w_j, the number of cutting patterns becomes prohibitively large to handle. But then the values $\lfloor u/w_j\rfloor$ are large and hence $n := \sum_{j=1}^{t}\lceil n_j/\lfloor u/w_j\rfloor\rceil$ is small, in which case the result of Corollary 4.1.6 using n-fold integer programming becomes particularly appealing.

4.2 Graver bases of n-fold matrices

Fix any pair of integer matrices A and B with the same number of columns, of dimensions $r \times q$ and $s \times q$, respectively. The *n-fold matrix of the ordered pair A, B* is the following $(s+nr)\times nq$ matrix,

$$[A,B]^{(n)} := \begin{pmatrix} B & B & B & \cdots & B \\ A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A \end{pmatrix}.$$

With this, the generalized n-fold integer programming problem can be conveniently written as

$$\min\{c^{\intercal}x : [A,B]^{(n)}x = b, x \in \mathbb{Z}_+^{nq}\}.$$

The n-fold of a single matrix A, defined in the introduction, is the special case $A^{(n)} = [A, I_q]^{(n)}$ with $B = I_q$ the $q \times q$ identity, giving the regular (non-generalized) n-fold integer programming problem.

We now discuss a recent result of [105] and its extension in [79] on the stabilization of Graver bases of n-fold matrices. Consider vectors $x = (x^1, \ldots, x^n)$ with $x^k \in \mathbb{Z}_+^q$ for $k = 1, \ldots, n$. The *type* of x is the number $|\{k : x^k \neq 0\}|$ of nonzero components $x^k \in \mathbb{Z}_+^q$ of x. The following result of [79] on the stabilization of Graver bases of $[A,B]^{(n)}$ extends the earlier result for $B = I_q$ from [105].

Proposition 4.2.1 *For every pair of integer matrices $A \in \mathbb{Z}^{r\times q}$ and $B \in \mathbb{Z}^{s\times q}$, there exists a constant $g(A,B)$ such that for all n, the Graver basis of $[A,B]^{(n)}$ consists of vectors of type at most $g(A,B)$.*

The smallest constant $g(A,B)$ possible in the proposition is called the *Graver complexity* of A, B. Using Proposition 4.2.1, we now show that $\mathcal{G}_{\mathrm{IP}}([A,B]^{(n)})$ can be computed in polynomial time.

Theorem 4.2.2 *Fix any pair of integer matrices $A \in \mathbb{Z}^{r\times q}$ and $B \in \mathbb{Z}^{s\times q}$. Then there is a polynomial time algorithm that, given n, computes the Graver basis $\mathcal{G}_{IP}([A,B]^{(n)})$ of the n-fold matrix $[A,B]^{(n)}$. In particular, the cardinality and the bit size of $\mathcal{G}_{IP}([A,B]^{(n)})$ are bounded by a polynomial function of n.*

Proof. Let $g := g(A,B)$ be the Graver complexity of A,B and consider any $n \geq g$. We show that the Graver basis of $[A,B]^{(n)}$ is the union of $\binom{n}{g}$ suitably embedded copies of the Graver basis of $[A,B]^{(g)}$. Consider any g indices $1 \leq k_1 < \cdots < k_g \leq n$ and define a map $\phi_{k_1,\ldots,k_g}$ from $\mathbb{Z}_+^{gq}$ to $\mathbb{Z}_+^{nq}$ by sending $x = (x^1,\ldots,x^g)$ to $y = (y^1,\ldots,y^n)$ defined by $y^{k_t} := x^t$ for $t = 1,\ldots,g$, and $y^k := 0$ for all other k.

We claim that the Graver basis of $[A,B]^{(n)}$ is the union of the images of the Graver basis of $[A,B]^{(g)}$ under the $\binom{n}{g}$ maps $\phi_{k_1,\ldots,k_g}$ for all $1 \leq k_1 < \cdots < k_g \leq n$, that is,

$$\mathcal{G}_{\mathrm{IP}}([A,B]^{(n)}) = \bigcup_{1\leq k_1<\cdots<k_g\leq n} \phi_{k_1,\ldots,k_g}(\mathcal{G}_{\mathrm{IP}}([A,B]^{(g)})). \qquad (4.2.1)$$

To see this, recall first that, by definition, the Graver basis of a matrix M is the set of all $\sqsubseteq$-minimal nonzero integer dependencies on M (where a dependency on M is a vector x satisfying $Mx = 0$). Thus, if $x = (x^1,\ldots,x^g) \in \mathcal{G}_{\mathrm{IP}}([A,B]^{(g)})$ then x is a $\sqsubseteq$-minimal nonzero dependency on $[A,B]^{(g)}$, implying that $\phi_{k_1,\ldots,k_g}(x)$ is a $\sqsubseteq$-minimal nonzero dependency on $[A,B]^{(n)}$ and hence $\phi_{k_1,\ldots,k_g}(x) \in \mathcal{G}_{\mathrm{IP}}([A,B]^{(n)})$. This establishes that the right-hand side of Equation (4.2.1) is contained in the left-hand side. Conversely, consider any $y \in \mathcal{G}_{\mathrm{IP}}([A,B]^{(n)})$. Then, by Proposition 4.2.1, the type of y is at most g, so there are indices $1 \leq k_1 < \cdots < k_g \leq n$ such that all nonzero components of y are among those of the reduced vector $x := (y^{k_1},\ldots,y^{k_g})$, and therefore $y = \phi_{k_1,\ldots,k_g}(x)$. Now, $y \in \mathcal{G}_{\mathrm{IP}}([A,B]^{(n)})$ implies that y is a $\sqsubseteq$-minimal nonzero dependency on $[A,B]^{(n)}$, and therefore x is a $\sqsubseteq$-minimal nonzero dependency on $[A,B]^{(g)}$ and hence $x \in \mathcal{G}_{\mathrm{IP}}([A,B]^{(g)})$, showing that $y \in \phi_{k_1,\ldots,k_g}(\mathcal{G}_{\mathrm{IP}}([A,B]^{(g)}))$. This establishes that the left-hand side of Equation (4.2.1) is contained in the right-hand side. Thus, the Graver basis of $[A,B]^{(n)}$ is indeed given by Equation (4.2.1).

Since A,B are fixed and hence $g = g(A,B)$ is constant, the g-fold matrix $[A,B]^{(g)}$ is also fixed and so the cardinality and bit size of its Graver basis $\mathcal{G}_{\mathrm{IP}}([A,B]^{(g)})$ are constant as well. It follows from Equation (4.2.1) that $|\mathcal{G}_{\mathrm{IP}}([A,B]^{(n)})| \leq \binom{n}{g}|\mathcal{G}_{\mathrm{IP}}([A,B]^{(g)})| \in O(n^g)$. Further, each element of $\mathcal{G}_{\mathrm{IP}}([A,B]^{(n)})$ is an nq-dimensional vector $\phi_{k_1,\ldots,k_g}(x)$ obtained from some $x \in \mathcal{G}_{\mathrm{IP}}([A,B]^{(g)})$ (of constant bit size) by appending zero components, and therefore is of linear bit size $O(n)$, showing that the bit size of the entire Graver basis $\mathcal{G}_{\mathrm{IP}}([A,B]^{(n)})$ is $O(n^{g+1})$. Finally, it is clear that the $\binom{n}{g} = O(n^g)$ images $\phi_{k_1,\ldots,k_g}(\mathcal{G}_{\mathrm{IP}}([A,B]^{(g)}))$ and their union $\mathcal{G}_{\mathrm{IP}}([A,B]^{(n)})$ can be computed in time polynomial in n, completing the proof. $\square$

Example 4.2.3 Consider the matrices $A = [1\ 1]$ and $B = I_2$. The Graver complexity of the pair A,B is $g(A,B) = 2$. The 2-fold matrix and its Graver basis, consisting of two antipodal vectors only, are

$$[A,B]^{(2)} = A^{(2)} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \qquad \mathcal{G}_{\mathrm{IP}}\left([A,B]^{(2)}\right) = \{\pm(1,-1,-1,1)\}.$$

By Theorem 4.2.2, the Graver basis of the 4-fold matrix $[A,B]^{(4)} = A^{(4)}$ can be computed by taking the union of the images of the $6 = \binom{4}{2}$ maps $\phi_{k_1,k_2} : \mathbb{Z}_+^{2\cdot 2} \longrightarrow \mathbb{Z}_+^{4\cdot 2}$ for $1 \leq k_1 < k_2 \leq 4$, and we obtain

$$[A,B]^{(4)} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}, \qquad \mathcal{G}_{\mathrm{IP}}\left([A,B]^{(4)}\right) = \left\{ \begin{array}{c} \pm(1,-1,-1,1,0,0,0,0) \\ \pm(1,-1,0,0,-1,1,0,0) \\ \pm(1,-1,0,0,0,0,-1,1) \\ \pm(0,0,1,-1,-1,1,0,0) \\ \pm(0,0,1,-1,0,0,-1,1) \\ \pm(0,0,0,0,1,-1,-1,1) \end{array} \right\}.$$

□

4.3 Polynomial time algorithm for n-fold integer programming

We now provide the polynomial time algorithm for the generalized n-fold integer programming problem

$$\min\{c^{\intercal}x : [A,B]^{(n)}x = b, x \in \mathbb{Z}_+^{nq}\}. \tag{4.3.1}$$

First, combining the results of the previous section and the previous chapter, we get a polynomial time procedure for converting any feasible solution to an optimal one. We record this result in the following lemma.

Lemma 4.3.1 *Fix any pair of integer matrices $A \in \mathbb{Z}^{r\times q}$ and $B \in \mathbb{Z}^{s\times q}$. Then there is a polynomial time algorithm that, given n, objective vector $c \in \mathbb{Z}_+^{nq}$, and non-negative integer vector $x \in \mathbb{Z}_+^{nq}$, solves the generalized n-fold integer programming problem in which x is feasible, i.e. the one with $b := [A,B]^{(n)}x$.*

Proof. First, apply the polynomial time algorithm underlying Theorem 4.2.2 on input n and compute the Graver basis $\mathcal{G}_{\mathrm{IP}}([A,B]^{(n)})$ of the n-fold matrix $[A,B]^{(n)}$. Then apply the polynomial time algorithm underlying Theorem 3.5.3 on input $[A,B]^{(n)}$, $\mathcal{G}_{\mathrm{IP}}([A,B]^{(n)})$, c and x, solving the integer programming problem in (4.3.1). □

We now show that, moreover, given any b, we can efficiently find an initial feasible solution to Problem (4.3.1).

Lemma 4.3.2 *Fix any pair of integer matrices $A \in \mathbb{Z}^{r\times q}$ and $B \in \mathbb{Z}^{s\times q}$. Then there is a polynomial time algorithm that, given n and demand vector $b \in \mathbb{Z}_+^{s+nr}$, either finds a feasible solution $x \in \mathbb{Z}_+^{nq}$ to the generalized n-fold integer programming problem in (4.3.1), or asserts that no feasible solution exists.*

Proof. Introduce $2n(r+s)$ auxiliary variables to the given generalized n-fold integer program, and denote by z the resulting vector of $n(2(r+s)+q)$ variables. Consider the auxiliary integer program of finding a non-negative integer vector z that minimizes the sum of the auxiliary variables subject to the following system of equations, with I_r and I_s the $r\times r$ and $s\times s$ identity matrices:

$$\begin{pmatrix} B & I_s & -I_s & 0 & 0 & B & I_s & -I_s & 0 & 0 & B & \cdots & B & I_s & -I_s & 0 & 0 \\ A & 0 & 0 & I_r & -I_r & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & A & 0 & 0 & I_r & -I_r & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ & & & & & & & & & & & \ddots & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & A & 0 & 0 & I_r & -I_r \end{pmatrix} \cdot z = b.$$

This auxiliary program is in fact again a generalized n-fold integer program, with matrices $\overline{A} = (A, 0, 0, I_r, -I_r)$ and $\overline{B} = (B, I_s, -I_s, 0, 0)$. Since A and B are fixed, so are $\overline{A}$ and $\overline{B}$. Due to the special structure of the auxiliary program, a feasible solution of this program can be written down easily in terms of b. Consequently, the auxiliary program can be solved by the algorithm underlying Lemma 4.3.1, in time polynomial in n and the bit size of b. Since the auxiliary objective is bounded below by zero, the algorithm will output an optimal solution z. If the optimal objective value is (strictly) positive, then the original n-fold program in (4.3.1) has no feasible solution, whereas if the optimal value is zero, then the restriction of z to the original variables is a feasible solution x of the original program in (4.3.1). $\square$

Combining the results of Lemmas 4.3.1 and 4.3.2, we obtain the main result of this chapter.

Theorem 4.0.2. *Fix any pair of integer matrices A, B of compatible sizes. Then there is a polynomial time algorithm that solves the generalized n-fold integer programming problem on any input n, b, c.*

Clearly, Theorem 4.0.1 is deduced from Theorem 4.0.2 in the special case $B = I_q$. We emphasize again that, by solving the generalized n-fold integer programming problem, we mean in the complete sense that the algorithm concludes with precisely one of the following: it either asserts that there is no feasible solution, or asserts that the objective function is unbounded, or returns an optimal solution.

Chapter 5

Exploiting symmetry in the computation of Graver bases

Graver bases have a variety of interesting applications. Unfortunately, the size of Graver bases increases quickly with the dimension, making it very hard if not impossible to compute them in practice. In several applications, however, as for example in algebraic statistics, the problems involve a high symmetry that should make it much easier to compute the Graver basis in terms of (relatively few) representatives. We have seen such a phenomenon already in the previous chapter. Nonetheless, let us demonstrate the problem by an example.

Example 5.0.1 Consider the set of all 3×3 tables/arrays whose entries are filled with integer numbers in such a way that the sums along each row and along each column are 0. One particular example is the table

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 3 & -2 \\ 0 & -2 & 2 \end{pmatrix}.$$

If we encode the 9 entries of the table as $z_1, \ldots, z_9$, then the set of 3×3 tables coincides with the integer vectors in the kernel of the matrix

$$A_{3\times3} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix},$$

that is, with all $z \in \mathbb{Z}^9$ satisfying $A_{3\times3}z = 0$. The Graver basis of $A_{3\times3}$ consists of all $\sqsubseteq$-minimal *nonzero* tables among them. The particular 3×3 table above does not belong to the Graver basis of $A_{3\times 3}$, since

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \sqsubseteq \begin{pmatrix} 1 & -1 & 0 \\ -1 & 3 & -2 \\ 0 & -2 & 2 \end{pmatrix}.$$

Using the computer program `4ti2` [70], we find that the following 15 vectors (and their negatives) constitute the Graver basis of $A_{3\times 3}$:

$$\begin{array}{rrrrrrrrr}
(1, & -1, & 0, & -1, & 1, & 0, & 0, & 0, & 0) \\
(0, & 0, & 0, & 1, & 0, & -1, & -1, & 0, & 1) \\
(1, & 0, & -1, & -1, & 0, & 1, & 0, & 0, & 0) \\
(1, & -1, & 0, & 0, & 0, & 0, & -1, & 1, & 0) \\
(0, & 0, & 0, & 1, & -1, & 0, & -1, & 1, & 0) \\
(1, & -1, & 0, & -1, & 0, & 1, & 0, & 1, & -1) \\
(0, & -1, & 1, & 1, & 0, & -1, & -1, & 1, & 0) \\
(1, & -1, & 0, & 0, & 1, & -1, & -1, & 0, & 1) \\
(1, & 0, & -1, & 0, & 0, & 0, & -1, & 0, & 1) \\
(0, & -1, & 1, & 0, & 1, & -1, & 0, & 0, & 0) \\
(0, & 1, & -1, & 1, & -1, & 0, & -1, & 0, & 1) \\
(0, & 0, & 0, & 0, & 1, & -1, & 0, & -1, & 1) \\
(0, & -1, & 1, & 0, & 0, & 0, & 0, & 1, & -1) \\
(1, & 0, & -1, & 0, & -1, & 1, & -1, & 1, & 0) \\
(1, & 0, & -1, & -1, & 1, & 0, & 0, & -1, & 1)
\end{array}$$

However, there is an obvious symmetry group $S_3 \times S_3 \times S_2$ operating on the set of 3×3 tables whose elements transform a given table $v \in \ker(A_{3\times 3})$ into another table $w \in \ker(A_{3\times 3})$ by suitably rearranging components (permuting rows or columns, flipping the table along the main diagonals). If we take these symmetries into account, we see that among these 15 elements there are in fact only two essentially different elements:

$$\begin{array}{rrrrrrrrr}
(1, & -1, & 0, & -1, & 1, & 0, & 0, & 0, & 0) \\
(1, & -1, & 0, & -1, & 0, & 1, & 0, & 1, & -1)
\end{array}$$

or, in a more array-like notation:

$$\begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \end{pmatrix}.$$

It should be clear that for bigger or for higher-dimensional tables, this difference in sizes becomes far more striking, since the acting symmetry groups are much bigger. $\square$

In the following, let a sublattice $\mathcal{L} \subseteq \mathbb{Z}^n$ be given and let $\mathcal{S}_\mathcal{L} \subseteq S_n$ be a group of symmetries such that for all $v \in \mathcal{L}$ and for all $\sigma \in \mathcal{S}_\mathcal{L}$ we have that also $\sigma(v) := (v^{\sigma(1)}, \ldots, v^{\sigma(n)}) \in \mathcal{L}$. Finally, denote by $\text{orb}_{\text{S}_\mathcal{L}}(v) := \{\sigma(v) : \sigma \in \mathcal{S}_\mathcal{L}\}$ the orbit of v under $\mathcal{S}_\mathcal{L}$.

Lemma 5.0.2 *Let $\mathcal{L} \subseteq \mathbb{Z}^n$ be a lattice, let $\mathcal{S}_{\mathcal{L}} \subseteq S_n$ be the group of its symmetries, and let $g, g', s, s' \in \mathcal{L}$. Then the following holds:*

- *If $g \sqsubseteq s$ then $\sigma(g) \sqsubseteq \sigma(s)$ for every $\sigma \in \mathcal{S}_{\mathcal{L}}$.*
- *If there is no $g' \in \mathrm{orb}_{\mathrm{S}_{\mathcal{L}}}(g)$ with $g' \sqsubseteq s$, then for every $s' \in \mathrm{orb}_{\mathrm{S}_{\mathcal{L}}}(s)$ there is no $g'' \in \mathrm{orb}_{\mathrm{S}_{\mathcal{L}}}(g)$ with $g'' \sqsubseteq s'$.*
- *If $v \in \mathcal{G}_{IP}(\mathcal{L})$ then $\mathrm{orb}_{\mathrm{S}_{\mathcal{L}}}(v) \subseteq \mathcal{G}_{IP}(\mathcal{L})$.*

Proof. The first statement follows immediately from the definition of $\sqsubseteq$.

For the second statement, let $s' = \sigma(s)$ for some $\sigma \in \mathcal{S}_{\mathcal{L}}$ and assume there is some $g'' \in \mathrm{orb}_{\mathrm{S}_{\mathcal{L}}}(g)$ with $g'' \sqsubseteq s'$. Then $g' := \sigma^{-1}(g'') \sqsubseteq \sigma^{-1}(s') = s$ and $g' \in \mathrm{orb}_{\mathrm{S}_{\mathcal{L}}}(g)$. A contradiction to the assumed non-existence of such g'.

For the last statement we have to show that with $v \in \mathcal{G}_{\mathrm{IP}}(\mathcal{L})$ also the full orbit $\mathrm{orb}_{\mathrm{S}_{\mathcal{L}}}(v)$ lies in $\mathcal{G}_{\mathrm{IP}}(\mathcal{L})$. For this it suffices to assume that there is some $\sigma(v) \in \mathrm{orb}_{\mathrm{S}_{\mathcal{L}}}(v)$ that could be written non-trivially as $\sigma(v) = w_1 + w_2$ with $w_1, w_2 \sqsubseteq \sigma(v)$. But this would imply $v = \sigma^{-1}(w_1) + \sigma^{-1}(w_2)$ with nonzero $\sigma^{-1}(w_1), \sigma^{-1}(w_2) \sqsubseteq v$, which contradicts $v \in \mathcal{G}_{\mathrm{IP}}(\mathcal{L})$. Thus, $\sigma(v)$ must belong to $\mathcal{G}_{\mathrm{IP}}(\mathcal{L})$. □

As a consequence of this lemma, $\mathcal{G}_{\mathrm{IP}}(\mathcal{L})$ decomposes completely into *full* orbits. Our task of computing $\mathcal{G}_{\mathrm{IP}}(\mathcal{L})$ thus reduces to computing representatives of these orbits, and to collect them into a set $\mathcal{G}^{\mathrm{sym}}(\mathcal{L})$. By the previous lemma, we recover $\mathcal{G}_{\mathrm{IP}}(\mathcal{L})$ via

$$\mathcal{G}_{\mathrm{IP}}(\mathcal{L}) = \bigcup_{v \in \mathcal{G}^{\mathrm{sym}}(\mathcal{L})} \mathrm{orb}_{\mathrm{S}_{\mathcal{L}}}(v).$$

Note that this last expression does not compute a superset of $\mathcal{G}_{\mathrm{IP}}(\mathcal{L})$. In contrast to this, the last statement of Lemma 5.0.2 fails to be true in general for minimal toric Gröbner bases or for minimal Markov bases associated with the lattice $\mathcal{L}$, see Chapter 7 for details.

5.1 Computing the symmetric Graver basis

In this section, we adapt Pottier's algorithm [102], Algorithm 2.4.1 from page 26, to deal with the symmetries of $\mathcal{L}$ in the computation of $\mathcal{G}_{\mathrm{IP}}(\mathcal{L})$. Although this algorithm is not the fastest way to compute Graver bases directly, it will be easier for us to exploit the given symmetries (and to present the main ideas). The state-of-the-art algorithm, Algorithm 2.5.1 from page 28, exploits the positive sum property of Graver bases and needs to break the symmetry to do so. Nonetheless, we show how to exploit the symmetries also in this situation and arrive at an even faster "symmetric" algorithm.

Let us now adapt Algorithm 2.4.1 to exploit the symmetries. Let us start with an immediate consequence of Lemma 5.0.2. For the definition of normalForm(s, G) see again Algorithm 2.4.2 on page 26.

Corollary 5.1.1 *If $s = g_1 + \ldots + g_r$, $g_i \sqsubseteq s$, $i = 1, \ldots, r$, then $\sigma(s) = \sigma(g_1) + \ldots + \sigma(g_r)$, $\sigma(g_i) \sqsubseteq \sigma(s)$, $i = 1, \ldots, r$ for every $\sigma \in \mathcal{S}_{\mathcal{L}}$. Consequently, if $G = \bigcup_{v \in G^{sym}} \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(v)$ and if* $\mathrm{normalForm}(s, G) = 0$ *then* $\mathrm{normalForm}(\sigma(s), G) = 0$ *for every $\sigma \in \mathcal{S}_{\mathcal{L}}$.*

In other words, if a representation $s = g_1 + \ldots + g_r$, $g_i \in G$, $g_i \sqsubseteq s$, $i = 1, \ldots, r$ has been found, the symmetry of $\mathcal{L}$ already guarantees existence of a similar representation for every element in $\mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(s)$. Thus, we have to check $\mathrm{normalForm}(\overline{s}, G) = 0$ only for *one* vector $\overline{s} \in \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(s)$.

Finally, we are in the position to exploit symmetries in Pottier's algorithm. The main difference to the original algorithm will be that instead of keeping the sets G and C in memory, we only store their representatives under the given symmetry in sets G^{sym} and C^{sym}. (At any point during the "symmetric" algorithm we may go back to the original algorithm by replacing all elements in G^{sym} and C^{sym} by the vectors from their orbits under $\mathcal{S}_{\mathcal{L}}$.)

Moreover, there a few more changes. Once we have found a nonzero vector f that is to be added to G (and to G^{sym} as a new representative), we assume that we add the full orbit $\mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(f)$ to G, as we know that the Graver basis would contain the full orbit if f was in fact a Graver basis element. Accordingly, instead of adding only the vectors

$$\bigcup_{g \in G} \{f + g\}$$

to C, we immediately include the vectors

$$\bigcup_{f' \in \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(f), g \in G} \{f' + g\}.$$

As G will always be a union of full orbits, this last expression can be transformed to

$$\bigcup_{f' \in \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(f), g \in G} \{f' + g\} = \bigcup_{g \in G^{\mathrm{sym}}} \bigcup_{\substack{f' \in \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(f) \\ g' \in \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(g)}} \{f' + g'\} = \bigcup_{g \in G^{\mathrm{sym}}} \bigcup_{g' \in \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(g)} \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(f + g''),$$

since $f' + g' = \sigma(f + g'')$ for some $\sigma \in \mathcal{S}_{\mathcal{L}}$ and some $g'' \in \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(g)$. Therefore, we update C^{sym} as follows:

$$C^{\mathrm{sym}} = C^{\mathrm{sym}} \cup \bigcup_{g \in G^{\mathrm{sym}}} \bigcup_{g' \in \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(g)} \{f + g'\}.$$

Note that since

$$\bigcup_{g' \in \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(g)} \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(f + g') = \bigcup_{f' \in \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(f)} \mathrm{orbs}_{\mathrm{S}_{\mathcal{L}}}(f' + g)$$

we have a choice in adding either all vectors $\{f + g'\}$ or all vectors $\{f' + g\}$ to C^{sym}. Clearly, one would choose to add as few new representatives to C^{sym} as possible to keep the number of S-vectors that need to be reduced small. After all, these reductions are the most expensive part of the algorithm. This saving in the number of critical vectors is what makes this algorithm much faster than Pottier's algorithm when applied to lattices with high symmetry.

In the following algorithm, $\mathrm{reps}_{\mathrm{S}_{\mathcal{L}}}(H)$ for a set $H \subseteq \mathcal{L}$ of vectors shall denote a set of representatives of H under the symmetry group $\mathcal{S}_{\mathcal{L}}$.

Algorithm 5.1.2 (Algorithm to Compute $G^{\mathbf{sym}}(\mathcal{L})$)

Input: generating set F of $\mathcal{L}$ over $\mathbb{Z}$

Output: a set G which contains $\mathcal{G}_{\text{IP}}(\mathcal{L})$

```
$G^{\text{sym}} := \text{reps}_{\text{S}_\mathcal{L}}(F \cup -F)$                         $G := \text{orbs}_{\text{S}_\mathcal{L}}(F \cup -F)$
$C^{\text{sym}} := \text{reps}_{\text{S}_\mathcal{L}}\left(\bigcup_{f,g\in G}\{f+g\}\right)$            $C := \bigcup_{f,g\in G}\{f+g\}$
while $C^{\text{sym}} \neq \emptyset$ do
    $s :=$ an element in $C^{\text{sym}}$
    $C^{\text{sym}} := C^{\text{sym}} \setminus \{s\}$                                $C := C \setminus \text{orbs}_{\text{S}_\mathcal{L}}(s)$
    $f := \text{normalForm}(s, G^{\text{sym}}) := \text{normalForm}(s, G)$      $f := \text{normalForm}(s, G)$
    if $f \neq 0$ then
        $G^{\text{sym}} := G^{\text{sym}} \cup \{f\}$                             $G := G \cup \text{orbs}_{\text{S}_\mathcal{L}}(f)$
        $C^{\text{sym}} := C^{\text{sym}} \cup \bigcup_{g\in G} \bigcup_{g'\in\text{orbs}_{\text{S}_\mathcal{L}}(g)} \{f+g'\}$    $C := C \cup \bigcup_{f'\in\text{orbs}_{\text{S}_\mathcal{L}}(f), g\in G} \{f'+g\}$
return $G = \text{orbs}_{\text{S}_\mathcal{L}}(G^{\text{sym}})$.
```

In this algorithm, we compute normalForm(s, G^{sym}) via normalForm(s, G^{sym}) := normalForm(s, G). Clearly, from a practical perspective, one would not want to keep the huge set G in memory or to recompute it when needed. Then, of course, one needs to think about how to compute normalForm(s, G^{sym}) efficiently if only G^{sym} instead of G is available. (For example, G might be simply too big to be kept in memory.) This is still an open question and any significant improvement in the solution of this problem would lead to an equally significant algorithmic improvement of the overall algorithm.

Lemma 5.1.3 *Algorithm 5.1.2 always terminates and returns a set G containing $\mathcal{G}_{IP}(\mathcal{L})$.*

Proof. To prove termination, consider the sequence of vectors in $G^{\text{sym}} \setminus \text{reps}_{\text{S}_\mathcal{L}}(F \cup -F) = \{f_1, f_2, \ldots\}$ as they are added to G^{sym} during the run of the algorithm. By construction, we have $f_i \not\sqsubseteq f_j$ whenever $i < j$. Thus, by the Gordan-Dickson Lemma, Lemma 2.1.3, this sequence must be finite and the algorithm terminates.

Note that throughout the run of the algorithm, we always have $G = \text{orbs}_{\text{S}_\mathcal{L}}(G^{\text{sym}})$. Upon termination we know that normalForm$(f+g, G) = 0$ for every pair of vectors $f, g \in G$. Thus, G must contain the Graver basis $\mathcal{G}_{\text{IP}}(\mathcal{L})$. $\square$

5.2 Computing the symmetric Graver basis faster

As in Section 2.5, let π_d be again the projection of an n-dimensional vector onto its first d components and assume the this projection of $\mathcal{L}$ is injective, that is, $\pi_d(\mathcal{L}) = \{0\}$. Moreover, remember that we defined $\|.\|$ in Section 2.5 as $\|v\| := \|\pi_d(v)\|_1$. Although this definition of $\|.\|$ in breaks most if not all existing symmetry in the problem, we will now combine the ideas of Sections 2.5 and 5.1 to a faster algorithm to compute symmetric Graver bases. The main idea is to use the norm $\|.\|$ defined on $\mathbb{Z}^n$ to define a norm on orbits: For $T \subseteq \mathcal{L}$, we define $\|T\| := \min\{\|v\| : v \in T\} = \min\{\|\pi_d(v)\|_1 : v \in T\}$. Then the new symmetric Graver basis algorithm looks as follows.

Algorithm 5.2.1 (Faster Algorithm to Compute $G^{\mathbf{sym}}(\mathcal{L})$)

```
Input: set $\overline{F} \subseteq \mathcal{L}$ such that $\pi_d(\overline{F})$ is the set of $\sqsubseteq$-minimal nonzero vectors in $\pi_d(\mathcal{L})$
Output: a set $G$ which contains $\mathcal{G}_{\mathrm{IP}}(\mathcal{L})$

$G^{\mathrm{sym}} := \mathrm{reps}_{S_\mathcal{L}}(\overline{F})$
$C^{\mathrm{sym}} := \mathrm{reps}_{S_\mathcal{L}}\left(\bigcup_{f,g\in G}\{f+g\}\right)$
while $C^{\mathrm{sym}} \neq \emptyset$ do
    $s :=$ an element in $C^{\mathrm{sym}}$ with smallest value of $\|\mathrm{orb}_{S_\mathcal{L}}(s)\|$
    $C^{\mathrm{sym}} := C^{\mathrm{sym}} \setminus \{s\}$
    $f := \mathrm{normalForm}(s, G^{\mathrm{sym}})$
    if $f \neq 0$ then
        $G^{\mathrm{sym}} := G^{\mathrm{sym}} \cup \{f\}$
        $C^{\mathrm{sym}} := C^{\mathrm{sym}} \cup \bigcup_{g\in G} \bigcup_{g'\in\mathrm{orb}_{S_\mathcal{L}}(g)} \{f+g'\}$
return $G = \mathrm{orb}_{S_\mathcal{L}}(G^{\mathrm{sym}})$.
```

Due to our special input set and our norm defined on orbits, we can again simplify and speed up the normal form computation $\mathrm{normalForm}(s, G^{\mathrm{sym}}) := \mathrm{normalForm}(s, \mathrm{orb}_{S_\mathcal{L}}(G^{\mathrm{sym}}))$ by using Algorithm 2.5.2 instead of Algorithm 2.4.2. Again, from a practical perspective, one would want to compute $\mathrm{normalForm}(s, G^{\mathrm{sym}})$ without recovering or storing the huge set $\mathrm{orb}_{S_\mathcal{L}}(G^{\mathrm{sym}})$.

On the other hand, it should be noted that we do not, as in Algorithm 2.5.1, have an orthant condition in Algorithm 5.2.1 that reduces the number of vectors added to C^{sym}. This is done to "undo" the symmetry breaking caused by $\|.\|$.

Lemma 5.2.2 *Algorithm 5.2.1 always terminates and returns the set $G = \mathcal{G}_{IP}(\mathcal{L})$.*

Proof. The subsequent proof follows similar lines as the proof of Lemma 2.5.3.

To prove termination, we consider again the sequence of vectors in $G^{\mathrm{sym}} \setminus \mathrm{reps}_{S_\mathcal{L}}(\overline{F}) = \{f_1, f_2, \ldots\}$ as they are added to G^{sym} during the run of the algorithm. By construction, we have $f_i \not\sqsubseteq f_j$ whenever $i < j$. Thus, by the Gordan-Dickson Lemma, Lemma 2.1.3, this sequence must be finite and the algorithm terminates.

Next we show that $\mathcal{G}_{\mathrm{IP}}(\mathcal{L}) \subseteq G$. Assume that this is not the case. Among all elements $z \in \mathcal{G}_{\mathrm{IP}}(\mathcal{L}) \backslash G$ choose one with $\|\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)\|$ smallest. Moreover, we may assume that z is a representative of $\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)$ with $\|\pi_d(z)\|_1 = \|\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)\|$. By our generating assumption on the set $\overline{F} \subseteq G$, there exists a non-trivial representation $z = \sum \alpha_i v_i$ with positive integers α_i and vectors $v_i \in G$ with $\pi_d(v_i) \sqsubseteq \pi_d(z)$. From the set of all such linear integer combinations choose one such that $\sum \alpha_i \|v_i\|_1$ is minimal. Note that from $\pi_d(v_i) \sqsubseteq \pi_d(z)$ and the fact that the relation $z = \sum \alpha_i v_i$ is non-trivial, we conclude $\|\pi_d(v_i)\|_1 < \|\pi_d(z)\|_1$ and thus $\|\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(v_i)\| < \|\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)\|$ for all i.

Let us assume first that $\sum \alpha_i \|v_i\|_1 > \|z\|_1$. Therefore, there have to exist vectors v_{i_1}, v_{i_2} in this representation which have some component $k = k_0$ of different signs. By construction, $k_0 > d$, as the v_i all have the same sign as z on the first d components.

The orbit of the vector $v_{i_1} + v_{i_2}$ was added to the set C^{sym} during the run of the algorithm. If we had $\|v_{i_1} + v_{i_2}\| = \|z\|$, then all other vectors v_i, $i \neq i_1, i_2$ satisfy $\|v_i\| = 0$ as $\pi_d(v_i) \sqsubseteq \pi_d(z)$ for all i. But $\pi_d(v_i) = 0$ implies $v_i = 0$ and thus $v_{i_1} + v_{i_2} = z$. Since $v_{i_1} + v_{i_2} (= z)$ is a vector whose orbit (or better: a representative of it) was added to C^{sym} during the run of the algorithm, a representative of $\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)$ is eventually chosen as $s \in C^{\mathrm{sym}}$. Being $\sqsubseteq$-minimal, we have $\operatorname{normalForm}(\operatorname{reps}_{\mathrm{S}_{\mathcal{L}}}(\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)), G^{\mathrm{sym}}) = \operatorname{reps}_{\mathrm{S}_{\mathcal{L}}}(\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z))$ and thus, $\operatorname{reps}_{\mathrm{S}_{\mathcal{L}}}(\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z))$ must have been added to G^{sym}, in contradiction to our assumption $z \notin G$.

Therefore, we may assume that $\|v_{i_1} + v_{i_2}\| < \|z\|$. However, since all Graver basis elements with norm strictly smaller than $\|z\|$ are assumed to be in G, we may continue literally as in the proof of Lemma 2.5.3: rewrite $z = \sum \alpha_i v_i$ and arrive at a contradiction to the minimality of $\sum \alpha_i \|v_i\|_1$.

Thus, we must have $\sum \alpha_i \|v_i\|_1 = \|z\|_1$, which implies that $v_i \sqsubseteq z$ for all i. As z is $\sqsubseteq$-minimal, this is only possible if the relation $z = \sum \alpha_i v_i$ is trivial, that is $z = v_1$. As $v_1 \in G$ we now also have $z \in G$ as desired.

To show that not only $\mathcal{G}_{\mathrm{IP}}(\mathcal{L}) \subseteq G$ but in fact $\mathcal{G}_{\mathrm{IP}}(\mathcal{L}) = G$ is true, assume $G \setminus \mathcal{G}_{\mathrm{IP}}(\mathcal{L}) \neq \emptyset$. Among all elements $z \in G \setminus \mathcal{G}_{\mathrm{IP}}(\mathcal{L})$ choose one with $\|\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)\|$ smallest. Moreover, we may again assume that z is a representative of $\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)$ with $\|\pi_d(z)\|_1 = \|\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)\|$. Note that by Lemma 5.0.2 and since $z \notin \mathcal{G}_{\mathrm{IP}}(\mathcal{L})$, not a single element in $\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)$ belongs to $\mathcal{G}_{\mathrm{IP}}(\mathcal{L})$.

We now start with a non-trivial representation $z = \sum \alpha_i v_i$ with positive integers α_i and vectors $v_i \in \overline{F} \subseteq G$ with $\pi_d(v_i) \sqsubseteq \pi_d(z)$. Clearly, $\pi_d(v_i) \sqsubseteq \pi_d(z)$ implies $\|\pi_d(v_i)\|_1 < \|\pi_d(z)\|_1$ and thus $\|\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(v_i)\| < \|\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)\|$ for all i. Next, we follow similar steps as above (or more precisely: as in the proof of Lemma 2.5.3) to change $z = \sum \alpha_i v_i$ into a representation $z = \sum \beta_j w_j$ with $\beta_j > 0$, $w_j \in G$ and $w_j \sqsubseteq z$ for all j. As z is not $\sqsubseteq$-minimal, this representation is not trivial.

Note that in $z = \sum \alpha_i v_i$, $z = \sum \beta_j w_j$, and in any intermediate representation $z = \sum \beta_k u_k$, we have that the summands v_i, w_j, u_k have a strictly smaller norm on the first d components as z. In fact, the same property holds for the sum of two summands that are iteratively needed for the rewriting steps. Thus, their corresponding orbits also have a strictly smaller norm than $\|\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)\|$. We conclude that at the time $\operatorname{reps}_{\mathrm{S}_{\mathcal{L}}}(\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z))$ was chosen from C^{sym} and then added to G^{sym}, representatives for all orbits $\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(w_j)$ were already in G^{sym}. Thus, $\operatorname{normalForm}(\operatorname{reps}_{\mathrm{S}_{\mathcal{L}}}(\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z)), G^{\mathrm{sym}})$ must have returned 0 in contradiction to the assumption that $\operatorname{reps}_{\mathrm{S}_{\mathcal{L}}}(\operatorname{orb}_{\mathrm{S}_{\mathcal{L}}}(z))$ was added to G^{sym}.

We conclude $G \setminus \mathcal{G}_{\mathrm{IP}}(\mathcal{L}) = \emptyset$ and hence $G = \mathcal{G}_{\mathrm{IP}}(\mathcal{L})$. $\square$

5.3 Computational experiments

In this section we report on computational experience with a few examples that have symmetry. These problem all deal with 3-way tables, that is, with $k_1 \times k_2 \times k_3$ tables of unit cubes, each containing an integer number. For each problem, the lattice $\mathcal{L}$ is the set of lattice points in the kernel of the matrix that encodes the conditions that the sums along each one-dimensional row parallel to a coordinate axis are 0. Example 5.0.1 presented the case of 3×3 tables.

The following running times demonstrate that, as expected, our symmetric algorithm heavily speeds up the computation. (Times are given in seconds on a Sun UltraSparc III+ with 1.05 GHz.)

Problem	$\lvert\mathcal{S}_{\mathcal{L}}\rvert$	Size of Graver basis	$\lvert G^{\mathrm{sym}}\rvert$	Algorithm 2.5.1	Algorithm 5.2.1
$3 \times 3 \times 3$	$1,296$	795	7	2	1
$3 \times 3 \times 4$	$1,728$	$19,722$	27	$1,176$	9
$3 \times 3 \times 5$	$8,640$	$263,610$	61	$560,517$	526
$3 \times 4 \times 4$	$6,912$	$4,617,444$	784	$--$	$260,590$

Chapter 6

Graver test sets for integer programs with $\mathbb{Z}$-convex objective

As we have seen in Chapter 3, integer linear optimization problems

$$(\mathrm{IP})_{c,b}: \qquad \min\{c^\intercal z : Az = b, z \in \mathbb{Z}_+^n\},$$

can be solved by a simple augmentation scheme, Algorithm 3.0.2. There are two key steps in this algorithmic scheme: finding an initial feasible solution and finding improving vectors. Remember that universal test sets, which depend only on the problem matrix A, are good for both of these tasks.

Example 6.0.1 Consider the problem

$$\min\{x + y : x, y \in \mathbb{Z}_+\}.$$

The Graver test set associated to the problem matrix $A = 0$ is $\{\pm(1,0), \pm(0,1)\}$. As one can easily check, already the subset $\{(1,0),(0,1)\}$ provides an improving direction to any non-optimal solution of this particular problem instance. Thus, with the help of $\{(1,0),(0,1)\}$, we can augment any given feasible solution to the (in this case unique) optimal solution $(0,0)$. $\square$

Intrinsic to the proofs that there do exist finite (universal) test sets for $(\mathrm{IP})_{c,b}$ and that they do indeed provide an improving direction to any non-optimal feasible solution, is the fact that both the objective function and the constraints are *linear*. Now let us observe what happens with a *non-linear* objective function.

Example 6.0.2 Consider the problem

$$\min\{(x+y)^2 + 4(x-y)^2 : x, y \in \mathbb{Z}_+\}.$$

As again $A = 0$, the corresponding Graver basis is $\mathcal{G}_{\text{IP}}(A) = \{\pm(1,0), \pm(0,1)\}$. However, this universal test set for the integer *linear* program $(\text{IP})_{c,b}$ does not provide an improving direction to any non-optimal feasible solution for the *quadratic* problem given above:

Clearly, $(0,0)$ is again the unique optimal solution with objective value 0. Now consider the point $(1,1)$ with objective value 4. There are 4 points reachable from $(1,1)$ via the directions given by $\mathcal{G}_{\text{IP}}(A)$: $(1,0)$ and $(0,1)$, both with objective value 5, and $(2,1)$ and $(1,2)$, both with objective value 13. Therefore, in order to reach the optimum $(0,0)$ from $(1,1)$, additional vectors are needed in the test set.

As we will see below, the set $\mathcal{G}_{\text{IP}}(A) \cup \{\pm(1,1)\}$ provides improving directions to any non-optimal solution of the above quadratic problem. Moreover, this property remains true even if we change the objective function in a certain way. (For details see below.) For example, with the directions from $\mathcal{G}_{\text{IP}}(A) \cup \{\pm(1,1)\}$ we can also find the optimum of the following program:

$$\min\{e^{|x+y-3|} + 4(x-y+2)^6 + 2x - y : x, y \in \mathbb{Z}_+\} \qquad \square$$

In this chapter we relieve the restriction to linear objective functions and employ test set methods for the solution of integer optimization problems

$$(\text{CIP})_{f,b}: \qquad \min\{f(z) : Az = b, z \in \mathbb{Z}_+^n\},$$

where $A \in \mathbb{Z}^{d\times n}$, $b \in \mathbb{Z}^d$, and where

$$f(z) := \sum_{i=1}^{s} f_i(c_i^\intercal z + c_{i,0}) + c^\intercal z.$$

Herein, $c \in \mathbb{R}^n$, $c_1, \ldots, c_s \in \mathbb{Z}^n$, $c_{1,0}, \ldots, c_{s,0} \in \mathbb{Z}$, and $f_i : \mathbb{R} \to \mathbb{R}$, $i = 1, \ldots, s$, are $\mathbb{Z}$-convex functions with minimum at 0. We call $g : \mathbb{R} \to \mathbb{R}$ a *$\mathbb{Z}$-convex* function with minimum at $\alpha \in \mathbb{Z}$, if the function $g(x+1) - g(x)$ is increasing on $x \in \mathbb{Z}$ and if $g(x+1) - g(x) \leq 0$ for all integers $x < \alpha$ and $g(x+1) - g(x) \geq 0$ for all integers $x \geq \alpha$. Clearly, these three conditions imply that $x = \alpha$ is a minimum of g over $\mathbb{Z}$. We will, however, restrict our attention to $\mathbb{Z}$-convex functions with minimum at 0. This is in fact no restriction, since we can transform any $\mathbb{Z}$-convex function g with minimum at α to one with minimum at 0 by considering $\overline{g}(x) = g(x+\alpha)$ instead.

The problem type $(\text{CIP})_{f,b}$ includes for example linear integer programs for $f_1 = \cdots = f_s = 0$, or quadratic integer programs for $f_i(x) = x^2$. However, one could apply our approach also to more exotic functions like $f_i(x) = |x|$ or $f_i(x) = -x$ for $x \leq 0$ and $f_i(x) = e^x$ for $x > 0$, that is, the functions f_i considered as functions from $\mathbb{R}$ to $\mathbb{R}$ need not be continuous.

Our main result is the following. Note that Murota, Saito, and Weismantel [97] arrive at the same result via a refinement of cones and a union of their inclusion-minimal Hilbert bases.

Theorem 6.0.3 *Let $A \in \mathbb{Z}^{d\times n}$ and $c_1, \ldots, c_s \in \mathbb{Z}^n$ be given. Denote by C the $s \times n$ matrix whose rows are formed by the vectors $c_1^\intercal, \ldots, c_s^\intercal$. Moreover, let I_s denote the $s \times s$ unit matrix. Then for any particular choice*

- *of $\mathbb{Z}$-convex functions $f_1, \ldots, f_s$ with minima at $x = 0$,*
- *of $c_{1,0}, \ldots, c_{s,0} \in \mathbb{Z}$, and*
- *of $c \in \mathbb{R}^n$,*

the set

$$\mathcal{H}_{CIP}(A, C) := \phi_n \left(\mathcal{G}_{IP} \begin{pmatrix} A & 0 \\ C & I_s \end{pmatrix} \right)$$

provides an improving direction to any non-optimal feasible solution of the problem $(CIP)_{f,b}$. Herein, ϕ_n defines the projection of a vector onto its first n components, and for a set G of vectors $\phi_n(G)$ denotes the set of images of elements in G under ϕ_n.

Trivially, $\mathcal{G}_{\mathrm{IP}}(A) \subseteq \mathcal{H}_{\mathrm{CIP}}(A, C)$ for any matrix C. However, as we have seen in Example 6.0.2, this inclusion can be strict.

For $f_1 = \cdots = f_s = 0$, we simply obtain $\mathcal{H}_{\mathrm{CIP}}(A, 0) = \mathcal{G}_{\mathrm{IP}}(A)$ as a (universal) test set for $(\mathrm{IP})_{c,b}$. But, as the following example shows, the set $\mathcal{G}_{\mathrm{IP}}(A)$ gives improving directions even for a far bigger problem class.

Example 6.0.4 Consider the family of problems $(\mathrm{CIP})_{f,b}$ where $c_1, \ldots, c_n$ are the unit vectors in $\mathbb{R}^n$, that is,

$$\min\{f(z) : Az = b, z \in \mathbb{Z}^n_+\}$$

with

$$f(z) := \sum_{i=1}^{s} f_i(z_i + c_{i,0}) + c^\intercal z.$$

As $C = I_n$, we need to compute the Graver basis of the Lawrence lifting

$$\begin{pmatrix} A & 0 \\ I_n & I_n \end{pmatrix}$$

of A. Since all elements in the kernel of this Lawrence lifting have the form $(u, -u)$ and since $(v, -v) \sqsubseteq (u, -u)$ in $\mathbb{Z}^{2n}$ if and only if $v \sqsubseteq u$ in $\mathbb{Z}^n$, this Graver basis is simply $\{(u, -u) : u \in \mathcal{G}_{\mathrm{IP}}(A)\}$. Thus, $\mathcal{H}_{\mathrm{CIP}}(A, I_n) = \mathcal{G}_{\mathrm{IP}}(A)$, showing that the set $\mathcal{G}_{\mathrm{IP}}(A)$ is also a test set for this bigger problem class where A is kept fixed and the remaining problem data is allowed to vary. $\square$

The remainder of this chapter is structured as follows: In Section 6.1 we study our test set approach for convex quadratic optimization problems, of which the Quadratic Assignment Problem (QAP) is probably the most famous example.

The quadratic assignment problem [28] deals with assigning n facilities to n locations such that a certain quadratic cost function is minimized. It can be formulated as the following problem involving permutation matrices :

$$\min\left\{\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} d_{ijkl}x_{ij}x_{kl} \quad + \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} : \begin{array}{l} \\ \sum_{j=1}^{n} x_{ij} = 1, \quad i \in \{1,\ldots,n\}, \\ \sum_{i=1}^{n} x_{ij} = 1, \quad j \in \{1,\ldots,n\}, \\ x_{ij} \in \{0,1\}, \quad i,j \in \{1,\ldots,n\}\right\}. \end{array}$$

The value d_{ijkl} can be seen as costs for assigning facility i to location j *and* facility k to location l, whereas c_{ij} models a fixed cost incurred by locating facility i to location j.

Even nowadays, QAP's of size $n > 30$ (that is, with more than only 900 binary variables) are still considered to be computationally extremely hard, if not intractable. One major problem in branch-and-bound algorithms that try to solve these problems is the lack of sharp lower bounds.

We conclude this chapter with Section 6.2, in which we prove our main theorem, Theorem 6.0.3.

6.1 Quadratic programs

In this section we deal with the special case of convex quadratic optimization problems

$$\min\{z^\intercal Qz + c^\intercal z : Az = b, z \in \mathbb{Z}_+^n\},$$

where Q is a symmetric, positive semi-definite matrix with only rational entries. These problems can be solved by the augmentation approach introduced in Chapter 3 using $\mathcal{H}_{\text{CIP}}(A, C)$. The reason for this is the following basic result from the theory of quadratic forms [90].

Lemma 6.1.1 *Let $Q \in \mathbb{Q}^{n\times n}$ be a symmetric matrix. Then there exist a diagonal matrix $D \in \mathbb{Q}^{n\times n}$ and an invertible matrix $U \in \mathbb{Q}^{n\times n}$ such that $Q = U^\intercal DU$. Moreover, each diagonal element d_{ii} of D is representable by the quadratic form $x^\intercal Qx$, that is, for all d_{ii} there is some $x_i \in \mathbb{R}^n$ such that $d_{ii} = x_i^\intercal Qx_i$.*

Corollary 6.1.2 *Let $Q \in \mathbb{Q}^{n\times n}$ be a symmetric positive semi-definite matrix. Then there exist a diagonal matrix $D \in \mathbb{Q}^{n\times n}$ with only non-negative entries and an invertible matrix $U \in \mathbb{Q}^{n\times n}$ such that $Q = U^\intercal DU$.*

Proof. This is an immediate consequence of Lemma 6.1.1, since $d_{ii} = x_i^\intercal Qx_i \geq 0$ for all i as Q is positive semi-definite. □

Thus, every convex quadratic objective function $z^\intercal Qz$ can be restated as $\sum_{i=1}^{s} \alpha_i(c_i^\intercal z)^2$ with $\alpha_i > 0$ and $c_i \in \mathbb{Z}^n$. Therefore, the test set approach from Chapter 3 is applicable to these problems using

$\mathcal{H}_{\text{CIP}}(A,C)$ and $f_i(x) = \alpha_i x_i^2$, $\alpha_i > 0$. Moreover, we should point out that $s \leq n$, that is, the Graver basis that has to be computed for $\mathcal{H}_{\text{CIP}}(A,C)$ involves at most $2n$ variables.

In the following, we will restrict our attention to quadratic 0-1 problems.

Corollary 6.1.3 *Any quadratic* 0-1 *optimization problem*

$$\min\{z^\intercal Qz + c^\intercal z : Az = b, z \in \{0,1\}^n\}$$

with symmetric matrix $Q \in \mathbb{Q}^{n\times n}$ can be rephrased as an equivalent problem

$$\min\{z^\intercal \overline{Q}z + \overline{c}^\intercal z : Az = b, z \in \{0,1\}^n\},$$

where $\overline{Q} \in \mathbb{Q}^{n\times n}$ is a symmetric, positive definite matrix.

Proof. As $z_i^2 - z_i = 0$ for $z \in \{0,1\}$, the given optimization problem is equivalent to

$$\min\{z^\intercal Qz + c^\intercal z + \lambda(z^\intercal I_n z - \mathbf{1}^\intercal z) : Az = b, z \in \{0,1\}^n\},$$

where $\lambda \in \mathbb{R}$ denotes some fixed scalar. As for sufficiently large $\lambda = \overline{\lambda} \in \mathbb{Q}_+$ the matrix $Q + \overline{\lambda}I_n$ becomes positive definite, Lemma 6.1.2 can be applied, giving the result with $\overline{Q} = Q + \overline{\lambda}I_n$ and $\overline{c} = c - \overline{\lambda}\mathbf{1}$. □

Consequently, *any* 0-1 quadratic optimization problem

$$\min\{z^\intercal Qz + c^\intercal z : Az = b, z \in \{0,1\}^n\}$$

can be written as

$$\min\left\{\sum_{i=1}^s \alpha_i (c_i^\intercal z)^2 + c^\intercal z : Az = b, z \in \{0,1\}^n\right\}$$

with $\alpha_i > 0$, and therefore, by using $\mathcal{H}_{\text{CIP}}(A,C)$, the test set approach from Chapter 3 can be applied. However, choosing different $\overline{\lambda}$ in the proof of Corollary 6.1.3, we get different equivalent formulations for the same problem $(\text{CIP})_{f,b}$. But as the following example shows, different problem formulations can lead to different test sets $\mathcal{H}_{\text{CIP}}(A,C)$ for the *same* problem. These sets, however, are test sets for two *different problem families* of which the given specific problem is a *common member.*

Example 6.1.4 Consider the quadratic 0-1 problem with $A = 0$ and

$$Q = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 2 \\ 1 & 2 & 0 \end{pmatrix}.$$

Since $A = 0$, we need to compute the Graver basis of $(C|I_3)$. But we have different choices for C. As $x_i^2 = x_i$, $i = 1,2,3$, we have

$$\begin{aligned} x^\intercal Qx &= 2x_1x_2 + 2x_1x_3 + 4x_2x_3 \\ &= (x_1+x_2+x_3)^2 + (x_2+x_3)^2 - x_1^2 - 2x_2^2 - 2x_3^2 \\ &= (x_1+x_2+x_3)^2 + (x_2+x_3)^2 - x_1 - 2x_2 - 2x_3 \end{aligned}$$

and

$$\begin{aligned} x^\intercal Q x &= 2x_1x_2 + 2x_1x_3 + 4x_2x_3 \\ &= (x_1 - 2x_2 + x_3)^2 + (3x_1 + x_2 + 4x_3)^2 + 12(x_1 - x_3)^2 - 22x_1^2 - 5x_2^2 - 29x_3^2 \\ &= (x_1 - 2x_2 + x_3)^2 + (3x_1 + x_2 + 4x_3)^2 + 12(x_1 - x_3)^2 - 22x_1 - 5x_2 - 29x_3. \end{aligned}$$

Therefore, the corresponding two matrices for the test set computations are

$$\left(\begin{array}{c|c} C' & I_2 \end{array} \right) = \left(\begin{array}{ccc|cc} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{array} \right)$$

and

$$\left(\begin{array}{c|c} C'' & I_3 \end{array} \right) = \left(\begin{array}{rrr|ccc} 1 & -2 & 1 & 1 & 0 & 0 \\ 3 & 1 & 4 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 1 \end{array} \right).$$

Using the software package 4ti2 [70], we obtain

$$\begin{aligned} \mathcal{H}_{\mathrm{CIP}}(A', C') &= \{(1,0,0),(0,1,0),(0,0,1),(1,-1,0),(0,1,-1),(1,0,-1)\} \\ \mathcal{H}_{\mathrm{CIP}}(A'', C'') &= \{(1,0,0),(0,1,0),(0,0,1),(1,1,0),(1,-1,0),(0,1,-1), \\ &\quad (0,1,1),(1,0,1),(1,0,-1),(1,1,-1),(1,1,1),(1,-1,1)\} \end{aligned}$$

Note that $\mathcal{H}_{\mathrm{CIP}}(A', C') \subsetneq \mathcal{H}_{\mathrm{CIP}}(A'', C'')$. $\square$

This gives us much freedom to rewrite particular 0-1 problems, possibly arriving at much smaller test sets for the same problem. As the following example shows, the same phenomenon happens also in the general (non-0-1) case.

Example 6.1.5 Consider the problem with $A = 0$ and

$$Q = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}.$$

Again, since $A = 0$, we need to compute the Graver basis of $(C|I_s)$ for some integer s, and as the following shows, we have more than one choice for C:

$$\begin{aligned} x^\intercal Q x &= 2x_1^2 + 2x_2^2 + 2x_3^2 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3 \\ &= (x_1 + x_2 + x_3)^2 + x_1^2 + x_2^2 + x_3^2 \\ &= (x_1 + x_2)^2 + (x_1 + x_3)^2 + (x_2 + x_3)^2 \end{aligned}$$

Corresponding to these two representations are the matrices

$$\left(\begin{array}{c|c} C' & I_4 \end{array} \right) = \left(\begin{array}{ccc|cccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right)$$

and

$$\left(\begin{array}{c|c} C'' & I_3 \end{array}\right) = \left(\begin{array}{ccc|ccc} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right).$$

Using `4ti2`, we obtain

$$\begin{aligned} \mathcal{H}_{\text{CIP}}(A', C') &= \{(1,0,0),(0,1,0),(0,0,1),(1,-1,0),(0,1,-1),(1,0,-1)\} \\ \mathcal{H}_{\text{CIP}}(A'', C'') &= \{(1,0,0),(0,1,0),(0,0,1),(1,-1,0),(1,0,-1),(0,1,-1), \\ &\quad (1,1,-1),(1,-1,1),(1,-1,-1)\} \end{aligned}$$

Note that again, $\mathcal{H}_{\text{CIP}}(A', C') \subsetneq \mathcal{H}_{\text{CIP}}(A'', C'')$. □

6.2 Proof of Theorem 6.0.3

In this section we prove the main theorem of this chapter, Theorem 6.0.3. First, we will collect some facts about Graver bases that will turn out very useful in the final proof.

Lemma 6.2.1 *Let $B = \begin{pmatrix} A & a & -a \end{pmatrix}$ be an integer matrix such that the two columns a and $-a$ differ only by a sign. Then the Graver basis of B can be constructed from the Graver basis of $B' = \begin{pmatrix} A & a \end{pmatrix}$ in the following way:*

$$\mathcal{G}_{IP}(B) = \{(u,v,w) : vw \leq 0, (u, v-w) \in \mathcal{G}_{IP}(B')\} \cup \{\pm(0,1,1)\}.$$

Proof. Let $(u,v,w) \in \mathcal{G}_{\text{IP}}(B)$. Since $\pm(0,1,1)$ are the only $\sqsubseteq$-minimal elements in $\ker_{\mathbb{Z}^{n+2}}(B)\setminus\{0\}$ with $u = 0$, we may assume that $u \neq 0$. Moreover, it holds $vw \leq 0$, since otherwise either $(u, v+1, w+1) \sqsubseteq (u,v,w)$ or $(u, v-1, w-1) \sqsubseteq (u,v,w)$ contradicts the $\sqsubseteq$-minimality of (u,v,w). Thus, without loss of generality, we assume in the following that $v \geq 0$ and $w \leq 0$. Next we show $(u, v-w) \in \mathcal{G}_{\text{IP}}(B')$.

Suppose that $(u, v-w) \notin \mathcal{G}_{\text{IP}}(B')$. Then there is some vector $(u', z') \in \mathcal{G}_{\text{IP}}(B')$ with $(u', z') \sqsubseteq (u, v-w)$ and $u' \neq 0$. (If $u' = 0$ then $z' = 0$ contradicting $(u', z') \in \mathcal{G}_{\text{IP}}(B')$ since $0 \notin \mathcal{G}_{\text{IP}}(B')$.) Of course, $(u', z') \neq (u, v-w)$, since we assume $(u', z') \in \mathcal{G}_{\text{IP}}(B')$ and $(u, v-w) \notin \mathcal{G}_{\text{IP}}(B')$. From $v - w \geq 0$ we get $0 \leq z' \leq v - w$. Next we show that $(u,v,w) \notin \mathcal{G}_{\text{IP}}(B)$ which will contradict our initial assumption $(u,v,w) \in \mathcal{G}_{\text{IP}}(B)$.

To prove this, note that $0 \leq \min(z', v) \leq v$ and $0 \geq -z' + \min(z', v) \geq w$. Whereas the first chain of inequalities holds because of $0 \leq z'$, the second can be seen as follows. If $z' \leq v$, we get $0 \geq -z' + z' = 0 \geq w$ by our assumption on w. If, on the contrary, $z' > v$ we get $0 \geq -z' + v \geq -(v-w) + v = w$, since $(u', z') \sqsubseteq (u, v-w)$ and thus $0 \leq z' \leq v - w$. But this implies that $(u', \min(z', v), -z' + \min(z', v)) \sqsubseteq (u,v,w)$. Moreover, we know that $u' \neq 0$ and $(u', \min(z', v), -z' + \min(z', v)) \in \ker_{\mathbb{Z}^{n+2}}(B)$. Thus, it remains to prove that $(u', \min(z', v), -z' + \min(z', v)) \neq (u,v,w)$, which implies that (u,v,w) is not $\sqsubseteq$-minimal in $\ker_{\mathbb{Z}^{n+2}}(B) \setminus \{0\}$ and therefore $(u,v,w) \notin \mathcal{G}_{\text{IP}}(B)$.

Suppose that $(u', \min(z', v), -z' + \min(z', v)) = (u, v, w)$ is true. This yields $u = u'$, $\min(z', v) = v$, and $w = -z' + \min(z', v) = -z' + v \geq -(v - w) + v = w$. But this implies that $z' = v - w$ and therefore $(u', z') = (u, v - w)$ in contrast to our assumption $(u', z') \neq (u, v - w)$ above. Thus $(u, v, w) \notin \mathcal{G}_{\text{IP}}(B)$. This shows the $\subseteq$-part of our equation.

It remains to prove that every vector (u, v, w) with $u \neq 0$, $v \geq 0$, $w \leq 0$, and $(u, v - w) \in \mathcal{G}_{\text{IP}}(B')$ belongs to $\mathcal{G}_{\text{IP}}(B)$. Clearly, $(u, v, w) \in \ker_{\mathbb{Z}^{n+2}}(B)$. Suppose that there exists a vector $(u', v', w') \in \ker_{\mathbb{Z}^{n+2}}(B)$ with $u' \neq 0$, $(u', v', w') \sqsubseteq (u, v, w)$, and $(u, v, w) \neq (u', v', w')$. But then we conclude $(u', v' - w') \in \ker_{\mathbb{Z}^{n+1}}(B')$ and $(u', v' - w') \sqsubseteq (u, v - w)$. If $(u, v - w) \neq (u', v' - w')$ was true, this would contradict $(u, v - w) \in \mathcal{G}_{\text{IP}}(B')$.

Therefore, suppose that we have $(u, v - w) = (u', v' - w')$, $(u', v', w') \sqsubseteq (u, v, w)$, and $(u, v, w) \neq (u', v', w')$. But then $0 \leq v' \leq v$ and $0 \geq w' \geq w$ together imply that $0 \leq v' - w' \leq v - w$. Since $u = u'$ and $(u, v, w) \neq (u', v', w')$, at least one of the inequalities $v' \leq v$ and $-w' \leq -w$ holds strictly. Therefore, $0 \leq v' - w' < v - w$, a contradiction to $v - w = v' - w'$. $\square$

Example 6.2.2 The Graver basis of $B' = (3\ 1)$ is $\mathcal{G}_{\text{IP}}(B') = \{\pm(1, -3)\}$. Thus, the Graver basis of $B = (3\ 1\ -1)$ is

$$\mathcal{G}_{\text{IP}}(B) = \{\pm(1, 0, 3), \pm(1, -1, 2), \pm(1, -2, 1), \pm(1, -3, 0), \pm(0, 1, 1)\},$$

as can easily be double-checked with `4ti2`. $\square$

A simple consequence of this lemma is

Corollary 6.2.3 *Let $B = \begin{pmatrix} A & a & a \end{pmatrix}$ be an integer matrix with two identical columns a. Then the Graver basis of B can be constructed from the Graver basis of $B' = \begin{pmatrix} A & a \end{pmatrix}$ in the following way:*

$$\mathcal{G}_{IP}(B) = \{(u, v, w) : vw \geq 0, (u, v + w) \in \mathcal{G}_{IP}(B')\} \cup \{\pm(0, 1, -1)\}.$$

Proof. The claim follows immediately from the fact that (u, v, w) is $\sqsubseteq$-minimal in $\ker \begin{pmatrix} A & a & a \end{pmatrix}$ if and only if $(u, v, -w)$ is $\sqsubseteq$-minimal in $\ker \begin{pmatrix} A & a & -a \end{pmatrix}$. $\square$

The following is an immediate consequence of Lemma 6.2.1 and of Corollary 6.2.3.

Lemma 6.2.4 *Let $A \in \mathbb{Z}^{d \times n}$ and let $B = \begin{pmatrix} A & a & \ldots & a & -a & \ldots & -a \end{pmatrix}$ be an integer matrix with finitely many multiple columns a and $-a$ which differ only in their signs. Then we have $\phi_n(\mathcal{G}_{IP}(B)) = \phi_n\left(\mathcal{G}_{IP}\begin{pmatrix} A & a \end{pmatrix}\right) \cup \{0\}$.*

Proof. The constructions in Lemma 6.2.1 and in Corollary 6.2.3 satisfy

$$\phi_n\left(\mathcal{G}_{\text{IP}}\begin{pmatrix} A & a & -a \end{pmatrix}\right) = \phi_n\left(\mathcal{G}_{\text{IP}}\begin{pmatrix} A & a \end{pmatrix}\right) \cup \{0\}$$

and

$$\phi_n\left(\mathcal{G}_{\text{IP}}\begin{pmatrix} A & a & a \end{pmatrix}\right) = \phi_n\left(\mathcal{G}_{\text{IP}}\begin{pmatrix} A & a \end{pmatrix}\right) \cup \{0\}.$$

Putting both constructions together iteratively, we get

$$\phi_n(\mathcal{G}_{\mathrm{IP}}(B)) = \phi_n\left(\mathcal{G}_{\mathrm{IP}}\begin{pmatrix} A & a \end{pmatrix}\right) \cup \{0\},$$

as claimed. □

Thus, in order to compute $\phi_n(\mathcal{G}_{\mathrm{IP}}(B))$, it suffices to compute $\phi_n\left(\mathcal{G}_{\mathrm{IP}}\begin{pmatrix} A & a \end{pmatrix}\right)$. The following is an immediate consequence to Lemma 6.2.4.

Corollary 6.2.5 *Let $A \in \mathbb{Z}^{d\times n}$, $c_1, \ldots, c_s \in \mathbb{Z}^n$, and $k \in \mathbb{Z}_{>0}$. Denote by C the $s\times n$ matrix whose rows are formed by the vectors $c_1^\intercal, \ldots, c_s^\intercal$, by the bold letter $\mathbf{1}$ the vector in R^k with all entries 1, and by I_s the $s \times s$ unit matrix. Then*

$$\phi_n(\mathcal{G}_{IP}(A_k)) = \phi_n\left(\mathcal{G}_{IP}\begin{pmatrix} A & 0 \\ C & I_s \end{pmatrix}\right) \cup \{0\},$$

where

$$A_k := \begin{pmatrix} A & & & & & & & \\ c_1^\intercal & -\mathbf{1} & \mathbf{1} & & & & & \\ c_2^\intercal & & & -\mathbf{1} & \mathbf{1} & & & \\ & & & & & \ddots & \ddots & & \\ c_s^\intercal & & & & & & & -\mathbf{1} & \mathbf{1} \end{pmatrix}.$$

Before we come to the proof of our main theorem of this chapter, let us prove two more useful facts.

Lemma 6.2.6 *Let g be a $\mathbb{Z}$-convex function with minimum at 0. Then for fixed $p \in \mathbb{Z}$ and for fixed $k \geq |p|$, an optimal solution to*

$$\min\{\sum_{j=1}^{k} \quad (g(j) - g(j-1))x_{i,j} + (g(-j) - g(-j+1))y_{i,j} : \\ p = \sum_{j=1}^{k} x_j - \sum_{j=1}^{k} y_j, \quad x_j, y_j \in \{0,1\}, \quad j = 1, \ldots, k\},$$

is given by

$$\begin{array}{ll} x_1 = \ldots = x_p = 1, x_{p+1} = \ldots = x_k = y_1 = \ldots = y_k = 0, & \text{if} \quad p > 0, \\ x_1 = \ldots = x_k = y_1 = \ldots = y_k = 0, & \text{if} \quad p = 0, \\ y_1 = \ldots = y_{-p} = 1, y_{-p+1} = \ldots = y_k = x_1 = \ldots = x_k = 0, & \text{if} \quad p < 0. \end{array}$$

The optimal value in each of these three cases is $g(p) - g(0)$.

Proof. The case $p = 0$ is trivial and the optimal objective value is $0 = g(0) - g(0)$.

Let us now consider the case $p > 0$. Clearly, since $p > 0$, some x_i must be positive. Suppose that in a minimal solution we have $x_i = 1$ and $y_j = 1$ for some i and some j. This cannot happen, since by putting $x_i = 0$ and $y_j = 0$ we would arrive at a solution with smaller objective value, as all coefficients in the objective function are positive. Thus, in a minimal solution $y_1 = \ldots = y_k = 0$.

Since g is a $\mathbb{Z}$-convex function with minimum at 0, the coefficients $g(j) - g(j-1)$ in the objective function are non-negative and form a non-decreasing sequence as $j > 0$ increases. Thus,

$$x_1 = \ldots = x_p = 1, x_{p+1} = \ldots = x_k = 0$$

leads to a minimal objective value. This value is

$$\sum_{j=1}^{p}(g(j) - g(j-1)) = g(p) - g(0).$$

For the case $p < 0$ we conclude analogously that $x_1 = \ldots = x_k = 0$. Moreover, since g is a $\mathbb{Z}$-convex function with minimum at 0, the coefficients $g(-j) - g(-j+1)$ in the objective function are non-negative and form a non-decreasing sequence as $j > 0$ increases. As above, this implies that

$$y_1 = \ldots = y_{-p} = 1, y_{-p-1} = \ldots = y_k = 0$$

leads to a minimal objective value. This value is again

$$\sum_{j=1}^{-p}(g(-j) - g(-j+1)) = g(p) - g(0)$$

and the claim is proved. □

Corollary 6.2.7 *Let $f_1, \ldots, f_s$ be $\mathbb{Z}$-convex functions with minimum at 0, $A \in \mathbb{Z}^{d\times n}$, $b \in \mathbb{Z}^d$, $c \in \mathbb{R}^n$, $c_1, \ldots, c_s \in \mathbb{Z}^n$, and $c_{1,0}, \ldots, c_{s,0} \in \mathbb{Z}$ be given. Then for fixed $z \in \mathbb{Z}^n$ and for fixed $k \geq \max\{|c_i^\intercal z + c_{i,0}|, i = 1, \ldots, s\}$, the optimal value of*

$$\begin{array}{lll} \min\{\sum\limits_{i=1}^{s}\sum\limits_{j=1}^{k} & (f_i(j) - f_i(j-1))x_{i,j} + (f_i(-j) - f_i(-j+1))y_{i,j} + c^\intercal z : & \\ & Az = b, & z \in \mathbb{Z}_+^n, \\ & c_i^\intercal z + c_{i,0} = \sum\limits_{j=1}^{k} x_{i,j} - \sum\limits_{j=1}^{k} y_{i,j}, & i = 1, \ldots, s, \\ & x_{i,j}, y_{i,j} \in \{0,1\}, & i = 1, \ldots, s, \\ & & j = 1, \ldots, k\}. \end{array}$$

is $f(z) - \sum\limits_{i=1}^{s} f_i(0)$, where

$$f(z) := \sum_{i=1}^{s} f_i(c_i^\intercal z + c_{i,0}) + c^\intercal z.$$

Proof. Since z is fixed, the problem decomposes into s smaller problems for which we can apply Lemma 6.2.6. Thus, the optimal value of the given problem is

$$\sum_{i=1}^{s}[f_i(c_i^\intercal z + c_{i,0}) - f_i(0)] + c^\intercal z = \sum_{i=1}^{s} f_i(c_i^\intercal z + c_{i,0}) + c^\intercal z - \sum_{i=1}^{s} f_i(0) = f(z) - \sum_{i=1}^{s} f_i(0).$$ □

Now let us finally prove our main theorem of this chapter, Theorem 6.0.3.

Proof. In order to prove this claim, assume that we are given $\mathbb{Z}$-convex functions $f_1, \ldots, f_s$ with minimum at 0, $c_{1,0}, \ldots, c_{s,0} \in \mathbb{Z}$, $b \in \mathbb{Z}^d$, and $c \in \mathbb{R}^n$. Moreover, assume that we are given a non-optimal feasible solution z_0 to $Az = b$, $z \in \mathbb{Z}^n_+$.

The theorem is proved if we can find some vector $t \in \mathcal{H}_{\text{CIP}}(A, C)$ such that $z_0 - t$ is feasible and such that $f(z_0 - t) < f(z_0)$. In the following, we construct such a vector t.

Since we assume z_0 to be non-minimal, there exists some better feasible solution z_1, say. Let

$$k := \max\{|c_i^\intercal z_0 + c_{i,0}|, |c_i^\intercal z_1 + c_{i,0}|, i = 1, \ldots, s\}$$

and consider the auxiliary integer linear program

$$\begin{array}{lll}
\text{(AIP)}: \min\{\sum\limits_{i=1}^{s}\sum\limits_{j=1}^{k} & (f_i(j) - f_i(j-1))x_{i,j} + (f_i(-j) - f_i(-j+1))y_{i,j} + c^\intercal z : & \\
& Az = b, & z \in \mathbb{Z}^n_+, \\
& c_i^\intercal z + c_{i,0} = \sum\limits_{j=1}^{k} x_{i,j} - \sum\limits_{j=1}^{k} y_{i,j}, & i = 1, \ldots, s, \\
& x_{i,j}, y_{i,j} \in \{0, 1\}, & i = 1, \ldots, s, \\
& & j = 1, \ldots, k\}.
\end{array}$$

By Lemma 6.2.6 and Corollary 6.2.7, the minimal values of (AIP) for fixed $z = z_0$ and for fixed $z = z_1$ are $f(z_0) - f_0$ and $f(z_1) - f_0$, where $f_0 = \sum\limits_{i=1}^{s} f_i(0)$. By (z_0, x_0, y_0) and (z_1, x_1, y_1) denote feasible solutions of (AIP) that achieve these values.

As $f(z_0) > f(z_1)$ by assumption, we have $f(z_0) - f_0 > f(z_1) - f_0$. Thus, (z_0, x_0, y_0) is a feasible solution of (AIP) that is not optimal. Therefore, there must exist some vector (t, u, v) in the Graver basis associated with the problem matrix of (AIP) that improves (z_0, x_0, y_0). Clearly, $t \neq 0$, since (z_0, x_0, y_0) is optimal for fixed z_0. We will now show that $t \in \mathcal{H}_{\text{CIP}}(A, C)$, that $z_0 - t$ is feasible for $(\text{CIP})_{f,b}$, and that $f(z_0 - t) < f(z_0)$. The claim then follows immediately, as these are the three conditions for an improving vector t.

The problem matrix associated to (AIP) is

$$A_k := \begin{pmatrix}
A & & & & & & & \\
c_1^\intercal & \mathbf{-1} & \mathbf{1} & & & & & \\
c_2^\intercal & & & \mathbf{-1} & \mathbf{1} & & & \\
 & & & & & \ddots & \ddots & \\
c_s^\intercal & & & & & & & \mathbf{-1} \quad \mathbf{1}
\end{pmatrix},$$

where

$$\phi_n(\mathcal{G}_{\text{IP}}(A_k)) = \phi_n\left(\mathcal{G}_{\text{IP}}\begin{pmatrix} A & 0 \\ C & I_s \end{pmatrix}\right) \cup \{0\} = \mathcal{H}_{\text{CIP}}(A, C) \cup \{0\},$$

by Corollary 6.2.5. Therefore, $(t, u, v) \in \mathcal{G}_{\text{IP}}(A_k)$ satisfies $t \in \mathcal{H}_{\text{CIP}}(A, C) \cup \{0\}$. Since $t \neq 0$, we have $t \in \mathcal{H}_{\text{CIP}}(A, C)$. Moreover, as $(z_0, x_0, y_0) - (t, u, v)$ is feasible for (AIP), we must have $A(z_0 - t) = b$ and $z_0 - t \geq 0$, implying that $z_0 - t$ is feasible for $(\text{CIP})_{f,b}$.

It remains to show $f(z_0 - t) < f(z_0)$. Let $(z_0 - t, x_2, y_2)$ be a feasible solution of (AIP) that achieves the minimal value $f(z_0 - t) - f_0$ of (AIP) for fixed $z = z_0 - t$, see Lemma 6.2.6 and Corollary 6.2.7 for its existence and construction. Clearly, this minimal objective value of (AIP) for fixed $z = z_0 - t$ is less than or equal to the objective value $\text{cost}(z_0 - t, x_0 - u, y_0 - v)$ of $(z_0, x_0, y_0) - (t, u, v)$ in (AIP), that is, $f(z_0 - t) - f_0 = \text{cost}(z_0 - t, x_2, y_2) \leq \text{cost}(z_0 - t, x_0 - u, y_0 - v)$. By construction, we have $\text{cost}(z_0 - t, x_0 - u, y_0 - v) < \text{cost}(z_0, x_0, y_0) = f(z_0) - f_0$. Therefore, $f(z_0 - t) - f_0 < f(z_0) - f_0$ and consequently $f(z_0 - t) - f(z_0)$. $\square$

Chapter 7

Computation of generating sets of lattice ideals

7.1 Introduction

In this chapter, we present a new algorithm due to Malkin [75] for computing a *generating set* of a *lattice ideal*

$$I(\mathcal{L}) := \langle x^{u^+} - x^{u^-} : u \in \mathcal{L} \rangle \subseteq k[x_1, ..., x_n],$$

where k is a field, $\mathcal{L}$ is a sub-lattice of $\mathbb{Z}^n$ with $\mathcal{L} \cap \mathbb{Z}^n_+ = \{0\}$, and

$$x^{u^+} - x^{u^-} := x_1^{u_1^+} x_2^{u_2^+} \cdots x_n^{u_n^+} - x_1^{u_1^-} x_2^{u_2^-} \cdots x_n^{u_n^-}.$$

Generating set and Gröbner basis computations for general ideals are usually very time consuming. Fortunately, in the special setting of lattice ideals, many improvements are possible. Two interesting areas of application of lattice ideals are algebraic statistics and integer programming. We chose to include the presentation of this algorithm into this thesis

- because it complements the project-and-lift approach to compute Graver bases, see Chapter 2,
- because Gröbner bases of lattice ideals are test sets for families of integer linear programs with fixed cost function, see Chapter 3,
- because Gröbner bases of lattice ideals will be used for counting lattice points in polytopes, see Chapter 8, and
- because we present an efficient encoding these Gröbner bases, see Chapter 13.

In general, a generating set of $I(\mathcal{L})$ is *not* readily available. For a basis S of the lattice $\mathcal{L}$ over $\mathbb{Z}^n$, the ideal $J(S) := \langle x^{u^+} - x^{u^-} : u \in S \rangle$ satisfies $J(S) \subseteq I(\mathcal{L})$, but usually one may not expect that

$J(S) = I(\mathcal{L})$. Also, when computing a Gröbner basis of a lattice ideal, computational experiments show that when $\mathcal{L} \cap \mathbb{Z}^n_+ = \{0\}$, the computation of a generating set usually takes much longer than computing the Gröbner basis from the generating set.

Generating sets of lattice ideals and Gröbner bases of lattice ideals have corresponding geometric concepts (see [116, 118, 120]), which we call *generating sets of lattices* and *Gröbner bases of lattices* respectively (see Section 7.2). These concepts are related as follows: if a set $S \subseteq \mathcal{L}$ is a generating set of $\mathcal{L}$ or a Gröbner basis of $\mathcal{L}$ with respect to a term order $\succ$, then $G := \{x^{u^+} - x^{u^-} : u \in S\}$ is respectively a generating set of $I(\mathcal{L})$ or a Gröbner basis of $I(\mathcal{L})$ with respect to $\succ$; and also, if a set of monic binomials G is a generating set of $I(\mathcal{L})$ or a Gröbner basis of $I(\mathcal{L})$ with respect to a term order $\succ$, then $S := \{\alpha - \beta : x^\alpha - x^\beta \in G\}$ is respectively a generating set of $\mathcal{L}$ or a Gröbner basis of $\mathcal{L}$ with respect to $\succ$. Note that we use the same order $\succ$ for monomials and vectors via the relation $x^\alpha \succ x^\beta$ if and only if $\alpha \succ \beta$ for $\alpha, \beta \in \mathbb{Z}^n_+$. Also, note that any minimal reduced Gröbner basis of a lattice ideal is a set of monic binomials.

In this chapter, we present existing theory and the new algorithm only in a geometric framework following the approach in [116], [118], and [120], since for lattice ideals, we prefer the geometric approach to the algebraic one. However, note that for every geometric concept presented, there exists an equivalent algebraic notion, although we do not present it here. Also, we have tried to make this chapter reasonably self contained, so for completeness, we present geometric proofs of existing results where pertinent.

Recently, there has been renewed interest in toric ideal computations because of applications in algebraic statistics. Here, we are interested in Markov bases, which are used in a Monte-Carlo Markov-Chain (MCMC) process to test validity of statistical models via sampling. Diaconis and Sturmfels [50] showed that a (preferably minimal) generating set of a lattice

$$\mathcal{L}_A := \{u \in \mathbb{Z}^n : Au = 0\}$$

for some matrix $A \in \mathbb{Z}^{d \times n}$ is a Markov basis. Note that $I(\mathcal{L}_A)$ is a toric ideal. However, at that time, no effective implementation of an algorithm to compute generating sets of toric ideals was available that could deal with moderate size problems in $50 - 100$ variables. This situation has changed by now: several such implementations are available. Using `4ti2` [70], Eriksson even reports, in [56], on successful computations of Gröbner bases and Markov bases of toric ideals in $2,048$ variables. His problems arise from phylogenetic trees in computational biology.

In integer programming, test sets of integer programs correspond to Gröbner bases of lattices (or lattice ideals). Test sets were introduced in [65]. Consider the general linear integer program

$$\min\{c^\intercal z_{\overline{\sigma}} : Az = b, z_{\overline{\sigma}} \geq 0, z \in \mathbb{Z}^n\}$$

where $c \in \mathbb{Q}^{|\overline{\sigma}|}$, $A \in \mathbb{Z}^{d \times n}$, $b \in \mathbb{Z}^d$, $\sigma \subseteq \{1, ..., n\}$, $\overline{\sigma} := \{1, ..., n\} \setminus \sigma$, and where $z_{\overline{\sigma}}$ is the set of variables indexed by $\overline{\sigma}$. Any integer program that has an optimal solution can be written in this form [34]. By projecting onto the $\overline{\sigma}$ variables, we can rewrite these integer programs in the equivalent and more convenient form

$$(\text{IP})^\sigma_{A,c,b} = \min\{c^\intercal z : A_{\overline{\sigma}} z \equiv b \pmod{A_\sigma \mathbb{Z}}, z \in \mathbb{Z}^{|\overline{\sigma}|}_+\},$$

where A_σ and $A_{\overline{\sigma}}$ are the sub-matrices of A whose columns are indexed by σ and $\overline{\sigma}$ respectively, and $A_\sigma\mathbb{Z} := \{A_\sigma z : z \in \mathbb{Z}^{|\sigma|}\}$. In the special case where $\sigma = \emptyset$, we set $A_\sigma\mathbb{Z} := \{0\}$, and the problem $(\mathrm{IP})^{\emptyset}_{A,c,b}$ simplifies to $(\mathrm{IP})_{A,c,b} := \min\{c^\intercal z : Az = b, z \in \mathbb{Z}^n_+\}$. Note that group relaxations and extended group relaxations of $(\mathrm{IP})_{A,c,b}$ are also of the form $(\mathrm{IP})^{\sigma}_{A,\overline{c},b}$ for some cost vector $\overline{c} \in \mathbb{Q}^{|\overline{\sigma}|}$ [80]. Without loss of generality, we assume that c is *generic* meaning that $(\mathrm{IP})^{\sigma}_{A,c,b}$ has a unique optimal solution for every feasible $b \in \mathbb{Z}^d$. We can always easily perturb a given c so that it is generic.

As defined in Chapter 3, a set $T \subseteq \mathbb{Z}^{|\overline{\sigma}|}$ is called a *test set* for $(\mathrm{IP})^{\sigma}_{A,c,b}$ if T contains an improving direction t for every non-optimal feasible solution $z \in \mathbb{Z}^{|\overline{\sigma}|}_+$ of $(\mathrm{IP})^{\sigma}_{A,c,b}$; that is, $z - t$ is also feasible and $c^\intercal(z-t) < c^\intercal z$. Clearly, $z - t$ being feasible implies that t is an element of the lattice

$$\mathcal{L}^{\sigma}_A := \{u \in \mathbb{Z}^{|\overline{\sigma}|} : A_{\overline{\sigma}} u \equiv 0 \pmod{A_\sigma\mathbb{Z}}\}.$$

Moreover, a set $T \subseteq \mathcal{L}^{\sigma}_A$ is called a test set for $(\mathrm{IP})^{\sigma}_{A,c}$ the class of integer programs $(\mathrm{IP})^{\sigma}_{A,c,b}$ for all $b \in \mathbb{Z}^d$, if T is a test set for every integer program in $(\mathrm{IP})^{\sigma}_{A,c}$. Graver showed that there exist *finite* sets T that are test sets for $(\mathrm{IP})_{A,c}$ ($\sigma = \emptyset$). In fact, his sets also constitute finite test sets for $(\mathrm{IP})^{\sigma}_{A,c}$ for arbitrary σ. Having a finite test set available, an optimal solution of $(\mathrm{IP})^{\sigma}_{A,c,b}$ can be found by iteratively improving any given non-optimal solution of $(\mathrm{IP})^{\sigma}_{A,c,b}$, see Chapter 3.

In [34] and [115], it was shown that given a generic cost vector c and a term order $\succ$ where c and $\succ$ are compatible, a set $S \subseteq \mathcal{L}^{\sigma}_A$ is a Gröbner basis of $\mathcal{L}^{\sigma}_A$ with respect to a $\succ$ if and only if S is a test set for $(\mathrm{IP})^{\sigma}_{A,c}$ where c and $\succ$ are compatible if $c^\intercal\alpha > c^\intercal\beta$ implies $\alpha \succ \beta$ for all $\alpha, \beta \in \mathbb{Z}^n_+$. A compatible term ordering $\succ$ exists for every generic c, and a compatible generic c exists for every term ordering $\succ$. Additionally, any lattice $\mathcal{L}$ can be written in the form $\mathcal{L}^{\sigma}_A$ for some matrix $A \in \mathbb{Z}^{n\times d}$ and some index set $\sigma \subseteq \{1, ..., n\}$, and so, Gröbner bases of lattices and test sets of integer programs really are equivalent concepts.

We define generating sets and Gröbner bases of lattices, in Section 7.2, in a geometric context and present the completion procedure [27, 26, 36], which is the main building block for the algorithms for computing generating sets.

In Section 7.3, we present the two main existing algorithms for computing generating sets: the algorithm of Hoşten and Sturmfels in [78], which we call the “Saturation” algorithm; and the algorithm of Bigatti, LaScala, and Robbiano in [21], which we call the “Lift-and-Project” algorithm. We also describe our new algorithm for computing generating sets: the “Project-and-Lift” algorithm. The Saturation Algorithm is based upon the result that $I(\mathcal{L}) = (\ldots((J(S) : x_1^\infty) : x_2^\infty)\ldots) : x_n^\infty$ where S is a lattice basis of $\mathcal{L}$ and $J(S)$ is defined as above. Using this result, we can compute a generating set of $I(\mathcal{L})$ from S via a sequence of *saturation* steps where each individual saturation step is performed via the completion procedure. The Lift-and-Project Algorithm is based upon the related result that $I(\mathcal{L}) = J(S) : (x_1 \cdot x_2 \cdot \ldots \cdot x_n)^\infty$. Here, a generating set is computed via the completion procedure using an additional variable. The Project-and-Lift Algorithm is strongly related to the Saturation Algorithm; however, the computational speed-up is enormous as will be seen in Section 7.7. In contrast with the Saturation Algorithm, which performs saturation steps in the original space of the lattice $\mathcal{L}$, the Project-and-Lift Algorithm performs saturation steps in projected subspaces of $\mathcal{L}$ and then lifts the result back into the original space.

We are mainly interested in computing a generating set of $\mathcal{L}$ where $\mathcal{L} \cap \mathbb{Z}_+^n = \{0\}$. However in Section 7.4, we address the question of how to compute a generating set $\mathcal{L}$ if $\mathcal{L} \cap \mathbb{Z}_+^n \neq \{0\}$. We demonstrate that the above methods for the case where $\mathcal{L} \cap \mathbb{Z}_+^n = \{0\}$ can be extended to this more general case; it happens to be more straight-forward in some ways.

The completion procedure as it is presented in Section 7.2 is not very efficient. In Section 7.5, we show how to increase the efficiency of the completion procedure. All the results in this section, which we present in a geometric framework, have corresponding results in an algebraic context. This section is rather technical and no other section depends upon it, so it may be skipped on first reading.

In Section 7.6, we give the solution of a computational challenge posed by Seth Sullivant to compute the Markov basis of $4 \times 4 \times 4$ tables with 2-marginals, a problem involving 64 variables. We solved this with the help of the new algorithm. Our computations led to $148,968$ elements in the minimal generating set of $I(A)$ which fall into 15 equivalence classes with respect to the underlying symmetry group $S_4 \times S_4 \times S_4 \times S_3$.

In Section 7.7, we compare the performance of the implementation of the Project-and-Lift algorithm in `4ti2` v.1.2 [70] with the implementation of the Saturation algorithm and the Lift-and-Project algorithm in Singular v3.0.0 [66] and in CoCoA 4.2 [30]. The Project-and-Lift algorithm is significantly faster than the other algorithms.

7.2 Generating sets and Gröbner bases

Given a lattice $\mathcal{L} \subseteq \mathbb{Z}^n$, and a vector $b \in \mathbb{Z}^n$, we define

$$\mathcal{F}_{\mathcal{L},b} := \{x : x \equiv b \pmod{\mathcal{L}}, x \in \mathbb{Z}_+^n\}.$$

For $S \subseteq \mathcal{L}$, we define $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$ to be the *undirected* graph with nodes $\mathcal{F}_{\mathcal{L},b}$ and edges (x,y) if $x - y \in S$ or $y - x \in S$ for $x, y \in \mathcal{F}_{\mathcal{L},b}$.

Definition 7.2.1 *A set $S \subseteq \mathcal{L}$ a **generating set of** $\mathcal{L}$ if the graph $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$ is connected for every $b \in \mathbb{Z}^n$.*

We remind the reader that connectedness of $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$ simply states that between each pair $x, y \in \mathcal{F}_{\mathcal{L},b}$, there exists a path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$. Note the difference between a *generating set* of a lattice and a *spanning set* of a lattice: a *spanning set of* $\mathcal{L}$ is any set $S \subseteq \mathcal{L}$ such that any point in $\mathcal{L}$ can be represented as an linear integer combination of the vectors in S. A generating set of $\mathcal{L}$ is a spanning set of $\mathcal{L}$, but the converse is not necessarily true.

Recall that for any lattice $\mathcal{L}$, we have $\mathcal{L} = \mathcal{L}_A^\sigma$ for some matrix $A \in \mathbb{Z}^{n \times d}$ and some index set $\sigma \subseteq \{1, ..., n\}$. Hence,

$$\mathcal{F}_{\mathcal{L},b} = \mathcal{F}_{A,\overline{b}}^\sigma := \{x \in \mathbb{Z}_+^{|\overline{\sigma}|} : A_{\overline{\sigma}} x \equiv \overline{b} \pmod{A_\sigma \mathbb{Z}}\}$$

for all $b \in \mathbb{Z}^n$ and all $\overline{b} \in \mathbb{Z}^d$ where $\overline{b} = A_{\overline{\sigma}} b$. So, $\mathcal{F}_{\mathcal{L},b}$ and $\mathcal{F}_{A,\overline{b}}^\sigma$ are dual representations of feasible sets.

Example 7.2.2 Let $S := \{(1,-1,-1,-3,-1,2),(1,0,2,-2,-2,1)\}$, and let $\mathcal{L} \subseteq \mathbb{Z}^6$ be the lattice spanned by S. So, by definition, S is a spanning set of $\mathcal{L}$, but S is not a generating set of $\mathcal{L}$. Observe that $\mathcal{L} = \mathcal{L}_A$ where

$$A = (\tilde{A}, I),\ \tilde{A} = \begin{pmatrix} -2 & -3 \\ +2 & -1 \\ +2 & +1 \\ -1 & +1 \end{pmatrix},\ \text{and } I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

So, for every $b \in \mathbb{Z}^6$, $\mathcal{F}_{\mathcal{L},b} = \mathcal{F}_{A,\overline{b}} = \{(x,s) : \tilde{A}x + Is = \overline{b}, x \in \mathbb{Z}^2_+, s \in \mathbb{Z}^4_+\}$ where $\overline{b} = Ab \in \mathbb{Z}^4$. Hence, the projection of $\mathcal{F}_{\mathcal{L},b}$ onto the (x_1,x_2)-plane is the set of integer points in the polyhedron $\{x \in \mathbb{R}^n_+ : Ax \leq \overline{b}\}$, and the s variables are the slack variables. Consider $b := (2,2,4,2,4,1)$; then, $\mathcal{F}_{\mathcal{L},b} = \mathcal{F}_{A,\overline{b}}$ where $\overline{b} = Ab = (-6,4,10,1)$ (see Figure 7.1a).

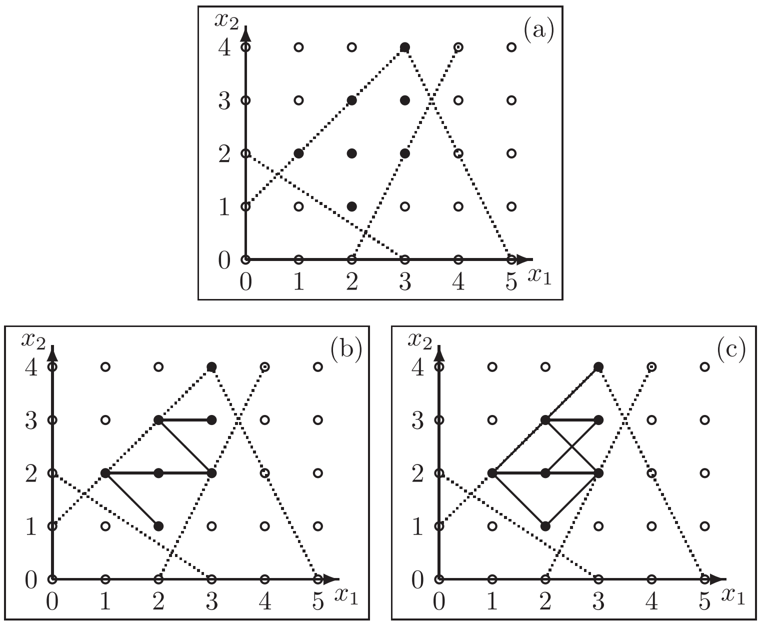


Figure 7.1: The set $\mathcal{F}_{\mathcal{L},b}$ and the graphs $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$ and $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S')$ projected onto the (x_1,x_2)-plane.

The graph of $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$ is not connected because the point $(3,4,12,2,0,0) \in \mathcal{F}_{\mathcal{L},b}$ is disconnected (see Figure 7.1b). Consider the enlarged set $S' := S \cup \{(1,1,5,-1,-3,0)\}$. The graph of $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S')$ is now connected (see Figure 7.1c); however, S' is still not a generating set of $\mathcal{L}$ since we have $\mathcal{F}_{\mathcal{L},b'} = \{(0,0,0,0,1,1),(0,1,3,1,0,0)\}$ for $b' := (0,0,0,0,1,1)$, and the graph $\mathcal{G}(\mathcal{F}_{\mathcal{L},b'}, S')$ is disconnected; since there are only two feasible points in $\mathcal{F}_{\mathcal{L},b'}$, the vector between them $(0,1,3,1,-1,-1)$ must be in any generating of $\mathcal{L}$. Finally, $S'' := S' \cup \{(0,1,3,1,-1,-1)\}$ is a generating set of $\mathcal{L}$. □

For the definition of a Gröbner basis, we need a term ordering $\succ$ for $\mathcal{L}$. We call $\succ$ a *term ordering* for $\mathcal{L}$ if $\succ$ is a total well-ordering on $\mathcal{F}_{\mathcal{L},b}$ for every $b \in \mathbb{Z}^n$ and $\succ$ is an additive ordering meaning that for all $b \in \mathbb{Z}^n$ and for all $x, y \in \mathcal{F}_{\mathcal{L},b}$, if $x \succ y$, then $x + \gamma \succ y + \gamma$ for every $\gamma \in \mathbb{Z}^n_+$ (note that $x + \gamma, y + \gamma \in \mathcal{F}_{\mathcal{L},b+\gamma}$). We also need the notion of a decreasing path: a path $(x^0, \ldots, x^k)$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ is *$\succ$-decreasing* if $x^i \succ x^{i+1}$ for $i = 0, \ldots, k-1$. We define $\mathcal{L}_\succ := \{u \in \mathcal{L} : u^+ \succ u^-\}$.

Definition 7.2.3 *A set $G \subseteq \mathcal{L}_\succ$ is a $\succ$-**Gröbner basis** of $\mathcal{L}$ if for every $x \in \mathbb{Z}^n_+$ there exists a $\succ$-decreasing path in $\mathcal{G}(\mathcal{F}_{\mathcal{L},x}, G)$ from x to the unique $\succ$-minimal element in $\mathcal{F}_{\mathcal{L},x}$.*

If $G \subseteq \mathcal{L}_\succ$ is a $\succ$-Gröbner basis, then G is a generating set of $\mathcal{L}$ since given $x, y \in \mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ for some $b \in \mathbb{Z}^n$, there exists a $\succ$-decreasing path from x to the unique $\succ$-minimal element in $\mathcal{F}_{\mathcal{L},b}$ and from y to the same element, and thus, x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$. Also, $G \subseteq \mathcal{L}_\succ$ is a Gröbner basis if and only if for every $x \in \mathbb{Z}^n_+$, x is either the unique $\succ$-minimal element in $\mathcal{F}_{\mathcal{L},b}$ or there exists a vector $u \in G$ such that $x - u \in \mathcal{F}_{\mathcal{L},b}$ and $x \succ x - u$; consequently, a Gröbner basis G is a test set for $(\mathrm{IP})^\sigma_{A,c}$ where $\mathcal{L}^\sigma_A = \mathcal{L}$ if c and $\succ$ are compatible.

The defining property of a Gröbner basis is very strong, so we redefine it in terms of reduction paths. A path $(x^0, \ldots, x^k)$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ is a *$\succ$-reduction path* if for *no* $i \in \{1, \ldots, k-1\}$, we have $x^i \succ x^0$ and $x^i \succ x^k$. For example, see Figure 7.2.

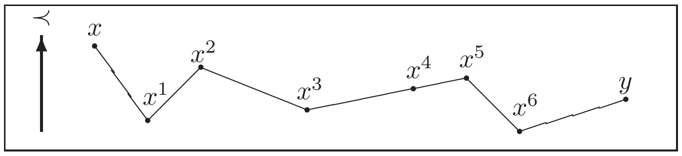


Figure 7.2: Reduction path between x and y.

Lemma 7.2.4 *A set $G \subseteq \mathcal{L}_\succ$ is a $\succ$-Gröbner basis of $\mathcal{L}$ if and only if for each $b \in \mathbb{Z}^n$ and for each pair $x, y \in \mathcal{F}_{\mathcal{L},b}$, there exists a $\succ$-reduction path in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ between x and y.*

Proof. If $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ contains $\succ$-decreasing paths from $x, y \in \mathcal{F}_{\mathcal{L},b}$ to the unique $\succ$-minimal element in $\mathcal{F}_{\mathcal{L},b}$, then joining the two paths (and removing cycles if necessary) forms a $\succ$-reduction path between x and y.

For the other direction, we assume that there is a $\succ$-reduction path between each pair $x, y \in \mathcal{F}_{\mathcal{L},b}$. Denote by x^* the unique $\succ$-minimal element in $\mathcal{F}_{\mathcal{L},b}$; thus, every $x \in \mathcal{F}_{\mathcal{L},b}$ is connected to x^* by a $\succ$-reduction path. In particular, by the definition of a $\succ$-reduction path, if $x \neq x^*$, then the first node $x^1 \neq x$ in this path must satisfy $x \succ x^1$. Repeating this argument iteratively with x^1 instead of x, we get a $\succ$-decreasing path from x to x^*. This follows from the fact that $\succ$ is a term ordering, which implies that every $\succ$-decreasing path must be finite. However, the only node from which the $\succ$-decreasing path cannot be lengthened is x^*. $\square$

Checking for a given $G \subseteq \mathcal{L}_\succ$ whether there exists a $\succ$-reduction path in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ for every $b \in \mathbb{Z}^n$ and for each pair $x, y \in \mathcal{F}_{\mathcal{L},b}$ involves infinitely many situations that need to be checked.

In fact, far fewer checks are needed: we only need to check for a $\succ$-reduction path from x to y if there exists a $\succ$-*critical path* from x to y.

Definition 7.2.5 *Given $G \subseteq \mathcal{L}_\succ$ and $b \in \mathbb{Z}^n$, a path (x,z,y) in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ is a* $\succ$-**critical path** *if $z \succ x$ and $z \succ y$.*

If (x,z,y) is a $\succ$-critical path in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$, then $x + u = z = y + v$ for some pair $u, v \in G$, in which case, we call (x,z,y) a $\succ$-critical path for (u,v) (see Figure 7.3).

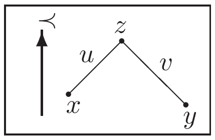


Figure 7.3: A critical path for (u,v) between x, z, and y.

The following lemma will be a crucial ingredient in the correctness proofs of the algorithms presented in Section 7.3. It will guarantee correctness of the algorithm under consideration, since the necessary reduction paths have been constructed during the run of the algorithm. In the next lemma, we cannot assume that G is a generating set of $\mathcal{L}$, since often this is what we are trying to construct.

Lemma 7.2.6 *Let $x, y \in \mathcal{F}_{\mathcal{L},b}$ for some $b \in \mathbb{Z}^n$, and let $G \subseteq \mathcal{L}_\succ$ where there is a path between x and y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$. If there exists a $\succ$-reduction path between x' and y' for every $\succ$-critical path (x', z', y') in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$, then there exists a $\succ$-reduction path between x and y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$.*

Proof. Assume on the contrary that no such $\succ$-reduction path exists from x to y. Among all paths $(x = x^0, \ldots, x^k = y)$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ choose one such that $\max_\succ\{x^0, \ldots, x^k\}$ is minimal. Such a minimal path exists since $\succ$ is a term ordering. Let $j \in \{0, \ldots, k\}$ where x^j attains this maximum. By assumption, $(x^0, \ldots, x^k)$ is not a $\succ$-reduction path, and thus, $x^j \succ x^0$ and $x^j \succ x^k$, and since x^j is maximal, we have $x^j \succ x^{j-1}$ and $x^j \succ x^{j+1}$. Let $u = x^j - x^{j-1}$ and $v = x^j - x^{j+1}$. Then (x^{j-1}, x^j, x^{j+1}) forms a $\succ$-critical path. Consequently, we can replace the path (x^{j-1}, x^j, x^{j+1}) with the $\succ$-reduction path $(x^{j-1} = \overline{x}^0, \ldots, \overline{x}^s = x^{j+1})$ in the path $(x^0, \ldots, x^k)$ and obtain a new path between x and y with the property that the $\succ$-maximum of the intermediate nodes is strictly less than $x^j = \max_\succ\{x^1, \ldots, x^{k-1}\}$ (see Figure 7.4). This contradiction proves our claim. $\square$

The following corollary is a straight-forward consequence of Lemma 7.2.6, but nonetheless, it is worthwhile stating explicitly.

Corollary 7.2.7 *Let $G \subseteq \mathcal{L}_\succ$. If for all $b' \in \mathbb{Z}^n$ and for every $\succ$-critical path (x', z', y') in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b'}, G)$, there exists a $\succ$-reduction path between x' and y', then for all $b \in \mathbb{Z}^n$ and for all $x, y \in \mathcal{F}_{\mathcal{L},b}$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$, there exists a $\succ$-reduction path between x and y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$.*

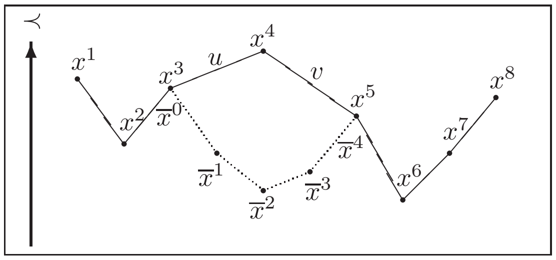


Figure 7.4: Replacing a critical path by a reduction path

Combining Corollary 7.2.7 with Lemma 7.2.4, we arrive at the following result for Gröbner bases.

Corollary 7.2.8 *A set $G \subseteq \mathcal{L}_\succ$ is a $\succ$-Gröbner basis of $\mathcal{L}$ if and only if G is a generating set of $\mathcal{L}$ and if for all $b \in \mathbb{Z}^n$ and for every $\succ$-critical path (x, z, y) in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$, there exists a $\succ$-reduction path between x and y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$.*

In Corollary 7.2.7 and Corollary 7.2.8, it is not necessary to check for a $\succ$-reduction path from x to y for every $\succ$-critical path (x, y, z) in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ for all $b \in \mathbb{Z}^n$. Consider the case where there exists another $\succ$-critical path (x', y', z') in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b'}, G)$ for some $b' \in \mathbb{Z}^n$ such that $(x, y, z) = (x' + \gamma, y' + \gamma, z' + \gamma)$ for some $\gamma \in \mathbb{Z}^n_+$. Then, a $\succ$-reduction path from x' to y' in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b'}, G)$ translates by γ to a $\succ$-reduction path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$. Thus, we only need to check for a $\succ$-reduction path from x' to y'.

A $\succ$-critical path (x, y, z) is *minimal* if there does not exist another $\succ$-critical path (x', y', z') such that $(x, y, z) = (x' + \gamma, y' + \gamma, z' + \gamma)$ for some $\gamma \in \mathbb{Z}^n_+$ where $\gamma \neq 0$, or equivalently, $\min\{x_i, y_i, z_i\} = 0$ for all $i = 1, \ldots, n$. Consequently, if there exists a $\succ$-reduction path between x and y for all minimal $\succ$-critical paths (x, y, z), then there exists a $\succ$-reduction path between x' and y' for all $\succ$-critical paths (x', y', z'). Also, for each pair of vectors $u, v \in \mathcal{L}$, there exists a unique minimal $\succ$-critical path $(x^{(u,v)}, z^{(u,v)}, y^{(u,v)})$ determined by $z^{(u,v)} := \max\{u^+, v^+\}$ component-wise, $x^{(u,v)} := z^{(u,v)} - u$ and $y^{(u,v)} := z^{(u,v)} - v$. So, any other $\succ$-critical path for (u, v) is of the form $(x^{(u,v)} + \gamma, z^{(u,v)} + \gamma, y^{(u,v)} + \gamma)$ for some $\gamma \in \mathbb{Z}^n_+$. Using minimal $\succ$-critical paths, we can rewrite Corollary 7.2.7 and Corollary 7.2.8, so that we only need to check for a *finite* number of $\succ$-reduction paths.

Lemma 7.2.9 *Let $G \subseteq \mathcal{L}_\succ$. If there exists a $\succ$-reduction path between $x^{(u,v)}$ and $y^{(u,v)}$ for every pair $u, v \in G$, then for all $b \in \mathbb{Z}^n$ and for all $x, y \in \mathcal{F}_{\mathcal{L},b}$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$, there exists a $\succ$-reduction path between x and y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$*

Corollary 7.2.10 *A set $G \subseteq \mathcal{L}_\succ$ is a $\succ$-Gröbner basis of $\mathcal{L}$ if and only if G is a generating set of $\mathcal{L}$ and for each pair $u, v \in G$, there exists a $\succ$-reduction path between $x^{(u,v)}$ and $y^{(u,v)}$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},z^{(u,v)}}, G)$.*

We now turn Lemma 7.2.9 into an algorithmic tool. The following algorithm, Algorithm 7.2.12 below, called a completion procedure [27], guarantees that if for a set $S \subseteq \mathcal{L}$ the points x and y

are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},x}, S)$, then there exists a $\succ$-reduction path between x and y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},x}, G)$, where G denotes the set returned by the completion procedure. Thus, if S is a generating set of $\mathcal{L}$, then Algorithm 7.2.12 returns a set G that is a $\succ$-Gröbner basis of $\mathcal{L}$ by Corollary 7.2.10.

Given a set $S \subseteq \mathcal{L}$, the completion procedure first sets $G := S$ and then directs all vectors in G according to $\succ$ such that $G \subseteq \mathcal{L}_\succ$. Note that at this point $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S) = \mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ for all $b \in \mathbb{Z}^n$. The completion procedure then determines whether the set G satisfies Lemma 7.2.9; in other words, it tries to find a reduction path from $x^{(u,v)}$ to $y^{(u,v)}$ for every pair $u, v \in G$. If G satisfies Lemma 7.2.9, then we are done. Otherwise, no $\succ$-reduction path was found for some (u, v), in which case, we add a vector to G so that a $\succ$-reduction path exists, and then again, test whether G satisfies Lemma 7.2.9, and so on.

To check for a $\succ$-reduction path, using the “Normal Form Algorithm”, Algorithm 7.2.11 below, we construct a maximal $\succ$-decreasing path in $\mathcal{G}(\mathcal{F}_{\mathcal{L},z^{(u,v)}}, G)$ from $x^{(u,v)}$ to some x', and a maximal $\succ$-decreasing path in $\mathcal{G}(\mathcal{F}_{\mathcal{L},z^{(u,v)}}, G)$ from $y^{(u,v)}$ to some y'. If $x' = y'$, then we have found a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$. Otherwise, we add the vector $r \in \mathcal{L}_\succ$ to G where $r := x' - y'$ if $x' \succ y'$, and $r := y' - x'$ otherwise, so therefore, there is now a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},z^{(u,v)}}, G)$. Note that before we add r to G, since the paths from x to x' and from y to y' are maximal, there does not exist $u \in G$ such that $x' \geq u^+$ or $y' \geq u^+$. Therefore, there does not exist $u \in G$ such that $r^+ \geq u^+$. This condition is needed to ensure that the completion procedure terminates.

Algorithm 7.2.11 (Normal Form Algorithm)

Input: a vector $x \in \mathbb{Z}^n_+$ and a set $G \subseteq \mathcal{L}_\succ$.

Output: a vector x' where there is a maximal $\succ$-decreasing path from x to x' in $\mathcal{G}(\mathcal{F}_{\mathcal{L},x}, G)$.

```
$x' := x$
while there is some $u \in G$ such that $u^+ \leq x'$ do
        $x' := x' - u$
return $x'$
```

We write normalForm(x, G) for the output of the Normal Form Algorithm.

Algorithm 7.2.12 (Completion procedure)

Input: a term ordering $\succ$ and a set $S \subseteq \mathcal{L}$.

Output: a set $G \subseteq \mathcal{L}_\succ$ such that if x, y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},x}, S)$, then there exists a $\succ$-reduction path between x and y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},x}, G)$.

```
$G := \{u : u^+ \succ u^-, u \in S\} \cup \{-u : u^- \succ u^+, u \in S\}$
$C := \{(u, v) : u, v \in G\}$
while $C \neq \emptyset$ do
```

```
        Select $(u, v) \in C$
        $C := C \setminus \{(u, v)\}$
        $r :=$ normalForm$(x^{(u,v)}, G)$ − normalForm$(y^{(u,v)}, G)$
        if $r \neq 0$ then
            if $r^- \succ r^+$ then $r := -r$
            $C := C \cup \{(r, s) : s \in G\}$
            $G := G \cup \{r\}$
return $G$.
```

We write $\mathcal{CP}(\succ, S)$ for the output of the Completion Procedure.

Lemma 7.2.13 *Algorithm 7.2.12 terminates and satisfies its specifications.*

Proof. Let $(r^1, r^2, \dots)$ be the sequence of vectors r that are added to the set G during the Algorithm 7.2.12. Since before we add r to G, there does not exist $u \in G$ such that $r^+ \geq u^+$, the sequence satisfies ${r^i}^+ \not\leq {r^j}^+$ whenever $i < j$. By the Gordan-Dickson Lemma, Lemma 1.1.1, such a sequence must be finite and thus, Algorithm 7.2.12 must terminate.

When the algorithm terminates, the set G must satisfy the property that for each $u, v \in G$, there exists a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$, and therefore, by Lemma 7.2.9, there exists a $\succ$-reduction path between x and y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ for all $x, y \in \mathcal{F}_{\mathcal{L},b}$ for all $b \in \mathbb{Z}^n$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$. Moreover, by construction, $S \subseteq G \cup -G$, and therefore, if x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$, then x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$. □

Note that the completion procedure preserves connectivity: given $x, y \in \mathcal{F}_{\mathcal{L},b}$ for some $b \in \mathbb{Z}^n$, if x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$, then x and y are also connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$.

7.3 Computing a generating set

In this section, we finally present three algorithms to compute a generating set of $\mathcal{L}$:

- the "Saturation" algorithm (Hoşten and Sturmfels [78]),
- the "Lift-and-Project" algorithm (Bigatti, LaScala, and Robbiano [21]), and
- our "Project-and-Lift" algorithm (Hemmecke and Malkin [75]).

Each algorithm produces a generating set of $\mathcal{L}$ that is not necessarily minimal, and so, once a generating set of $\mathcal{L}$ is known, a minimal generating set of $\mathcal{L}$ can be computed by a single Gröbner basis computation (see [29] for more details). The fundamental idea behind all three algorithms is essentially the same, and the main algorithmic building block of the algorithms is the completion procedure.

7.3.1 The "Saturation" algorithm

Let $x, y \in \mathcal{F}_{\mathcal{L},b}$ for some $b \in \mathbb{Z}^n$, and let $S \subseteq \mathcal{L}$. Observe that if x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$, then $x + \gamma$ and $y + \gamma$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$ for any $\gamma \in \mathbb{Z}^n_+$ since we can just translate any path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$ by γ giving a path from $x + \gamma$ to $y + \gamma$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b+\gamma}, S)$. However, it is not necessarily true that $x + \gamma$ and $y + \gamma$ are also connected for any $\gamma \in \mathbb{Z}^n$ (γ may be negative) where $x + \gamma \geq 0$ and $y + \gamma \geq 0$.

Given a set $S \subseteq \mathcal{L}$, the Saturation algorithm constructs a set T such that if x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$ for some $b \in \mathbb{Z}^n$, then $x + \gamma$ and $y + \gamma$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, T)$ for any $\gamma \in \mathbb{Z}^n$ where $x + \gamma \geq 0$ and $y + \gamma \geq 0$. Importantly then, if S spans $\mathcal{L}$, then T must be a generating set of $\mathcal{L}$. This follows since if S spans $\mathcal{L}$, then for all $b \in \mathbb{Z}^n$ and for all $x, y \in \mathcal{F}_{\mathcal{L},b}$, there must exist a $\gamma \in \mathbb{Z}^n_+$ such that $x + \gamma$ and $y + \gamma$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, T)$, and hence, x and y must also be connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, T)$ from our assumption about T.

For convenience, we need some new notation. Given $x, y \in \mathbb{Z}^n_+$, we define $x \wedge y$ as the component-wise minimum of x and y – that is, $(x \wedge y)_i = \min\{x_i, y_i\}$ for all $i = 1, \ldots, n$. Also, given $\sigma \subseteq \{1, \ldots, n\}$, we define $x \wedge_\sigma y$ as the component-wise minimum of x and y for the σ components and 0 otherwise – that is, $(x \wedge y)_i = \min\{x_i, y_i\}$ if $i \in \sigma$ and $(x \wedge y)_i = 0$ otherwise.

Definition 7.3.1 *Let $\sigma \subseteq \{1, \ldots, n\}$, and let $S, T \subseteq \mathcal{L}$. The set T is **σ-saturated on** S if and only if for all $b \in \mathbb{Z}^n$ and for all $x, y \in \mathcal{F}_{\mathcal{L},b}$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$, the points $x - \gamma$ and $y - \gamma$ are also connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b-\gamma}, T)$ where $\gamma = x \wedge_\sigma y$.*

So, when T is σ-saturated on S, if x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$, then $x + \gamma$ and $y + \gamma$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, T)$ for any $\gamma \in \mathbb{Z}^n$ (γ can be negative) where $x + \gamma \geq 0$, $y + \gamma \geq 0$, and $\mathrm{supp}(\gamma) \subseteq \sigma$. Saturation is thus concerned with the connectivity of a set T in relation to the connectivity of another set S. Note that, by definition, a set $S \subseteq \mathcal{L}$ is $\emptyset$-saturated on itself. Also observe that if S spans $\mathcal{L}$, then $T \subseteq \mathcal{L}$ is $\{1, \ldots, n\}$-saturated on S if and only if T is a generating set of $\mathcal{L}$.

The fundamental idea behind the Saturation algorithm is given $S, T \subseteq \mathcal{L}$ where T is σ-saturated on S for some $\sigma \subseteq \{1, \ldots, n\}$, we can compute a set T' that is a $(\sigma \cup \{i\})$-saturated on S for any $i \in \overline{\sigma}$. Therefore, given a set $S \subseteq \mathcal{L}$ that spans $\mathcal{L}$, starting from a set $T = S$, which is $\emptyset$-saturated on S, if we do this repeatedly for each $i \in \{1, \ldots, n\}$, we arrive at a set $T' \subseteq \mathcal{L}$ that is $\{1, \ldots, n\}$-saturated on S and, therefore, a generating set of $\mathcal{L}$.

The following two lemmas are fundamental to the saturation algorithm. First, we extend the definition of reduction paths. Given $\varphi \in \mathbb{Q}^n$, a path $(x^0, \ldots, x^k)$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ is an *φ-reduction path* if for *no* $j \in \{1, \ldots, k-1\}$, we have $\varphi^\intercal x^j > \varphi^\intercal x^0$ and $\varphi^\intercal x^j > \varphi^\intercal x^k$. Also, we define e^i to be the ith unit vector and $\overline{e}^i = -e^i$. So, given $b \in \mathbb{Z}^n$, the path $(x^0, \ldots, x^k) \subseteq \mathcal{F}_{\mathcal{L},b}$ is a $\overline{e}^i$-reduction path if $x^j_i \geq x^0_i$ or $x^j_i \geq x^k_i$ for $j = 1, \ldots, k-1$.

Lemma 7.3.2 *Let $S, T \subseteq \mathcal{L}$ and $i \in \{1, ..., n\}$. The set T is $\{i\}$-saturated on S if and only if for all $b \in \mathbb{Z}^n$ and for all $x, y \in \mathcal{F}_{\mathcal{L},b}$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$, there exists a $\overline{e}^i$-reduction path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, T)$.*

Proof. Let $x, y \in \mathcal{F}_{\mathcal{L},b}$ for some $b \in \mathbb{Z}^n$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$ and let $\gamma = x \wedge_{\{i\}} y$.

Assume that T is $\{i\}$-saturated on S, and so, $x - \gamma$ and $y - \gamma$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b-\gamma}, T)$. Let $(x - \gamma = x^0, \ldots, x^k = y - \gamma)$ be a path from $x - \gamma$ and $y - \gamma$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b-\gamma}, T)$. The path $(x = x^0 + \gamma, \ldots, x^k + \gamma = y)$ is a $\overline{e}^i$-reduction path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, T)$.

Conversely, by assumption, there exists a $\overline{e}^i$-reduction path $(x = x^0, \ldots, x^k = y)$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, T)$. The path $(x - \gamma = x^0 - \gamma, \ldots, x^k - \gamma = y - \gamma)$ is thus a feasible path from $x - \gamma$ to $y - \gamma$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b-\gamma}, T)$. □

Given any vector $\varphi \in \mathbb{Q}^n$ and a term order $\prec$, we define the order $\prec_\varphi$ where $x \prec_\varphi y$ if $\varphi^\intercal x < \varphi^\intercal y$ or $\varphi^\intercal x = \varphi^\intercal y$ and $x \prec y$. Since we assume $\mathcal{L} \cap \mathbb{Z}^n_+ = \{0\}$, the order $\prec_{\overline{e}^i}$ is thus a term ordering for $\mathcal{L}$. Importantly then, a $\prec_{\overline{e}^i}$-reduction path is also an $\overline{e}^i$-reduction path. Let $T = \mathcal{CP}(\prec_{\overline{e}^i}, S)$. Then, by the properties of the completion procedure, for all $b \in \mathbb{Z}^n$ and $x, y \in \mathcal{F}_{\mathcal{L},b}$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$, there exists a $\prec_{\overline{e}^i}$-reduction path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, T)$; T is therefore $\{i\}$-saturated on S.

Let $S, T \subseteq \mathcal{L}$ and T is σ-saturated on S for some $\sigma \subseteq \{1, \ldots, n\}$. Let $T' = \mathcal{CP}(\prec_{\overline{e}^i}, T)$. So, T' is therefore $\{i\}$-saturated on T. For the saturation algorithm to work, we need that T' is also $(\sigma \cup \{i\})$-saturated on S, which follows from Lemma 7.3.3 below.

Lemma 7.3.3 *Let $\sigma, \tau \subseteq \{1, \ldots, n\}$ and $S, T, U \subseteq \mathcal{L}$. If U is σ-saturated on S, and T is τ-saturated on U, then T is $(\sigma \cup \tau)$-saturated on S.*

Proof. Let $b \in \mathbb{Z}^n$, and $x, y \in \mathcal{F}_{\mathcal{L},b}$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$. Let $\alpha = x \wedge_\sigma y$. Since T is σ-saturated on S, $x - \alpha$ and $y - \alpha$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b-\alpha}, T)$. Let $\beta = x - \alpha \wedge_\tau y - \alpha$. Then, since U is τ-saturated on T, $x - \alpha - \beta$ and $y - \alpha - \beta$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b-\alpha-\beta}, U)$. Let $\gamma = \alpha + \beta$; then, $\gamma = x \wedge_{(\sigma\cup\tau)} y$. Therefore, there is a path from $x - \gamma$ to $y - \gamma$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b-\gamma}, U)$ as required. □

We now arrive at the Saturation algorithm below.

Algorithm 7.3.4 (Saturation algorithm)

Input: a spanning set S of $\mathcal{L}$.

Output: a generating set G of $\mathcal{L}$.

```
$G := S$
$\sigma := \emptyset$
while $\sigma \neq \{1, ..., n\}$ do
    Select $i \in \overline{\sigma}$
    $G := \mathcal{CP}(\prec_{\overline{e}^i}, G)$
    $\sigma := \sigma \cup \{i\}$
return $G$.
```

Lemma 7.3.5 *Algorithm 7.3.4 terminates and satisfies its specifications.*

Proof. Algorithm 7.3.4 terminates, since Algorithm 7.2.12 always terminates. We show at the beginning of each iteration that G is σ-saturated on S, and so, at the end of the algorithm G is $\{1, \ldots, n\}$-saturated on S; therefore, G is a generating set of $\mathcal{L}$. At the beginning of the first iteration, G is σ-saturated on S since $\sigma = \emptyset$ and $G = S$. So, we can assume it is true for the current iteration, and now, we show it is true for the next iteration. Let $G' := \mathcal{CP}(\prec_{\overline{e}^i}, G)$. Then, by Lemma 7.3.2, G' is $\{i\}$-saturated on G, and so, by Lemma 7.3.3, G' is $(\sigma \cup \{i\})$-saturated on S. So, G is σ-saturated on S at the beginning of the next iteration. □

During the Saturation algorithm, we saturate n times, once for each $i \in \{1, \ldots, n\}$. However, as proven in [77], it is in fact only necessary to perform at most $\lfloor \frac{n}{2} \rfloor$ saturations. Given $S, T \subseteq \mathcal{L}$, we can show that there always exists a $\sigma \subseteq \{1, \ldots, n\}$ where $|\sigma| \leq \lfloor \frac{n}{2} \rfloor$ such that if T is σ-saturated on S, then T is $\{1, \ldots, n\}$-saturated on S. The following two lemmas prove the result.

Lemma 7.3.6 *Let $\sigma \subseteq \{1, \ldots, n\}$, $S, T \subseteq \mathcal{L}$ where T is σ-saturated on S, and $u \in S$. If $\mathrm{supp}(u^-) \subseteq \sigma$ or $\mathrm{supp}(u^+) \subseteq \sigma$, then T is $(\mathrm{supp}(u) \cup \sigma)$-saturated on S.*

Proof. Assume that $\mathrm{supp}(u^-) \subseteq \sigma$. Let $x, y \in \mathcal{F}_{\mathcal{L},b}$ for some $b \in \mathbb{Z}^n$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$. Let $\alpha = x \wedge_{\mathrm{supp}(u^+)} y$ and $\beta = x - \alpha \wedge_\sigma y - \alpha$. We must show that $x - \alpha - \beta$ and $y - \alpha - \beta$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b-\alpha-\beta}, T)$ since $\alpha + \beta = x \wedge_{(\mathrm{supp}(u^+) \cup \sigma)} y$. By translating the path from x to y by α, we get a path from $x - \alpha$ to $y - \alpha$ that is non-negative on all components except $\mathrm{supp}(u^+)$. This path can transformed into a path that is non-negative on all components except $\mathrm{supp}(u^-)$ by adding u to the start of the path as many times as necessary and subtracting u from the end of the path the same number of times. Therefore, $x - \alpha + \gamma$ and $y - \alpha + \gamma$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b-\alpha+\gamma}, S)$ for some $\gamma \in \mathbb{Z}^n_+$ where $\mathrm{supp}(\gamma) \subseteq \mathrm{supp}(u^-) \subseteq \sigma$. Observe that $\mathrm{supp}(\beta + \gamma) \subseteq \sigma$. Thus, since T is σ-saturated, $x - \alpha - \beta$ and $y - \alpha - \beta$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b-\alpha-\beta}, T)$ as required.

The case where T is $\mathrm{supp}(u^+) \subseteq \sigma$ is essentially the same as above. □

Lemma 7.3.7 *Let $S, T \subseteq \mathcal{L}$. There exists a $\sigma \subseteq \{1, \ldots, n\}$ where $|\sigma| \leq \lfloor \frac{n}{2} \rfloor$ such that if T is σ-saturated on S, then T is $\{1, \ldots, n\}$-saturated on S.*

Proof. We show this by construction. Without loss of generality, we assume that $\mathcal{L}$ is not contained in any of the linear subspaces $\{x_i : x_i = 0, x \in \mathbb{R}^n\}$ for $i = 1, \ldots, n$; otherwise, we may simply delete this component.

Let $\sigma = \emptyset$, $\tau = \emptyset$, and $U = \emptyset$. Repeat the following steps until $\tau = \{1, \ldots, n\}$.

1. Select $u \in S$ such that $\mathrm{supp}(u) \setminus \tau \neq \emptyset$.
2. If $|\mathrm{supp}(u^+) \setminus \tau| \geq |\mathrm{supp}(u^-) \setminus \tau|$, then $\sigma := \sigma \cup \mathrm{supp}(u^-)$, else $\sigma := \sigma \cup \mathrm{supp}(u^+)$.
3. Set $\tau := \tau \cup \mathrm{supp}(u)$, and set $U := U \cup \{u\}$.

The procedure must terminate since during each iteration we increase the size of τ. Note that, at termination, $U \subseteq S$, $\bigcup_{u\in U} \operatorname{supp}(u) = \tau = \{1,\ldots,n\}$, and for all $u \in U$ either $\operatorname{supp}(u^+) \subseteq \sigma$ or $\operatorname{supp}(u^-) \subseteq \sigma$. Therefore, by applying Lemma 7.3.6 recursively for each $u \in U$, we have that if T is σ-saturated on S, then T is $\{1,\ldots,n\}$-saturated on S. Lastly, since in each iteration we add at least twice as many components to τ as to σ, we conclude that at termination $|\sigma| \le \lfloor \frac{n}{2} \rfloor$. $\square$

Example 7.3.8 Consider again the set $S := \{(1,-1,-1,-3,-1,2),(1,0,2,-2,-2,1)\}$. Let $\mathcal{L}$ be the lattice spanned by S, and let $\sigma = \{1,6\}$. Then, since $\operatorname{supp}((1,-1,-1,-3,-1,2)^+) = \{1,6\}$ and $\operatorname{supp}((1,-1,-1,-3,-1,2)) = \{1,2,3,4,5,6\}$, we conclude by Lemma 7.3.6 that if a set $T \subseteq \mathcal{L}$ is $\{1,6\}$-saturated on S, then T is $\{1,2,3,4,5,6\}$-saturated on S. So, to compute a generating set of $\mathcal{L}$, we only need to saturate on $\{1,6\}$. The following table gives the values of σ, i, and G at each stage of the Saturation algorithm when constructing a set that is $\{1,6\}$-saturated on S, and hence, a generating set of $\mathcal{L}$.

σ	i	$G := \mathcal{CP}(\prec_{\overline{e}^i}, G)$
$\emptyset$	1	$\{(-1,0,-2,2,2,-1),(-1,1,1,3,1,-2),(-1,2,4,4,0,-3),(0,-1,-3,-1,1,1)\}$
$\{1\}$	6	$\{(0,1,3,1,-1,-1),(-1,1,1,3,1,-2),(-1,0,-2,2,2,-1),(-1,-1,-5,1,3,0),(1,2,8,0,-4,-1)\}$

Observe that after the first iteration, that G is not a generating set of $\mathcal{L}$. The set G does not contain the vector $(-1,-1,-5,1,3,0)$, and so, the graph $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ where $b = (0,0,0,1,3,0)$ is disconnected. Note that the final set G is not a *minimal* generating set of $\mathcal{L}$; the vector $(1,2,8,0,-4,-1)$ is not needed. See [29] for an algorithm to compute a minimal generating set. $\square$

We now introduce the concept of a σ-generating set of $\mathcal{L}$ for some $\sigma \subseteq \{1,\ldots,n\}$ – a generalization of a generating set of $\mathcal{L}$. These new generating sets provide useful insights into saturation and the inspiration for the Project-and-Lift algorithm as well as a point of reference to compare the two algorithms.

Firstly, we define $\mathcal{F}^\sigma_{\mathcal{L},b} := \{z : z \equiv b \pmod{\mathcal{L}}, z_{\overline{\sigma}} \ge 0, z \in \mathbb{Z}^n\}$ where $\sigma \subseteq \{1,\ldots,n\}$, so we now allow the σ components to be negative. Given $S \subseteq \mathcal{L}$, analogous to $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$, we define $\mathcal{G}(\mathcal{F}^\sigma_{\mathcal{L},b}, S)$ to be the undirected graph with nodes $\mathcal{F}^\sigma_{\mathcal{L},b}$ and edges (x,y) if $x-y \in S$ or $y-x \in S$. Observe that a path in $\mathcal{G}(\mathcal{F}^\sigma_{\mathcal{L},b}, S)$ is non-negative on the $\overline{\sigma}$ components and may be negative on the σ components. Analogous to a generating set of $\mathcal{L}$, a set $S \subseteq \mathcal{L}$ is a *σ-generating set of $\mathcal{L}$* if the graph $\mathcal{G}(\mathcal{F}^\sigma_{\mathcal{L},b}, S)$ is connected for every $b \in \mathbb{Z}^n$. Note that $\emptyset$-generating sets are equivalent to generating sets and $\{1,\ldots,n\}$-generating sets are equivalent to spanning sets.

Lemma 7.3.9 *Let $\sigma \subseteq \{1,\ldots,n\}$ and $S,T \subseteq \mathcal{L}$ where S spans $\mathcal{L}$. If T is σ-saturated on S, then T is a $\overline{\sigma}$-generating set of $\mathcal{L}$.*

Proof. Let $x,y \in \mathcal{F}^{\overline{\sigma}}_{\mathcal{L},b}$ for some $b \in \mathbb{Z}^n$. We must show that x and y are connected in $\mathcal{G}(\mathcal{F}^{\overline{\sigma}}_{\mathcal{L},b}, S)$. Since S spans $\mathcal{L}$, there must exist a $\gamma \in \mathbb{Z}^n_+$ such that $x+\gamma$ and $y+\gamma$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b+\gamma}, S)$. Let $\alpha,\beta \in \mathbb{Z}^n_+$ where $\alpha+\beta=\gamma$, $\operatorname{supp}(\alpha) \subseteq \sigma$, and $\operatorname{supp}(\beta) \subseteq \overline{\sigma}$. Since T is σ-saturated on S and $\operatorname{supp}(\alpha) \subseteq \sigma$, the points $x+\beta = x+\gamma-\alpha$ and $y+\beta = y+\gamma-\alpha$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b+\beta}, S)$. Therefore, since $\operatorname{supp}(\beta) \subseteq \overline{\sigma}$, x and y are connected in $\mathcal{G}(\mathcal{F}^{\overline{\sigma}}_{\mathcal{L},b}, S)$. $\square$

Interestingly, the converse of Lemma 7.3.9 is not true in general: a $\overline{\sigma}$-generating set is not necessarily a σ-saturated set. Let $S, T \subseteq \mathcal{L}$ where S spans $\mathcal{L}$, $\sigma \subseteq \{1, \ldots, n\}$, and let $x, y \in \mathcal{F}_{\mathcal{L},b}$ for some $b \in \mathbb{Z}^n$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$. If T is a $\overline{\sigma}$-generating set of $\mathcal{L}$, then $x - \gamma$ and $y - \gamma$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L}^{\overline{\sigma}},b-\gamma}, T)$ where $\gamma = x \wedge_\sigma y$. In other words, there is a path from $x - \gamma$ and $y - \gamma$ that remains non-negative on the σ components but may be negative on the $\overline{\sigma}$ components. On the other hand, if T is σ-saturated on S, then $x - \gamma$ and $y - \gamma$ are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b-\gamma}, T)$ where again $\gamma = x \wedge_\sigma y$. In other words, there is a path from $x - \gamma$ and $y - \gamma$ that remains non-negative on *all* the components. So, while $\overline{\sigma}$-generating sets, like σ-saturated sets, ensure path non-negativity on the σ components, they do not preserve existing path non-negativity on the other $\overline{\sigma}$ components like σ-saturated sets do. Indeed, $\overline{\sigma}$-generating sets say nothing at all about the path non-negativity of the $\overline{\sigma}$ components. So, σ-saturation is a stronger concept than $\overline{\sigma}$-generation.

In the Project-and-Lift algorithm, we compute $\overline{\sigma}$-generating sets instead of σ-saturated sets. By doing so, we can effectively *ignore* the $\overline{\sigma}$ components, and therefore, we compute smaller intermediate sets, although we start and finish at the same point.

7.3.2 The "Project-and-Lift" algorithm

Given $\sigma \subseteq \{1, \ldots, n\}$, we define the projective map $\pi_\sigma : \mathbb{Z}^n \mapsto \mathbb{Z}^{|\overline{\sigma}|}$ that projects a vector in $\mathbb{Z}^n$ onto the $\overline{\sigma} = \{1, \ldots, n\} \setminus \sigma$ components. For convenience, we write $\mathcal{L}^\sigma$ where $\sigma \subseteq \{1, \ldots, n\}$ as the projection of $\mathcal{L}$ onto the $\overline{\sigma}$ components – that is, $\mathcal{L}^\sigma = \pi_\sigma(\mathcal{L})$. Note that $\mathcal{L}^\sigma$ is also a lattice.

The fundamental idea behind the Project-and-Lift algorithm is that using a set $S \subseteq \mathcal{L}^{\{i\}}$ that is a generating set of $\mathcal{L}^{\{i\}}$ for some $i \in \{1, \ldots, n\}$, we can compute a set $S' \subseteq \mathcal{L}^{\{i\}}$ such that S' lifts to a generating set of $\mathcal{L}$. So, for some $\sigma \subseteq \{1, \ldots, n\}$, since $\mathcal{L}^\sigma$ is also a lattice, starting with a generating set of $\mathcal{L}^\sigma$, we can compute a generating of $\mathcal{L}^{\sigma \setminus \{i\}}$ for some $i \in \sigma$. So, by doing this repeatedly for every $i \in \sigma$, we attain a generating set of $\mathcal{L}$.

First, we extend the definition of Gröbner bases. Given $\varphi \in \mathbb{Q}^n$, recall that a path $(x^0, \ldots, x^k)$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ is a φ-reduction path if for *no* $j \in \{1, \ldots, k-1\}$, we have $\varphi^\intercal x^j > \varphi^\intercal x^0$ and $\varphi^\intercal x^j > \varphi^\intercal x^k$. A set $G \subseteq \mathcal{L}$ is a *φ-Gröbner basis* of $\mathcal{L}$ if for all $b \in \mathbb{Z}^n$ and for every pair $x, y \in \mathcal{F}_{\mathcal{L},b}$, there exists a φ-reduction path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$.

The following lemma is fundamental to the Project-and-Lift algorithm. Note that the property that $\ker(\pi_{\{i\}}) \cap \mathcal{L} = \{0\}$ for some $i \in \{1, \ldots, n\}$ means that each vector in $\mathcal{L}^{\{i\}}$ lifts to a unique vector in $\mathcal{L}$, and thus, the inverse map $\pi^{-1}_{\{i\}} : \mathcal{L}^{\{i\}} \mapsto \mathcal{L}$ is well-defined. Moreover, by linear algebra, there must exist a vector $\omega^i \in \mathbb{Q}^{n-1}$ such that for all $u \in \mathcal{L}^{\{i\}}$, we have ${\omega^i}^\intercal u = (\pi^{-1}_{\{i\}}(u))_i$. We will always write such a vector as ω^i, and also, we define $\overline{\omega}^i = -\omega^i$.

Lemma 7.3.10 *Let $i \in \{1, \ldots, n\}$ where $\ker(\pi_{\{i\}}) \cap \mathcal{L} = \{0\}$, and let $S \subseteq \mathcal{L}^{\{i\}}$. The set S is a $\overline{\omega}^i$-Gröbner basis of $\mathcal{L}^{\{i\}}$ if and only if $\pi^{-1}_{\{i\}}(S)$ is a $\overline{e}^i$-Gröbner basis of $\mathcal{L}$.*

Proof. Assume S is a $\overline{\omega}^i$-Gröbner basis of $\mathcal{L}^{\{i\}}$. Let $x, y \in \mathcal{F}_{\mathcal{L},b}$ for some $b \in \mathbb{Z}^n$. We need to show that there is a $\overline{e}^i$-reduction path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, \pi^{-1}_{\{i\}}(S))$. Let $\overline{x} = \pi_{\{i\}}(x)$, $\overline{y} = \pi_{\{i\}}(y)$, and

$\overline{b} = \pi_{\{i\}}(b)$. By assumption, there exists a $\overline{\omega}^i$-reduction path $(\overline{x} = \overline{x}^0, \ldots, \overline{x}^k = \overline{y})$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L}^{\{i\}},\overline{b}}, S)$. So, we have $\omega^{i\mathsf{T}}\overline{x}^j \geq \omega^{i\mathsf{T}}\overline{x}$ or $\omega^{i\mathsf{T}}\overline{x}^j \geq \omega^{i\mathsf{T}}\overline{y}$ for all j. We now lift this $\overline{\omega}^i$-reduction path in $\mathcal{G}(\mathcal{F}_{\mathcal{L}^{\{i\}},\overline{b}}, S)$ to a $\overline{e}^i$-reduction path in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, \pi^{-1}_{\{i\}}(S))$.

Let $x^j = x + \pi^{-1}_{\{i\}}(\overline{x}^j - \overline{x}) = y + \pi^{-1}_{\{i\}}(\overline{x}^j - \overline{y})$ for all $j = 0, \ldots, k$. Hence, $\pi_{\{i\}}(x^j) = \overline{x}^j$ and $x^j_i = x_i + \omega^{i\mathsf{T}}\overline{x}^j - \omega^{i\mathsf{T}}\overline{x} = y_i + \omega^{i\mathsf{T}}\overline{x}^j - \omega^{i\mathsf{T}}\overline{y}$, and so, $x^j_i \geq x_i$ or $x^j_i \geq y_i$. Also, $x^j - x^{j-1} = \pi^{-1}_{\{i\}}(\overline{x}^j - \overline{x}^{j-1}) \in \pi^{-1}_{\{i\}}(S)$ for all $j = 1, \ldots, k$. Therefore, $(x = x^0, \ldots, y^k = y)$ is an $\overline{e}^i$-reduction path in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, \pi^{-1}_{\{i\}}(S))$ as required.

Assume that $\pi^{-1}_{\{i\}}(S)$ is a $\overline{e}^i$-Gröbner basis of $\mathcal{L}$. Let $x, y \in \mathcal{F}_{\mathcal{L}^{\{i\}},b}$ for some $b \in \mathbb{Z}^{n-1}$, and let $\gamma = \omega^{i\mathsf{T}}(x - y)$. If $\gamma > 0$, then let $\overline{x} = (x, \gamma)$ and $\overline{y} = (y, 0)$, else let $\overline{x} = (x, 0)$ and $\overline{y} = (y, -\gamma)$; hence, $\overline{x}, \overline{y} \in \mathcal{F}_{\mathcal{L},\overline{b}}$ for some $\overline{b} \in \mathbb{Z}^n$, and $\pi_{\{i\}}(\overline{b}) = b$, $\pi_{\{i\}}(\overline{x}) = x$, $\pi_{\{i\}}(\overline{y}) = y$, and $\min\{\overline{x}_i, \overline{y}_i\} = 0$. By assumption, there exists a $\overline{e}^i$-reduction path $(\overline{x} = \overline{x}^0, \ldots, \overline{x}^k = \overline{y})$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},\overline{b}}, \pi^{-1}_{\{i\}}(S))$. Let $x^j = \pi_{\{i\}}(\overline{x}^j)$. So, $(x = x^0, \ldots, x^k = y)$ is a path in $\mathcal{G}(\mathcal{F}_{\mathcal{L}^{\{i\}},b}, S)$. Moreover, since $\overline{x}^j_i \geq \overline{x}_i$ or $\overline{x}^j_i \geq \overline{y}_i$ for all j, we have $\omega^{i\mathsf{T}}x^j \geq \omega^{i\mathsf{T}}x$ or $\omega^{i\mathsf{T}}x^j \geq \omega^{i\mathsf{T}}y$ for all j. Therefore, the path is a $\overline{\omega}^i$-reduction path. □

By definition, a $\overline{e}^i$-Gröbner basis of $\mathcal{L}$ is also a generating set of $\mathcal{L}$. On the other hand, a generating set of $\mathcal{L}$ is also a $\overline{e}^i$-Gröbner basis of $\mathcal{L}$. This follows since, given a generating set of $\mathcal{L}$, for any $x, y \in \mathcal{F}_{\mathcal{L},b}$ for any b, there must exist a path from $x - \gamma$ to $y - \gamma$ where $\gamma = x \wedge_{\{i\}} y$, and by translating such a path by γ, we get a $\overline{e}^i$-reduction path from x to y. This can also be shown using Lemma 7.3.2. So, we arrive at the following corollary.

Corollary 7.3.11 *Let $i \in \{1, \ldots, n\}$ where $\ker(\pi_{\{i\}}) \cap \mathcal{L} = \{0\}$, and let $S \subseteq \mathcal{L}^{\{i\}}$. The set S is a $\overline{\omega}^i$-Gröbner basis of $\mathcal{L}^{\{i\}}$ if and only if $\pi^{-1}_{\{i\}}(S)$ is a generating set of $\mathcal{L}$.*

Given a vector $\varphi \in \mathbb{Q}^n$, any $\prec_\varphi$-reduction path is also a φ-reduction path, and so, any $\prec_\varphi$-Gröbner basis is also a φ-Gröbner basis. So, given a set $S \subseteq \mathcal{L}^{\{i\}}$ that generates $\mathcal{L}^{\{i\}}$, we can compute a $\overline{\omega}^i$-Gröbner basis $S' \subseteq \mathcal{L}^{\{i\}}$ of $\mathcal{L}^{\{i\}}$ by running the completion procedure with respect to $\prec_{\overline{\omega}^i}$ on S. That is, $S' = \mathcal{CP}(\prec_{\overline{\omega}^i}, S)$. Hence, by Lemma 7.3.10, $\pi^{-1}_{\{i\}}(S')$ is a generating set of $\mathcal{L}$.

We can apply the above reasoning to compute a generating set of $\mathcal{L}^{\sigma\setminus\{i\}}$ from a generating set of $\mathcal{L}^\sigma$ for some $\sigma \subseteq \{1, \ldots, n\}$ and $i \in \sigma$. First, analogously to $\pi_{\{i\}}$ and $\prec_{\overline{\omega}^i}$ in the context of $\mathcal{L}^{\{i\}}$ and $\mathcal{L}$, we define $\pi^\sigma_{\{i\}}$ and $\prec^\sigma_{\overline{\omega}^i}$ in the same way except in the context of $\mathcal{L}^\sigma$ and $\mathcal{L}^{\sigma\setminus\{i\}}$ respectively.

We can now present our Project-and-Lift algorithm.

Algorithm 7.3.12 (Project-and-Lift algorithm)

Input: a set $S \subseteq \mathcal{L}$ that spans $\mathcal{L}$.

Output: a generating G set of $\mathcal{L}$

Find a set $\sigma \subseteq \{1, \ldots, n\}$ such that $\ker(\pi_\sigma) \cap \mathcal{L} = \{0\}$ and $\mathcal{L}^\sigma \cap \mathbb{Z}^{|\overline{\sigma}|}_+ = \{0\}$.

Compute a set $G \subseteq \mathcal{L}^\sigma$ such that G is a generating set of $\mathcal{L}^\sigma$ using S.

```
while $\sigma \neq \emptyset$ do
    Select $i \in \sigma$
    $G := (\pi^{\sigma}_{\{i\}})^{-1}(\mathcal{CP}(\prec^{\sigma}_{\overline{\omega}^i}, G))$
    $\sigma := \sigma \setminus \{i\}$
return $G$.
```

Lemma 7.3.13 *Algorithm 7.3.12 terminates and satisfies its specifications.*

Proof. Algorithm 7.3.12 terminates, since Algorithm 7.2.12 always terminates.

We claim that for each iteration of the algorithm, G is a generating set of $\mathcal{L}^{\sigma}$, $\ker(\pi_{\sigma}) \cap \mathcal{L} = \{0\}$, and $\mathcal{L}^{\sigma} \cap \mathbb{Z}^{|\overline{\sigma}|}_{+} = \{0\}$; therefore, at termination, G is a generating set of $\mathcal{L}$. This is true for the first iteration, so we assume it is true for the current iteration.

If $\sigma = \emptyset$, then there is nothing left to do, so assume otherwise. Since by assumption, $\mathcal{L}^{\sigma} \cap \mathbb{Z}^{|\overline{\sigma}|}_{+} = \{0\}$ and $\ker(\pi_{\sigma}) \cap \mathcal{L}^{\sigma} = \{0\}$, we must have $\ker(\pi^{\sigma}_{\{i\}}) \cap \mathcal{L}^{\sigma\setminus\{i\}} = \{0\}$, and so, the inverse map $(\pi^{\sigma}_{\{i\}})^{-1} : \mathcal{L}^{\sigma} \to \mathcal{L}^{\sigma\setminus\{i\}}$ is well-defined. Let $i \in \sigma$, $G' := (\pi^{\sigma}_{\{i\}})^{-1}(\mathcal{CP}(\prec^{\sigma}_{\{i\}}, G))$, and $\sigma' := \sigma \setminus \{i\}$. Then, by Corollary 7.3.11, G' is a generating set of $\mathcal{L}^{\sigma'}$. Also, since $\sigma' \subseteq \sigma$, we must have $\ker(\pi_{\sigma'}) \cap \mathcal{L} = \{0\}$ and $\mathcal{L}^{\sigma'} \cap \mathbb{Z}^{|\overline{\sigma}'|}_{+} = \{0\}$. Thus, the claim is true for the next iteration. □

In our Project-and-Lift algorithm, we need to find a set $\sigma \subseteq \{1, \ldots, n\}$ such that $\ker(\pi_{\sigma}) \cap \mathcal{L}^{\sigma} = \{0\}$ and $\mathcal{L}^{\sigma} \cap \mathbb{Z}^{|\overline{\sigma}|}_{+} = \{0\}$, and then, we need to compute a generating set of $\mathcal{L}^{\sigma}$.

For our purposes, the larger σ the better. However, in general, finding the largest σ is difficult; thus, we use the following method for finding a *good* σ. Let B be a basis for the lattice $\mathcal{L}$ ($\mathcal{L}$ is spanned by the rows of B). Let $k := \operatorname{rank}(B)$. Any k linearly independent columns of B then suffice to give a set $\overline{\sigma}$ such that every vector in $\mathcal{L}^{\sigma}$ lifts to a unique vector in $\mathcal{L}$; that is, $\ker(\pi_{\sigma}) \cap \mathcal{L}^{\sigma} = \{0\}$. Such a set $\overline{\sigma}$ can be found via Gaussian elimination. If $\mathcal{L}^{\sigma} \cap \mathbb{Z}^{|\overline{\sigma}|}_{+} \neq \{0\}$, then remove some $i \in \sigma$ from σ ($\sigma := \sigma \setminus \{i\}$) and recompute $\mathcal{L}^{\sigma} \cap \mathbb{Z}^{|\overline{\sigma}|}_{+}$. Continue to do so until $\mathcal{L}^{\sigma} \cap \mathbb{Z}^{|\overline{\sigma}|}_{+} = \{0\}$. This procedure must terminate since $\mathcal{L} \cap \mathbb{Z}^{n}_{+} = \{0\}$ by assumption. To check if $\mathcal{L}^{\sigma} \cap \mathbb{Z}^{|\overline{\sigma}|}_{+} = \{0\}$, we can either solve a linear programming problem or compute the extreme rays of $\mathcal{L}^{\sigma} \cap \mathbb{Z}^{|\overline{\sigma}|}_{+}$ (see for example [11, 60, 74]). In practice, we compute extreme rays using the algorithm in [74].

Once we have found such a σ, we can compute a generating set of $\mathcal{L}^{\sigma}$ using either the Saturation algorithm, the Min-Max algorithm, or any other such algorithm. In practice and for this chapter, we use the Saturation algorithm. It is possible that there does not exist such a σ except the trivial case where $\sigma = \emptyset$, and so, the Project-and-Lift algorithm reduces to just the initial phase of computing a generating set of $\mathcal{L}$ using some other algorithm. Though in practice, we usually found a non-trivial σ. We refer the reader to Section 7.4 for a description of a complete Project-and-Lift algorithm whereby we do not need another algorithm to start with.

Example 7.3.14 Consider again the set $S := \{(1, -1, -1, -3, -1, 2), (1, 0, 2, -2, -2, 1)\}$. Let $\mathcal{L}$ be the lattice spanned by S. Let $\sigma = \{3, 4, 5, 6\}$. Then, $\ker(\pi_{\sigma}) \cap \mathcal{L}^{\sigma} = \{0\}$. Note that we have

$\pi_\sigma(S) = \{(1,-1),(1,0)\}$. However, $\mathcal{L}^\sigma \cap \mathbb{Z}_+^{|\overline{\sigma}|} \neq \{0\}$. So, we set $\sigma = \{3,4,6\}$, which leads to $\pi_\sigma(S) = \{(1,-1,-1),(1,0,-2)\}$, and $\mathcal{L}^\sigma \cap \mathbb{Z}_+^{|\overline{\sigma}|} = \{0\}$.

The set $G = \{(0,-1,1),(-1,2,0)\}$ is a generating set of $\mathcal{L}^\sigma$. We can compute this using the saturation algorithm. The following table gives the values of σ, i, ω^i, and G at each stage of the Project-and-Lift algorithm.

σ	i	ω^i	$\mathcal{CP}(\prec^\sigma_{\overline{\omega}^i,G})$	$G := (\pi^\sigma_{\{i\}})^{-1}(\mathcal{CP}(\prec^\sigma_{\omega^i},G))$
$\{3,4,6\}$	3	$(2,3,0)$	$\{(0,-1,1),(1,-2,0)\}$	$\{(0,-1,-3,1),(1,-2,-4,0)\}$
$\{4,6\}$	4	$(-2,1,0,0)$	$\{(0,-1,-3,1),(1,-2,-4,0)\}$	$\{(0,-1,-3,-1,1),(1,-2,-4,-4,0)\}$
$\{6\}$	6	$(1,-1,0,0,0)$	$\{(0,1,3,1,-1),(-1,1,1,3,1)$, $(-1,0,-2,2,2),(-1,-1,-5,1,3)$, $(1,2,8,0,-4)\}$	$\{(0,1,3,1,-1,-1),(-1,1,1,3,1,-2)$, $(-1,0,-2,2,2,-1),(-1,-1,-5,1,3,0)$, $(1,2,8,0,-4,-1)\}$

The final G is a generating set of $\mathcal{L}$. Again note that it is not minimal; the vector $(1,2,8,0,-4,-1)$ is not needed and can be removed from G, and G will still be a generating set of $\mathcal{L}$. □

The concepts of σ-generating sets of $\mathcal{L}$ and generating sets of $\mathcal{L}^\sigma$ are, in fact, equivalent. So, as discussed before, unlike that Saturation algorithm, the Project-and-Lift algorithm computes σ-generating sets and thus does less work than the Saturation algorithm.

Lemma 7.3.15 *Let $\sigma \subseteq \{1,\ldots,n\}$ where $\ker(\pi_\sigma) \cap \mathcal{L} = \{0\}$ and $S \subseteq \mathcal{L}^\sigma$. The set S is a generating set of $\mathcal{L}^\sigma$ if and only if $\pi_\sigma^{-1}(S)$ is a σ-generating set of $\mathcal{L}$.*

Proof. Recall that a σ-generating set of $\mathcal{L}$ is a set where for all $b \in \mathbb{Z}^n$ and for all $x, y \in \mathcal{F}^\sigma_{\mathcal{L},b}$, there exists a path from x to y in $\mathcal{G}(\mathcal{F}^\sigma_{\mathcal{L},b}, S)$. Observe that $\pi_\sigma(\mathcal{F}^\sigma_{\mathcal{L},b}) = \mathcal{F}_{\mathcal{L}^\sigma,\pi_\sigma(b)}$, and moreover, a path in $\mathcal{G}(\mathcal{F}^\sigma_{\mathcal{L},b}, S)$ projects to a path in $\mathcal{G}(\mathcal{F}_{\mathcal{L}^\sigma,\pi_\sigma(b)}, S)$. Hence, a σ-generating set of $\mathcal{L}$ projects to a generating set of $\mathcal{L}^\sigma$. So, if S is a σ-generating set of $\mathcal{L}$, then $\pi_\sigma(S)$ is a generating set of $\mathcal{L}^\sigma$. Also, assuming $\ker(\pi_\sigma) \cap \mathcal{L} = \{0\}$, if S is a generating set of $\mathcal{L}^\sigma$, then $\pi_\sigma^{-1}(S)$ is a σ-generating set of $\mathcal{L}$. This follows since a path in $\mathcal{G}(\mathcal{F}_{\mathcal{L}^\sigma,b}, S)$ can be lifted to a path in $\mathcal{G}(\mathcal{F}^\sigma_{\mathcal{L},\overline{b}}, S)$ where $\pi_\sigma(\overline{b}) = b$. □

Observe that if $\ker(\pi_\sigma) \cap \mathcal{L} \neq \{0\}$, then a path in $\mathcal{G}(\mathcal{F}_{\mathcal{L}^\sigma,b}, \pi_\sigma^{-1}(S))$ cannot necessarily be lifted to a path in $\mathcal{G}(\mathcal{F}^\sigma_{\mathcal{L},\overline{b}}, \pi_\sigma^{-1}(S))$ – the path may become disconnected – although, we can easily rectify this by adding a spanning set of the lattice $\ker(\pi_\sigma) \cap \mathcal{L}$ to $\pi_\sigma^{-1}(S)$.

The Project-and-Lift algorithm has some interesting properties. As we saw in Lemma 7.3.10, $\overline{\omega}^i$-reduction paths lift to $\overline{e}^i$-reduction paths and $\overline{e}^i$-reduction paths project to $\overline{\omega}^i$-reduction paths. The same holds true for $\prec_{\overline{\omega}^i}$-reduction paths and $\prec_{\overline{e}^i}$-reduction paths, shown in exactly the same way, giving the following lemma.

Lemma 7.3.16 *Let $i \in \{1,\ldots,n\}$ where $\ker(\pi_{\{i\}}) \cap \mathcal{L} = \{0\}$, and let $S \subseteq \mathcal{L}^{\{i\}}$. Let $\prec$ be a term order. The set $\pi_{\{i\}}^{-1}(S)$ is a $\prec_{\overline{e}^i}$-Gröbner basis of $\mathcal{L}$ if and only if S is a $\prec_{\overline{\omega}^i}$-Gröbner basis of $\mathcal{L}^{\{i\}}$.*

So, during the Project-and-Lift algorithm, we compute a $\prec_{\overline{\omega}^i}$-Gröbner basis for some i and then lift it to a $\prec_{\overline{e}^i}$-Gröbner basis. We then compute a $\prec_{\overline{\omega}^j}$-Gröbner basis using some $j \neq i$ and again lift it to a $\prec_{\overline{e}^j}$Gröbner basis, and repeat. So effectively, the Project-and-Lift algorithm just converts one Gröbner basis into another and lifts to another Gröbner basis. We therefore could use a Gröbner walk algorithm to move from one Gröbner basis to another (see [32, 59]). We have not yet implemented such an algorithm. It would be interesting to see its performance.

There are, in fact, two essentially equivalent ways to compute a $\prec_{\overline{e}^i}$-Gröbner basis of $\mathcal{L}$ from a set $S \subseteq \mathcal{L}^{\{i\}}$ that generates $\mathcal{L}^{\{i\}}$ for some $i \in \{1, \ldots, n\}$ where $\ker(\pi_{\{i\}}) \cap \mathcal{L} = \{0\}$, as needed by the Project-and-Lift algorithm. Firstly, as we have already seen, the set $T = \pi^{-1}_{\{i\}}(\mathcal{CP}(\prec_{\overline{w}^i}, S))$ is a $\prec_{\overline{e}^i}$-Gröbner basis of $\mathcal{L}$, but also, the set $T' = \mathcal{CP}(\prec_{\overline{e}^i}, \pi^{-1}_{\{i\}}(S))$ is also a $\prec_{\overline{e}^i}$-Gröbner basis of $\mathcal{L}$. Essentially, to compute a $\prec_{\overline{e}^i}$-Gröbner basis of $\mathcal{L}$, we do not need a generating set of $\mathcal{L}$, but instead, we only need a $\{i\}$-generating set of $\mathcal{L}$. This follows from the following Lemmas, which are analogous to Lemmas 7.3.10 and 7.3.16 respectively.

Lemma 7.3.17 *Let $i \in \{1, \ldots, n\}$ where $\ker(\pi_{\{i\}}) \cap \mathcal{L} = \{0\}$, and let $S, T \subseteq \mathcal{L}$ where S is a $\{i\}$-generating set of $\mathcal{L}$. The set T is a $\overline{e}^i$-Gröbner basis of $\mathcal{L}$ if and only if for all $b \in \mathbb{Z}^n$ and for all $x, y \in \mathcal{F}_{\mathcal{L},b}$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$, there exists an $\overline{e}^i$-reduction path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, T)$.*

Proof. The forwards direction must hold by definition. Conversely, let $x, y \in \mathcal{F}_{\mathcal{L},b}$ for some $b \in \mathbb{Z}^n$. Since S is a $\{i\}$-generating set of $\mathcal{L}$, there must exist $\gamma \in \mathbb{Z}^n_+$ where $\text{supp}(\gamma) \subseteq \{i\}$, such that $x+\gamma$ is connected to $y+\gamma$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b+\gamma}, S)$. So, by assumption, there exists an $\overline{e}^i$-reduction path from $x+\gamma$ to $y+\gamma$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b+\gamma}, T)$, which translates to a path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, T)$ since $\text{supp}(\gamma) \subseteq \{i\}$. □

An analogous results holds for $\prec_{\overline{e}^i}$-Gröbner bases for similar reasons.

Lemma 7.3.18 *Let $i \in \{1, \ldots, n\}$ where $\ker(\pi_{\{i\}}) \cap \mathcal{L} = \{0\}$, and let $S, T \subseteq \mathcal{L}$ where S is a $\{i\}$-generating set of $\mathcal{L}$. The set T is a $\prec_{\overline{e}^i}$-Gröbner basis of $\mathcal{L}$ if and only if for all $b \in \mathbb{Z}^n$ and for all $x, y \in \mathcal{F}_{\mathcal{L},b}$ where x and y are connected in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, S)$, there exists an $\prec_{\overline{e}^i}$-reduction path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, T)$.*

Therefore, if $S \subseteq \mathcal{L}^{\{i\}}$ generates $\mathcal{L}^{\{i\}}$, then $\pi^{-1}_{\{i\}}(S)$ is a $\{i\}$-generating set of $\mathcal{L}$ by Lemma 7.3.15. So, the set $T' = \mathcal{CP}(\prec_{\overline{e}^i}, \pi^{-1}_{\{i\}}(S))$ is a $\prec_{\overline{e}^i}$-Gröbner basis of $\mathcal{L}$. Moreover, when computing $\mathcal{CP}(\prec_{\overline{w}^i}, S)$ and computing $\mathcal{CP}(\prec_{\overline{e}^i}, \pi^{-1}_{\{i\}}(S))$, the completion procedure performs essentially the same sequence of steps producing essentially the same output data and intermediate data with the exception that they perform the computation in different spaces. These two approaches are thus algorithmically equivalent.

In one iteration, the Saturation algorithm computes $\mathcal{CP}(\prec_{\overline{e}^i}, T)$, in the space $\mathcal{L}$, for some set $T \subseteq \mathcal{L}$ that is σ-saturated on some spanning set for some $\sigma \subseteq \{1, \ldots, n\}$ and $i \in \overline{\sigma}$. On the other hand, in one iteration, the Project-and-Lift algorithm effectively computes, in the space $\mathcal{L}^{\sigma \setminus \{i\}}$, $\mathcal{CP}(\prec_{\overline{e}^i}, T)$ for some set $T \subseteq \mathcal{L}^{\sigma \setminus \{i\}}$ that is a $\{i\}$-generating set of $\mathcal{L}^{\sigma}$ for some $\sigma \subseteq \{1, \ldots, n\}$ and $i \in \sigma$. So, the algorithms are very similar, but the Project-and-Lift algorithm performs intermediate steps in subspaces whereas the Saturation algorithm performs intermediate steps in the original space.

7.3.3 The "Lift-and-Project" algorithm

The idea behind this algorithm is to lift a spanning set S of $\mathcal{L} \subseteq \mathbb{Z}^n$ to a spanning set $S' \subseteq \mathbb{Z}^{n+1}$ of $\mathcal{L}' \subseteq \mathbb{Z}^{n+1}$ in such a way that we can compute a set $G' \subseteq \mathcal{L}'$ that generates $\mathcal{L}'$ in only one saturation step. Then, we project G' to $G \subseteq \mathcal{L}$, so that G is a generating set of $\mathcal{L}$.

Let S be a spanning set of $\mathcal{L} \subseteq \mathbb{Z}^n$. Let $S' := \{(u, 0) : u \in S\} \cup \{(1, \dots, 1, -1)\}$, and let $\mathcal{L}' \subseteq \mathbb{Z}^{n+1}$ be the lattice spanned by S'. Since the vector $(1, \dots, 1, -1)$ is in S', it follows from Lemma 7.3.6, that if a set $G' \subseteq \mathcal{L}'$ is $\{n+1\}$-saturated on S', then G' is $\{1, \dots, n+1\}$-saturated on S, and hence, G' is a generating set of $\mathcal{L}'$. Also, since $\mathcal{L} \cap \mathbb{Z}^n_+ = \{0\}$, then $\mathcal{L}' \cap \mathbb{Z}^{n+1}_+ = \{0\}$. Now, using exactly the same idea behind the Saturation algorithm, if we let $G' := \mathcal{CP}(\prec_{\overline{e}^{n+1}}, S')$, then G' must be a generating set for the lattice $\mathcal{L}'$ by Lemma 7.3.2.

So, at the moment, we have a generating set G' for $\mathcal{L}'$, and from this, we need to extract a generating set of $\mathcal{L}$. We define the linear map $\rho : \mathbb{Z}^{n+1} \mapsto \mathbb{Z}^n$ where

$$\rho(u') := (u'_1 + u'_{n+1}, u'_2 + u'_{n+1}, \dots, u'_n + u'_{n+1}).$$

Observe that ρ maps $\mathbb{Z}^{n+1}$ onto $\mathbb{Z}^n$, maps $\mathcal{L}'$ onto $\mathcal{L}$, and maps $\mathcal{F}_{\mathcal{L}',b'}$ onto $\mathcal{F}_{\mathcal{L},b}$ where $b = \rho(b')$. Let $G := \{\rho(u') : u' \in G'\} \setminus \{0\}$. So, $G \subset \mathcal{L}$, and we now show that in fact G generates $\mathcal{L}$. Let $({x'}^0, \dots, {x'}^k)$ be a path in $\mathcal{G}(\mathcal{F}_{\mathcal{L}',b'}, G')$. Then, $(\rho({x'}^0), \dots, \rho({x'}^k))$ is a walk from $\rho({x'}^0)$ to $\rho({x'}^k)$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},\rho(b)}, G)$, so after removing cycles, we have a path from $\rho({x'}^0)$ to $\rho({x'}^k)$. Cycles may exist because the kernel of ρ is non-trivial – $\ker(\rho) = \{(\gamma, \dots, \gamma, -\gamma) : \gamma \in \mathbb{Z}\}$. Let $x, y \in \mathcal{F}_{\mathcal{L},b}$ for some $b \in \mathbb{Z}^n$, and let $x' := (x, 0)$, $y' := (y, 0)$, and $b' := (b, 0)$; hence, $\rho(x') = x$, $\rho(y') = y$, and $\rho(b') = b$. Then, since G' is a generating set of $\mathcal{L}'$ there must exist a path from x' to y' in $\mathcal{G}(\mathcal{F}_{\mathcal{L}',b'}, G')$, and therefore, there exists a path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$. Hence, G is a generating set of $\mathcal{L}$. We thus arrive at the Lift-and-Project algorithm.

Algorithm 7.3.19 (Lift-and-Project algorithm)

```
Input: a set $S \subseteq \mathcal{L}$ that spans $\mathcal{L}$.
Output: a generating set $G$ of $\mathcal{L}$

$S' := \{(u, 0) : u \in S\} \cup \{(1, \dots, 1, -1)\}$
$G' := \mathcal{CP}(\prec_{n+1}, S')$
$G := \{\rho(u') : u' \in G'\} \setminus \{0\}$
return $G$.
```

To make the algorithm more efficient, we can use a different additional vector to $(1, \dots, 1, -1)$. By Lemma 7.3.7, we know that given a spanning set S, there exists a σ where $|\sigma| \leq \lfloor \frac{n}{2} \rfloor$ such that if T is σ-saturated on S, then T is a generating set of $\mathcal{L}$. Then, instead of $(1, \dots, 1, -1)$, it suffices to use the additional vector $s_\sigma = \sum_{i \in \sigma} e^i - e^{n+1}$, which has the important property that $\mathrm{supp}(s_\sigma^+) = \sigma$ and $\mathrm{supp}(s_\sigma^-) = \{n+1\}$. Recall that e^i is the ith unit vector. Set $S' := \{(u, 0) : u \in S\} \cup \{s_\sigma\}$, and let $\mathcal{L}'$ be the lattice spanned by S'. Then, from Lemma 7.3.6, since $s_\sigma \in S'$, if a set $G' \subseteq \mathcal{L}'$ is

$\{n+1\}$-saturated on S', then G' is $(\sigma \cup \{n+1\})$-saturated on S'. Also, since $\{(u,0) : u \in S\} \subseteq S'$, from the proof of Lemma 7.3.7, it follows that if G' is σ-saturated on S', then G' is $\{1,\dots,n\}$-saturated. Hence, G' is $\{1,\dots,n+1\}$-saturated, and therefore, a generating set of $\mathcal{L}'$. So again, we can compute a generating set G' of $\mathcal{L}'$ in one saturation step. Also, we similarly define the linear map $\rho_\sigma : \mathbb{Z}^{n+1} \mapsto \mathbb{Z}^n$ where $\rho_\sigma(x') := (x'_1, x'_2, \dots, x'_n) + (\sum_{i\in\sigma} e_i)x_{n+1}$. Then, $G := \{\rho_\sigma(x') : x' \in G'\}$ is a generating set of $\mathcal{L}$. As a general rule, the smaller the size of σ, the faster the algorithm.

7.4 What if $\mathcal{L} \cap \mathbb{Z}^n_+ \neq \{0\}$?

If $\mathcal{L} \cap \mathbb{Z}^n_+ \neq \{0\}$, then computing a generating set of the lattice $\mathcal{L}$ is actually more straight-forward than otherwise. The vectors in $\mathcal{L} \cap \mathbb{Z}^n_+$ are very useful when constructing generating sets.

We say that component $i \in \{1,\dots,n\}$ is *unbounded* if there exists a $u \in \mathcal{L} \cap \mathbb{Z}^n_+$ where $i \in \operatorname{supp}(u)$ and *bounded* otherwise. From Farkas' lemma, i is unbounded if and only if the linear program $\max\{x_i : x \equiv 0 \pmod{\mathcal{L}}, x \in \mathbb{R}^n_+\}$ is unbounded. To find a $u \in \mathcal{L}$ such that $u \gneq 0$ and $i \in \operatorname{supp}(u)$, and so also, to check whether i is unbounded, we can solve a linear program or compute the extreme rays of $\mathcal{L} \cap \mathbb{Z}^n_+$ (see for example [11, 60, 74]). Given a term order $\prec$ of $\mathcal{L}$, the order $\prec_{\overline{e}^i}$ is a term order if and only if i is bounded.

Using the following lemma, we can extend the Saturation algorithm to the more general case where $\mathcal{L} \cap \mathbb{Z}^n_+ \neq \{0\}$.

Lemma 7.4.1 *Let $S \subseteq \mathcal{L}$. If there exists $u \in S$ where $u \in \mathcal{L} \cap \mathbb{Z}^n_+$ and $u \neq 0$, then S is* $\operatorname{supp}(u)$*-saturated (on S).*

Proof. By definition, S is $\emptyset$-saturated (on S). Since $u \geq 0$, we have $\operatorname{supp}(u^-) = \emptyset$, and so it follows immediately from Lemma 7.3.6 that S is $\operatorname{supp}(u)$-saturated (on S). $\square$

We can now extend the Saturation algorithm. Let $\tau \subseteq \{1,\dots,n\}$ be the set of unbounded components, and let S be a spanning set of $\mathcal{L}$. Then, for each $i \in \tau$, find a $u \in \mathcal{L}$ such that $u \geq 0$ and $u_i > 0$ and add u to S. Or equivalently, find a single $u \geq 0$ such that $\operatorname{supp}(u) = \tau$ and add it to S. Now S is τ-saturated (on S) by Lemma 7.4.1. So, if a set T is $\overline{\tau}$-saturated on S, then T is $\{1,\dots,n\}$-saturated on S by Lemma 7.3.3, and so, T is a generating set of $\mathcal{L}$. So, we iteratively compute $T := \mathcal{CP}(\prec_{\overline{e}^i}, T)$ for every $i \in \overline{\tau}$; then, T is $\overline{\tau}$-saturated on S as required.

The Project-and-Lift algorithm can also be extended to the more general case where $\mathcal{L} \cap \mathbb{Z}^n_+ \neq \{0\}$. First, we need the following lemma.

Lemma 7.4.2 *Let $\sigma \subseteq \{1,\dots,n\}$, $S \subseteq \mathcal{L}$, and $u \in \mathcal{L} \cap \mathbb{Z}^n_+$ and $u \neq 0$. If S is a σ-generating set of $\mathcal{L}$ and $u \in S$, then S is a $(\sigma \setminus \operatorname{supp}(u))$-generating set of $\mathcal{L}$.*

Proof. Let $x, y \in \mathcal{F}_{\mathcal{L},b}$ for some $b \in \mathbb{Z}^n$. Since S is a σ-generating set of $\mathcal{L}$, there exists a path from x to y in $\mathcal{G}(\mathcal{F}^\sigma_{\mathcal{L},b}, S)$. This path can transformed into a path in $\mathcal{G}(\mathcal{F}^{(\sigma\setminus\operatorname{supp}(u))}_{\mathcal{L},b}, S)$ by adding u to the start of the path as many times as necessary and subtracting u from the end of the path the same number of times. $\square$

Let $\sigma \subseteq \{1,\ldots,n\}$ where $\ker(\pi_\sigma) \cap \mathcal{L}^\sigma = \{0\}$. Let $S \subseteq \mathcal{L}^\sigma$ where S is a generating set of $\mathcal{L}^\sigma$. Also, let $i \in \sigma$. We now show how to construct a generating set of $\mathcal{L}^{\sigma\setminus\{i\}}$, and thus by induction, a generating set of $\mathcal{L}$. Firstly, since S is a generating set of $\mathcal{L}^\sigma$, $\pi^{-1}_{\{i\}}(S)$ is a $\{i\}$-generating of $\mathcal{L}^{\sigma\setminus\{i\}}$ from Lemma 7.3.15. If i is unbounded for $\mathcal{L}^{\sigma\setminus\{i\}}$, then there exists a $u \in \mathcal{L}^{\sigma\setminus\{i\}}$ such that $u \geq 0$ and $u_i > 0$. Thus, after adding u to $(\pi^\sigma_{\{i\}})^{-1}(S)$, we then have a generating set of $\mathcal{L}^{\sigma\setminus\{i\}}$. If i is bounded, then compute $S := \mathcal{CP}(\prec_{\overline{\omega}^i}, S)$, and $(\pi^\sigma_{\{i\}})^{-1}(S)$ is then a generating set of $\mathcal{L}^{\sigma\setminus\{i\}}$.

We first need to find an initial σ and S such that $\ker(\pi_\sigma) \cap \mathcal{L}^\sigma = \{0\}$ and S is a generating set of $\mathcal{L}^\sigma$. Let B be a lattice basis of $\mathcal{L}$ where the rows of B span $\mathcal{L}$. Let $\overline{\sigma}$ be any rank(B) linearly independent columns of B. Let $S = \pi_\sigma(B)$. Then, every vector in $\mathcal{L}^\sigma$ lifts to a unique vector in $\mathcal{L}$. Then, computing a $u \in \mathcal{L}^\sigma$ such that $u > 0$ can be done by Gaussian elimination. After adding u to S, S is a generating set of $\mathcal{L}^\sigma$ as required.

We can also extend the Lift-and-Project algorithm in a similar way. As above, we can find a set S such that S is τ-saturated (on S). The set

$$S' := \{(u,0) : u \in S\} \cup \left\{ \sum_{i\in\overline{\tau}} e_i - e_{n+1} \right\}$$

is also τ-saturated (on S') for the same reasons. If a set $T' \subseteq \mathcal{L}'$ is $\{n+1\}$-saturated on S', then T' is $(\overline{\tau} \cup \{n+1\})$-saturated on S' by Lemma 7.4.1, and so, by Lemma 7.3.3, T' is $\{1,...,n+1\}$-saturated on S' since S' is τ-saturated (on S'); thus, T' is then a generating set of $\mathcal{L}'$. Hence, in one saturation step, we can compute a generating set of $\mathcal{L}'$. Note that the component $n+1$ is bounded by construction. Then, the set $T := \{\rho_{\overline{\tau}}(u') : u' \in T'\}$ is a generating set of $\mathcal{L}$.

7.5 Speeding-up the completion procedure

Finally, before presenting computational experience, we talk about ways in which the key algorithm, Algorithm 7.2.12, can be improved. This leads us to the critical pair criteria.

Algorithm 7.2.12 has to test for a reduction path between $x^{(u,v)}$ and $y^{(u,v)}$ for all critical pairs $C := \{(u,v) : u,v \in G\}$. In the case of lattice ideals, computational profiling shows that this is the most time consuming part of the computation. So, we wish to reduce the number of critical pairs that we test, and avoid this expensive test as often as possible. We present three criteria that can reduce the number of critical pairs that need to be tested.

Criteria 1 and 3 (see [25, 26, 63]) are translated from the theory of Gröbner bases into a geometric context. Criterion 2 is specific to lattice ideals and corresponds to using the homogeneous Buchberger algorithm [29, 117], but we give a slightly more general result. Note that all three criteria can be applied simultaneously.

Criterion 1: The Disjoint-Positive-Support criterion

For a pair $u, v \in G$, the Disjoint-Positive-Support criterion is a simple and quick test for a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$. So, using this quick test for a $\succ$-reduction path, we can sometimes avoid the more expensive test.

Given $u, v \in G$, if $\mathrm{supp}(u^+) \cap \mathrm{supp}(v^+) = \emptyset$, then there exists a simple $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$ using u and v in reverse order (see Figure 7.5).

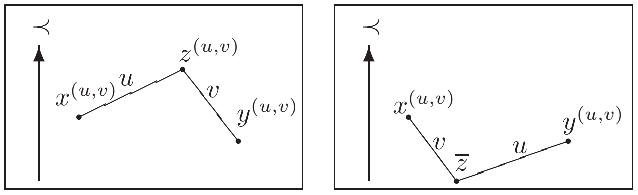


Figure 7.5: Criterion 1.

Criterion 2: The Cancelation criterion

Let G be a generating set of $\mathcal{L}$. If $\mathrm{supp}(x^{(u,v)}) \cap \mathrm{supp}(y^{(u,v)}) \neq \emptyset$ for some $u, v \in G$ (or equivalently, $\mathrm{supp}(u^-) \cap \mathrm{supp}(v^-) \neq \emptyset$), then we do not need to check for a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$ (we can remove the pair (u, v) from C).

To show that this criteria holds, we need the concept of a grading. Let $w \in \mathbb{Q}^n$. If $w^\intercal x = w^\intercal y$ for all $x, y \in \mathcal{F}_{\mathcal{L},b}$ for all $b \in \mathbb{Z}^n$, then we call w a grading of $\mathcal{L}$, and we define $\deg_w(\mathcal{F}_{\mathcal{L},b}) := w^\intercal b$ called the *w-degree* of $\mathcal{F}_{\mathcal{L},b}$. Importantly, if $\mathcal{L} \cap \mathbb{Z}^n_+ = \{0\}$ (which we assume), then it follows from Farkas' lemma that there exists a strictly positive grading $w \in \mathbb{Q}^n_+$ of $\mathcal{L}$.

First, we prove an analogous result to Corollary 7.2.8.

Lemma 7.5.1 *A set $G \subseteq \mathcal{L}_\succ$ is a $\prec$-Gröbner basis of $\mathcal{L}$ if and only if G is a generating set of $\mathcal{L}$ and if for every $\succ$-critical path (x, z, y) in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$ for all $b \in \mathbb{Z}^n$ where* $\mathrm{supp}(x) \cap \mathrm{supp}(y) = \emptyset$*, there exists a $\succ$-reduction path between x and y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$.*

Proof. The forwards implication follows from Corollary 7.2.8. For the backwards implication, we need to show that for every $\succ$-critical path (x, y, z) where $\mathrm{supp}(x) \cap \mathrm{supp}(y) \neq \emptyset$, there exists a $\succ$-reduction path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},x}, G)$, in which case, there is a $\succ$-reduction path for all $\succ$-critical paths, and so by Corollary 7.2.8, G is a Gröbner basis. Assume on the contrary that this is not the case. Let w be a strictly positive grading of $\mathcal{L}$. Among all such $\succ$-critical paths (x, z, y) where $\mathrm{supp}(x) \cap \mathrm{supp}(y) \neq \emptyset$ and there is no $\succ$-reduction path from x to y, choose a $\succ$-critical path (x, z, y) such that $\deg_w(\mathcal{F}_{\mathcal{L},x})$ is minimal. Let $\gamma := x \wedge y$, $\overline{x} := x - \gamma$, and $\overline{y} := y - \gamma$. Note that $\gamma \neq 0$ since $\mathrm{supp}(x) \cap \mathrm{supp}(y) \neq \emptyset$. Because G is a generating set of $\mathcal{L}$, there must exist a path from $\overline{x}$ to $\overline{y}$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},\overline{x}}, G)$. Also, since w is strictly positive, $\deg_w(\mathcal{F}_{\mathcal{L},\overline{x}}) < \deg_w(\mathcal{F}_{\mathcal{L},x})$; therefore, by the minimality assumption on $\deg_w(\mathcal{F}_{\mathcal{L},x})$, we can now conclude that for all $\succ$-critical paths in $\mathcal{G}(\mathcal{F}_{\mathcal{L},\overline{x}}, G)$ there exists a $\succ$-reduction path. Consequently, by Lemma 7.2.6, there exists a $\succ$-reduction path between $\overline{x}$ and $\overline{y}$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},\overline{x}}, G)$. This $\succ$-reduction path, however, can be translated by γ to a $\succ$-reduction path from x to y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},x}, G)$ (see Figure 7.6a). But this contradicts our assumption that there is no such path between x and y. □

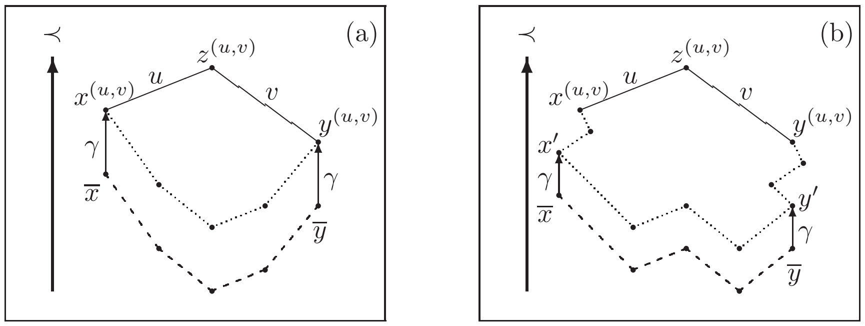


Figure 7.6: Criterion 2.

Now, for all $u, v \in G$, if $\text{supp}(x^{(u,v)}) \cap \text{supp}(y^{(u,v)}) \neq \emptyset$, then $\text{supp}(x) \cap \text{supp}(y) \neq \emptyset$ for all $\succ$-critical paths (x, z, y) for (u, v). Using this observation, we arrive at an analogous result to Corollary 7.2.10.

Corollary 7.5.2 *Let $G \subseteq \mathcal{L}$ be a generating set of $\mathcal{L}$; then, G is a $\prec$-Gröbner basis of $\mathcal{L}$ if and only if for each pair $u, v \in G$ where $\text{supp}(x^{(u,v)}) \cap \text{supp}(y^{(u,v)}) = \emptyset$, there exists a $\succ$-reduction path between $x^{(u,v)}$ and $y^{(u,v)}$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},z^{(u,v)}}, G)$.*

We can extend these results further leading to a more powerful elimination criterion. Let $u, v \in G$. We say the pair (u, v) satisfies Criterion 2 if there exists $x', y' \in \mathcal{F}_{\mathcal{L},z^{(u,v)}}$ such that there exists a $\succ$-decreasing path in $\mathcal{G}(\mathcal{F}_{\mathcal{L},z^{(u,v)}}, G)$ from $x^{(u,v)}$ to x' and from $y^{(u,v)}$ to y', and $\text{supp}(x') \cap \text{supp}(y') \neq \emptyset$. Importantly, if (u, v) satisfies Criterion 2, then we do not have to test for a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$. Thus, we arrive at an extension of Corollary 7.5.2. Observe that the previous results are just a special case where $x' = x^{(u,v)}$ and $y' = y^{(u,v)}$.

Lemma 7.5.3 *Let $G \subseteq \mathcal{L}$ be a generating set of $\mathcal{L}$; then, G is a $\prec$-Gröbner basis of $\mathcal{L}$ if and only if for each pair $u, v \in G$ where (u, v) does not satisfy Criterion 2, there exists a $\succ$-reduction path between $x^{(u,v)}$ and $y^{(u,v)}$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},z^{(u,v)}}, G)$.*

If there is a $\succ$-reduction path from x' to y', then there exists a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$. Since $\text{supp}(x') \cap \text{supp}(y') \neq \emptyset$, then $\gamma = x' \wedge y' \neq 0$. Let $\overline{x} = x' - \gamma$ and $\overline{y} = y' - \gamma$. So, if there exists a $\succ$-reduction path from $\overline{x}$ to $\overline{y}$, then there must exist a $\succ$-reduction path from x' to y', and therefore also, there must exist a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$ (see Figure 7.6b). Again, we let w be a strictly positive grading of $\mathcal{L}$, and so similarly to above, $\deg_w(\mathcal{F}_{\mathcal{L},\overline{x}}) < \deg_w(\mathcal{F}_{\mathcal{L},z^{(u,v)}})$. So, the proof of Lemma 7.5.3 is essentially as before.

For a pair $u, v \in G$, Criterion 2 can be checked not only before we search for a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$ but also while searching for a $\succ$-reduction path. When searching for a $\succ$-reduction path, we construct a $\succ$-decreasing path from $x^{(u,v)}$ to $\text{normalForm}(x^{(u,v)}, G)$ and a $\succ$-decreasing path from $y^{(u,v)}$ to $\text{normalForm}(y^{(u,v)}, G)$. Therefore, we can take any point x' on the $\succ$-decreasing path from $x^{(u,v)}$ to $\text{normalForm}(x^{(u,v)}, G)$ and any point y' on the decreasing path from $y^{(u,v)}$ to normalForm$(y^{(u,v)}, G)$ and check Criterion 2, that is, we check if $\operatorname{supp}(x') \cap \operatorname{supp}(y') \neq \emptyset$. If this is true, then we can eliminate (u, v).

We wish to point out explicitly here that Criterion 2 can be applied without choosing the vector pairs $u, v \in G$ in a particular order during Algorithm 7.2.12. In fact, when running Algorithm 7.2.12, if we apply Criterion 2 to eliminate a pair $u, v \in G$, it does not necessarily mean that there is a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},z^{(u,v)}}, G)$ at that particular point in time in the algorithm but instead that a $\succ$-reduction path will exist when the algorithm terminates. This approach is in contrast to existing approaches that use the homogeneous Buchberger algorithm to compute a Gröbner basis whereby vector pairs $u, v \in G$ must be chosen in an order compatible with increasing $\deg_w(\mathcal{F}_{\mathcal{L},z^{(u,v)}})$ for some strictly positive grading w. This can be computationally costly. When we use these existing approaches, if a pair (u, v) is eliminated by Criterion 2, then it is necessarily the case that there already exists a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$.

Since we need a generating set of $\mathcal{L}$ for Criterion 2, we cannot apply Criterion 2 during the Saturation algorithm (Algorithm 7.3.4), and also, we cannot apply Criterion 2 when $\mathcal{L} \cap \mathbb{Z}^n_+ \neq \{0\}$. However, we can apply a less strict version. Given $u, v \in G$ and $\tau \subseteq \{1, \ldots, n\}$, we say that (u, v) satisfies Criterion 2 with respect to τ, if there exists $x', y' \in \mathcal{F}_{\mathcal{L},z^{(u,v)}}$ such that there exist decreasing paths in $\mathcal{G}(\mathcal{F}_{\mathcal{L},z^{(u,v)}}, G)$ from $x^{(u,v)}$ to x' and from $y^{(u,v)}$ to y', and $\operatorname{supp}(x') \cap \operatorname{supp}(y') \cap \tau \neq \emptyset$.

Let $S, T \subseteq \mathcal{L}$ where S spans $\mathcal{L}$ and T is a σ-saturated set on S for some $\sigma \subseteq \{1, \ldots, n\}$. During an iteration of the Saturation algorithm 7.3.4, we compute a $(\sigma \cup \{i\})$-saturated set of S, by computing $\mathcal{CP}(\prec_{\bar{e}^i}, T)$. While computing $\mathcal{CP}(\prec_{\bar{e}^i}, T)$ here, we may apply Criterion 2 with respect to σ. For an algebraic proof of this, see [21].

Also, we can use Criterion 2 when $\mathcal{L} \cap \mathbb{Z}^n_+ \neq \{0\}$ if we have a generating set of $\mathcal{L}$. Let τ be the set of bounded components. Then, we may apply Criterion 2 with respect to τ. Moreover, if we do not have a generating set and we are running the Saturation algorithm when $\mathcal{L} \cap \mathbb{Z}^n_+ \neq \{0\}$, we may apply Criterion 2 with respect to $\sigma \cap \tau$.

Criterion 3: The (u, v, w) criterion

Before presenting the (u, v, w) criterion, we need a another result, Lemma 7.5.4, that is a less strict version of Lemma 7.2.6. First, we need to define a new type of path. A path $(x^0, \ldots, x^k)$ is z-**bounded** (with respect to $\prec$) if $x^i \prec z$ for all $i = 0, \ldots, k$. So, z is a strict upper bound on the path. Note that for a $\succ$-critical path (x, z, y), a $\succ$-reduction path from x to y is a z-bounded path.

Lemma 7.5.4 *Let $b \in \mathbb{Z}^n$, $x, y \in \mathcal{F}_{\mathcal{L},b}$, and let $G \subseteq \mathcal{L}_\succ$ where there is a path between x and y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$. If there exists a z'-bounded path between x' and y' for every $\succ$-critical path (x', z', y') in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$, then there exists a $\succ$-reduction path between x and y in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$.*

If we now re-examine the proof of Lemma 7.5.4, we find that we only need z'-bounded paths between x' and y' for every $\succ$-critical path (x', z', y') in $\mathcal{G}(\mathcal{F}_{\mathcal{L},b}, G)$, and that, a $\succ$-reduction path from x' and y' is more than we need. The proof proceeds in the same way as Lemma 7.2.6.

From Lemma 7.5.4, we arrive at an analogous result to Corollary 7.2.10.

Corollary 7.5.5 *A set $G \subseteq \mathcal{L}_{\succ}$ is a $\prec$-Gröbner basis of $\mathcal{L}$ if and only if G is a generating set of $\mathcal{L}$ and if for each pair $u, v \in G$, there exists a $z^{(u,v)}$-bounded path between $x^{(u,v)}$ and $y^{(u,v)}$ in $\mathcal{G}(\mathcal{F}_{\mathcal{L},z^{(u,v)}}, G)$.*

Corollary 7.5.5 does not fundamentally change Algorithm 7.2.12 since to test for a $z^{(u,v)}$-bounded path from $x^{(u,v)}$ to $y^{(u,v)}$, we still test for a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$ which is a $z^{(u,v)}$-bounded path. However, we can use Corollary 7.5.5 to reduce the number of critical pairs $u, v \in G$ for which we need to compute a $\succ$-reduction path.

Now, we are able to present the (u, v, w) criterion. Let $u, v, w \in G$ where $z^{(u,v)} \geq w^+$ (or equivalently, $z^{(u,v)} \geq z^{(u,w)}$ and $z^{(u,v)} \geq z^{(w,v)}$), and let $\overline{z} = z^{(u,v)} - w$. Then, a $z^{(u,v)}$-bounded path from $x^{(u,w)}$ to $\overline{z}$, and a $z^{(u,v)}$-bounded path from $\overline{z}$ to $y^{(w,v)}$ combine to form a $z^{(u,v)}$-bounded path from $x^{(u,v)}$ to $y^{(u,v)}$. Moreover, $(x^{(u,v)}, z^{(u,v)}, \overline{z})$ is a $\succ$-critical path for (u, w) and $(\overline{z}, z^{(u,v)}, y^{(u,v)})$ is a $\succ$-critical path for (w, v) (see Figure 7.7). Therefore, a $z^{(u,w)}$-bounded path from $x^{(u,w)}$ to $y^{(u,w)}$ and a $z^{(w,v)}$-bounded path from $x^{(w,v)}$ to $y^{(w,v)}$ combine to form a $z^{(u,v)}$-bounded path from $x^{(u,v)}$ to $y^{(u,v)}$, and so, we can remove (u, v) from C.

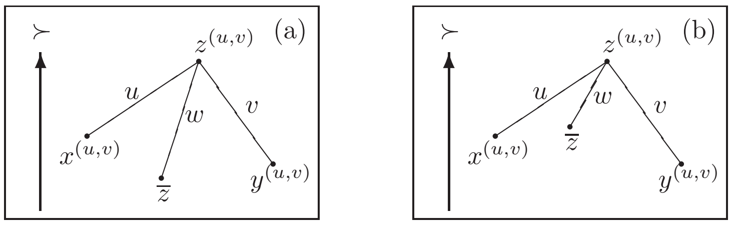


Figure 7.7: Criterion 3.

Note that in Figure 7.7a, a $\succ$-reduction path from $x^{(u,v)}$ to $\overline{z}$ and a $\succ$-reduction path from $\overline{z}$ to $y^{(u,v)}$ do combine to give a $\succ$-reduction path from $x^{(u,v)}$ to $y^{(u,v)}$; however, this is not the case in Figure 7.7b which is why we need the concept of bounded paths.

We can extend the previous result. Let $u, v, \in G$, and $w^1, \ldots, w^k \in G$ where $z^{(u,v)} \geq (w^i)^+$ for all $i = 1, \ldots, k$. If there exists a bounded path for the critical pairs (u, w^1), (w^k, v), and (w^i, w^{i+1}) for all $i = 1, \ldots, k-1$, then there is a bounded path for (u, v). However, note that this can also be implied by a bounded path for (u, w^i) and (w^i, v) for any $i = 1, \ldots, k$.

Unfortunately, we cannot just remove from C all pairs $u, v \in G$ where there exists a $w \in G$ such that $z^{(u,v)} \geq w^+$. It may happen that in addition to $z^{(u,v)} \geq w^+$, we also have $z^{(u,w)} \geq v^+$, in which case, we would eliminate both the pairs (u, v) and (u, w) leaving only (v, w) which is not sufficient. Moreover, at the same time, we may also have $z^{(w,v)} \geq u^+$, and we would eliminate all three pairs. To avoid these circular relationships, Gebauer and Möller [63] devised the following critical pair elimination criteria which we use in practice in `4ti2` v1.2.

Let $G = \{u^1, u^2, \ldots, u^{|G|}\}$, and let $u^i, u^j \in G$ where $i < j$. We define that the pair (u^i, u^j) satisfies Criterion 3 if there exists $u^k \in G$ such that one of the following conditions hold:

1. $z^{(u^i,u^j)} \gneq z^{(u^i,u^k)}$, and $z^{(u^i,u^j)} \gneq z^{(u^j,u^k)}$;
2. $z^{(u^i,u^j)} = z^{(u^i,u^k)}$, $z^{(u^i,u^j)} \gneq z^{(u^j,u^k)}$, and $k < j$;
3. $z^{(u^i,u^j)} \gneq z^{(u^i,u^k)}$, $z^{(u^i,u^j)} = z^{(u^j,u^k)}$, and $k < i$; or
4. $z^{(u^i,u^j)} = z^{(u^i,u^k)} = z^{(u^j,u^k)}$, and $k < i < j$.

So, if a pair (u^i, u^j) satisfies Criterion 3, we can eliminate it. For example, if $G = \{u^1, u^2, u^3\}$ where $z^{u^1u^2} = z^{u^1u^3} \gneq z^{u^2u^3}$, then applying Criterion 3 to all three pairs (u^1, u^2), (u^1, u^3), and (u^2, u^3) would eliminate only (u^1, u^3).

After eliminating all pairs that satisfy Criterion 3, we are left with a set $C' \subseteq C = \{(u, v) : u, v \in G\}$ of critical pairs such that if there exists a $z^{(u',v')}$-bounded path from $x^{(u',v')}$ to $y^{(u',v')}$ for all $(u', v') \in C'$, then there exists a $z^{(u,v)}$-bounded path from $x^{(u,v)}$ to $y^{(u,v)}$ for all $(u, v) \in C$. However, this set of pairs may not be minimal. In [29], Caboara, Kreuzer, and Robbiano describe an algebraic algorithm for computing a minimal set of critical pairs with computational results. Their computational results show that the Gebauer and Möller criteria give a good approximation to the minimal set of critical pairs. We found that the Gebauer and Möller criteria were sufficient for our computations.

7.6 The $4 \times 4 \times 4$-challenge

The challenge posed by Seth Sullivant amounts to checking whether a given set of 145, 512 integer vectors in $\mathbb{Z}^{64}$ is a Markov basis for the statistical model of $4 \times 4 \times 4$ contingency tables with 2-marginals. If $x = (x_{ijk})_{i,j,k=1,\ldots,4}$ denotes a $4 \times 4 \times 4$ array of integer numbers, the defining equations for the sampling moves are

$$\sum_{i=1}^{4} x_{ijk} = 0 \qquad \text{for } j, k = 1, \ldots, 4,$$
$$\sum_{j=1}^{4} x_{ijk} = 0 \qquad \text{for } i, k = 1, \ldots, 4,$$
$$\sum_{k=1}^{4} x_{ijk} = 0 \qquad \text{for } i, j = 1, \ldots, 4.$$

This leads to a problem matrix $A_{444} \in \mathbb{Z}^{48\times 64}$ of rank 37 and $\mathcal{L}_{A_{444}} = \{z : A_{444}z = 0, z \in \mathbb{Z}^{64}\}$. Note that the 145, 512 vectors in the conjectured Markov basis fall into 14 equivalence classes under the natural underlying symmetry group $S_4 \times S_4 \times S_4 \times S_3$.

In [9], Aoki and Takemura have computed these 14 symmetry classes via an analysis of sign patterns and under exploitation of symmetry. They claimed that the corresponding 145, 512 vectors form the unique inclusion-minimal Markov basis of A_{444}.

Using our Project-and-Lift algorithm, however, we have computed the Markov basis from the problem matrix A_{444} within less than 7 days on a Sun Fire V890 Ultra Sparc IV processor with 1200 MHz. Note that the symmetry of the problem was *not* used by the algorithm. This leaves room for a further significant speed-up. Our computation produced $148,968$ vectors; that is, there are additionally $3,456$ Markov basis elements. These vectors form a single equivalence class under $S_4 \times S_4 \times S_4 \times S_3$ of a norm 28 vector z_{15} (or equivalently, of a degree 14 binomial).

A quick check via a Hilbert basis computation with `4ti2` shows that these Markov basis elements are indispensable, since $\{z \in \mathbb{Z}_+^{64} : A_{444} z = A_{444} z_{15}^+\} = \{z_{15}^+, z_{15}^-\}$. As also all the other $145,512$ Markov basis elements were indispensable, the Markov basis of $4 \times 4 \times 4$ contingency tables with 2-marginals is indeed unique. At least this claim can be saved from [9], although we have finally given a computational proof. Here is the list of the 15 orbit representatives, written as binomials:

1. $x_{111}x_{144}x_{414}x_{441} - x_{114}x_{141}x_{411}x_{444}$
2. $x_{111}x_{144}x_{334}x_{341}x_{414}x_{431} - x_{114}x_{141}x_{331}x_{344}x_{411}x_{434}$
3. $x_{111}x_{122}x_{134}x_{143}x_{414}x_{423}x_{432}x_{441} - x_{114}x_{123}x_{132}x_{141}x_{411}x_{422}x_{434}x_{443}$
4. $x_{111}x_{144}x_{324}x_{333}x_{341}x_{414}x_{423}x_{431} - x_{114}x_{141}x_{323}x_{331}x_{344}x_{411}x_{424}x_{433}$
5. $x_{111}x_{144}x_{234}x_{243}x_{323}x_{341}x_{414}x_{421}x_{433} - x_{114}x_{141}x_{233}x_{244}x_{321}x_{343}x_{411}x_{423}x_{434}$
6. $x_{111}x_{122}x_{133}x_{144}x_{324}x_{332}x_{341}x_{414}x_{423}x_{431} - x_{114}x_{123}x_{132}x_{141}x_{322}x_{331}x_{344}x_{411}x_{424}x_{433}$
7. $x_{111}x_{144}x_{222}x_{234}x_{243}x_{323}x_{341}x_{414}x_{421}x_{432} - x_{114}x_{141}x_{223}x_{232}x_{244}x_{321}x_{343}x_{411}x_{422}x_{434}$
8. $x_{111}x_{144}x_{222}x_{233}x_{324}x_{332}x_{341}x_{414}x_{423}x_{431} - x_{114}x_{141}x_{223}x_{232}x_{322}x_{331}x_{344}x_{411}x_{424}x_{433}$
9. $x_{111}x_{112}x_{133}x_{144}x_{223}x_{224}x_{232}x_{241}x_{314}x_{322}x_{413}x_{421} -$
   $x_{113}x_{114}x_{132}x_{141}x_{221}x_{222}x_{233}x_{244}x_{312}x_{324}x_{411}x_{423}$
10. $x_{111}x_{112}x_{133}x_{144}x_{224}x_{232}x_{243}x_{313}x_{322}x_{341}x_{414}x_{421} -$
    $x_{113}x_{114}x_{132}x_{141}x_{222}x_{233}x_{244}x_{312}x_{321}x_{343}x_{411}x_{424}$
11. $x_{111}x_{134}x_{143}x_{222}x_{233}x_{241}x_{314}x_{323}x_{342}x_{412}x_{424}x_{431} -$
    $x_{114}x_{133}x_{141}x_{223}x_{231}x_{242}x_{312}x_{324}x_{343}x_{411}x_{422}x_{434}$
12. $x_{111}x_{134}x_{143}x_{224}x_{232}x_{241}x_{314}x_{323}x_{342}x_{412}x_{421}x_{433} -$
    $x_{114}x_{133}x_{141}x_{221}x_{234}x_{242}x_{312}x_{324}x_{343}x_{411}x_{423}x_{432}$
13. $x_{111}^2x_{124}x_{133}x_{144}x_{214}x_{223}x_{242}x_{313}x_{332}x_{341}x_{414}x_{424}x_{431} -$
    $x_{114}^2x_{123}x_{131}x_{141}x_{213}x_{222}x_{244}x_{311}x_{333}x_{342}x_{411}x_{424}x_{432}$
14. $x_{111}^2x_{124}x_{133}x_{144}x_{214}x_{232}x_{243}x_{312}x_{323}x_{341}x_{414}x_{422}x_{431} -$
    $x_{114}^2x_{123}x_{131}x_{141}x_{212}x_{233}x_{244}x_{311}x_{322}x_{343}x_{411}x_{424}x_{432}$
15. $x_{111}^2x_{133}x_{144}x_{223}x_{224}x_{232}x_{242}x_{313}x_{322}x_{341}x_{414}x_{422}x_{431} -$
    $x_{113}x_{114}x_{131}x_{141}x_{222}^2x_{233}x_{244}x_{311}x_{323}x_{342}x_{411}x_{424}x_{432}$

7.7 Computational experience

We now compare the implementation of our new algorithm in `4ti2` v.1.2 [70] with the implementation of the Saturation algorithm [78] and the Lift-and-Project algorithm [21] in Singular v3.0.0 [66] (algorithmic options 'hs' and 'blr') and in CoCoA 4.2 [30] (functions 'Toric' and 'Toric.Sequential').

Name	Software	Function	Algorithm
Sing-blr	Singular v3.0.0	toric, option "blr"	Lift-and-Project
Sing-hs	Singular v3.0.0	toric, option "hs"	Saturation
CoCoA-t	CoCoA v4.2	Toric	Lift-and-Project
CoCoA-ts	CoCoA v4.2	Toric.Sequential	Saturation
P&L	`4ti2` v1.2	groebner	Project-and-Lift
`4ti2`-gra	`4ti2` v1.2	graver	Graver basis [76]

The first 4 problems correspond to three-way tables with 2-marginals, whereas $K4$ and $K5$ correspond to the binary models on the complete graphs K_4 and K_5, respectively. The problem `s-magic333` is taken from an application in [3] and computes the relations among the 66 elements of the Hilbert basis elements of $3 \times 3 \times 3$ semi-magic hypercubes. The example `grin` is taken from [78], while the examples `hppi10-hppi14` correspond to the computation of homogeneous primitive partition identities, see for example Chapters 6 and 7 in [114]. Finally, the examples `cuww1-cuww5` arise from knapsack problems presented in [35].

The computations were done on a Sun Fire V890 Ultra Sparc IV processor with 1200 MHz. Computation times are given in seconds, rounded up. See Figure 7.8. The running times give a clear ranking of the implementations: from left to write the speed increases and in *all* problems, the presented Project-and-Lift algorithm wins significantly.

Problem	Vars.	GB size	Sing-blr	Sing-hs	CoCoA-t	CoCoA-ts	P&L	`4ti2`-gra
333	27	110	30	4	1	1	1	
334	36	626	–	197	3,024	5	1	
335	45	3,260	–	23,700	–	233	27	
344	48	7,357	–	–	–	2,388	252	
K4	16	61	1	1	1	1	1	
K5	32	13,181	–	–	13,366	2,814	715	
s-magic333	66	1,664	–	–	35	55	3	
grin	8	214	4	4	1	1	1	
hppi10	20	1,830	1,064	483	16	14	3	2
hppi11	22	3,916	15,429	3,588	129	82	13	11
hppi12	24	8,569	–	43,567	1,534	554	60	51
hppi13	26	16,968	–	–	8,973	4,078	290	259
hppi14	28	34,355	–	–	–	30,973	1,219	1,126
cuww1	5	5	–	–	–	–	1	
cuww2	6	15	–	–	–	–	1	
cuww3	6	16	–	–	–	–	2	
cuww4	7	7	–	–	–	–	1	
cuww5	8	27	–	–	–	–	2	

Figure 7.8: Comparison of computing times.

The advantage of our Project-and-Lift algorithm is that it performs computations in projected subspaces of $\mathcal{L}$. Thus, we obtain comparably small intermediate sets during the computation. Only the final iteration that deals with all variables reaches the true output size. In contrast to this, the Saturation algorithm usually comes close to the true output size already after the first saturation and then continues computing with as many vectors. See Figure 7.9, for a comparison of intermediate set sizes in each iteration for computing $3 \times 4 \times 4$ tables.

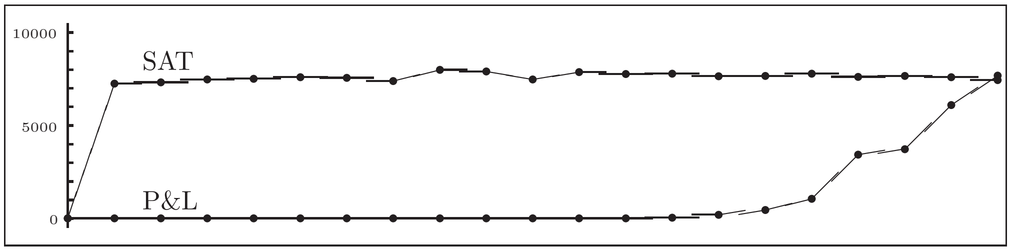


Figure 7.9: Comparison of intermediate set sizes in each iteration.

Moreover, the Project-and-Lift algorithm, performs Gröbner basis computations using a generating set, and thus can take full advantage of Criterion 2 which, as computational experience shows, is extremely effective. In fact, we only applied Criterion 2 and 1 (applied in that order) for the Project-and-Lift algorithm since Criterion 3 only slowed down the algorithm. However, for the Saturation algorithm where we cannot apply Criterion 2 fully, Criterion 3 was very effective. In this case, we applied Criterion 1, then 3, and then 2, in that order.

Note that in the knapsack problems `cuww1-cuww5` the initial set σ chosen by the Project-and-Lift Algorithm 7.3.12 is empty. Thus, Algorithm 7.3.12 simplifies to the Saturation Algorithm 7.3.4. In fact, only a single saturation is necessary for each problem.

To us, the following observations were surprising.

- While Singular did not accept the inhomogeneous problems `cuww1-cuww5` as input, CoCoA either could not solve them or produced incorrect answers.
- It is not clear why the CoCoA function Toric works well on problems `hppi10-hppi14`, but runs badly on the table problems 334, 335, 344.
- Problems `hppi10-hppi14` are in fact Graver basis computations (see for example Chapter 14 in [114]), for which `4ti2` has the state-of-the-art algorithm and implementation. Initially, it was a surprise to us that our Project-and-Lift Algorithm 7.3.12 comes so close to the speed of the state-of-the-art algorithm that computes Graver bases directly, Chapter 2. However, it turns out that our Project-and-Lift Algorithm 7.3.12 is an extension of the Project-and-Lift algorithm presented in Chapter 2 to lattice ideal computations.

Part II

Short rational generating functions

Chapter 8

Counting via Hilbert bases

8.1 The counting problem

Counting lattice points inside convex rational polyhedra is a truly fundamental and useful step in many mathematical investigations. It appears, for instance, in the context of Combinatorics [94, 110], Representation Theory [87, 107], Statistics [49, 57], and Number Theory [17, 98]. Lattices and polytopes are also at the foundation of Discrete Optimization [67, 106]. This justifies the development of algorithms and of computer software that could count or list all lattice points in an arbitrary *rational* convex polyhedron.

In fact, we are not only interested in counting the lattice points in P, but we wish to do the same for dilations of P, where for $P = \{x : Ax \leq b, x \in \mathbb{R}^n\}$ and $t > 0$ we call

$$tP := \{tx : x \in P\} = \{x : Ax \leq tb, x \in \mathbb{R}^n\}$$

the *t-dilation* of P. Then the function $i_P : \mathbb{Z}_+ \to \mathbb{Z}_+$ defined by

$$i_P(t) := |tP \cap \mathbb{Z}^n|$$

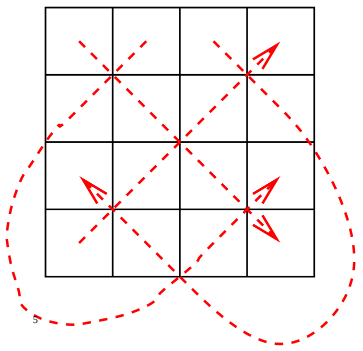
is what we are interested in. This function i_P was first studied by Ehrhart [53] and has received a lot of attention in combinatorics. It is known to be a polynomial when all vertices of P are integral and it is a quasi-polynomial for arbitrary rational polytopes. It is called the *Ehrhart quasi-polynomial* in honor of its discoverer [110, Chapter 4]. A function $f : \mathbb{N} \to \mathbb{C}$ is called a *quasi-polynomial* if there is an integer $N > 0$ and polynomials $f_0, \ldots, f_{N-1}$ such that $f(s) = f_i(s)$ if $s \equiv i \pmod{N}$. The integer N is called a *quasi-period* of f. Therefore, by counting the number of lattice points for sufficiently many dilations of a rational polytope, we could interpolate its Ehrhart quasi-polynomial.

In this chapter and in the following chapter, we present two algorithms to find i_P. Both algorithmic approaches find a "short" rational function expression that encodes the infinite sum $\sum_{t=0}^{\infty} i_P(t) z^t$, the so-called *Ehrhart series*. While the first approach uses only concepts already introduced in this thesis, Hilbert bases of cones and Gröbner bases of toric ideals, the second approach introduces short rational generating functions to encode sets of lattice points together with Barvinok's algorithm to find such a representation.

As a simple application of counting, let us present the problem of counting magic squares and hypercubes. Magic squares and cubes are very popular combinatorial objects (see [7, 61, 101] and their references).

A *magic square* is a square matrix whose entries are non-negative integers and whose row sums, column sums, and main diagonal sums add up to the same integer number s. We call s the *magic sum* of the square. In the literature there have been many variations on the definition of magic squares. For example, one popular variation of our definition adds the restriction of using the integers $1, \ldots, n^2$ as entries (such magic squares are commonly called *natural* or *pure* and a large part of the literature consists of procedures for constructing such examples, see [7, 61, 101]). We restrict the entries of the squares to arbitrary non-negative integers and consider other kinds of restrictions instead:

Semi-magic squares is the case when only the row and column sums are considered. This apparent simplification has in fact a very rich theory and several open questions remain (see [31, 55, 111] and references within. Semi-magic squares are called magic squares in these references). *Pandiagonal magic squares* are magic squares with the additional property that any broken-line diagonal sum adds up to the same integer (see Figure 8.1).



12	0	5	7
0	12	7	5
7	5	0	12
5	7	12	0

Figure 8.1: Four broken diagonals of a square and a pandiagonal magic square.

There are analogous definitions in higher dimensions. A *semi-magic hypercube* is a d-dimensional $n \times n \times \cdots \times n$ array of n^d non-negative integers, which sum up to the same number s for any line parallel to some axis. A *magic hypercube* is a semi-magic cube that has the additional property that the sums of all the main diagonals, the 2^{d-1} copies of the diagonal $x_{1,1,\ldots,1}, x_{2,2,\ldots,2}, \ldots, x_{n,n,\ldots,n}$ under the symmetries of the d-cube, are also equal to the magic sum. For example, in a $2 \times 2 \times 2$ cube there are 4 diagonals with sums $x_{1,1,1}+x_{2,2,2} = x_{2,1,1}+x_{1,2,2} = x_{1,1,2}+x_{2,2,1} = x_{1,2,1}+x_{2,1,2}$. We can see a magic $3 \times 3 \times 3$ cube in Figure 8.2 (the number 14 is at the central $(2, 2, 2)$ position). From now on, when referring to any of these structures, we will use the terminology *magic arrays*.

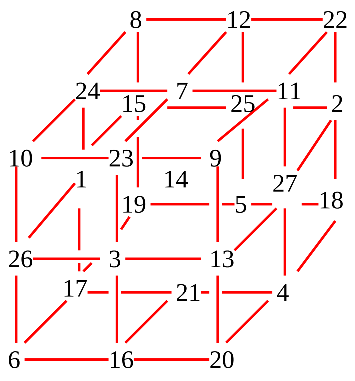


Figure 8.2: A magic cube.

Example 8.1.1 As an example we consider the case of 3×3 magic squares of magic sum t, whose counting formula has been know at least since 1915 [94]. Any magic 3×3 square with magic sum t lies in the t-dilation of the polytope $P_{3\times3}$ described by the system of equations

$$\begin{aligned}
x_{11} + x_{12} + x_{13} &= 1,\\
x_{21} + x_{22} + x_{23} &= 1,\\
x_{31} + x_{32} + x_{33} &= 1,\\
x_{11} + x_{21} + x_{31} &= 1,\\
x_{12} + x_{22} + x_{32} &= 1,\\
x_{13} + x_{23} + x_{33} &= 1,\\
x_{11} + x_{22} + x_{33} &= 1,\\
x_{31} + x_{22} + x_{13} &= 1,
\end{aligned}$$

and the inequalities $x_{ij} \geq 0$.

Using for example the method described in the next section, we obtain the counting formula

$$\sum_{t=0}^{\infty} i_{P_{3\times3}}(t) z^t = \frac{(1+z^3)^2}{(1-z^3)^3} = 1 + 5z^3 + 13z^6 + 25z^9 + 41z^{12} + \dots$$

or more explicitly

$$i_{P_{3\times3}}(t) = \begin{cases} \frac{2}{9}t^2 + \frac{2}{3}t + 1 & \text{if } 3|t, \\ 0 & \text{otherwise.} \end{cases}$$

The latter counting formula tells us that there exist magic 3×3 squares only for magic sums which are divisible by 3. In fact, it can be shown that this is true for any magic $3 \times 3 \times \cdots \times 3$ hypercube.

8.2 Hilbert bases for counting and element generation

In this section, we use Hilbert bases to find a short rational expression for $\sum_{t=0}^{\infty} i_P(t) z^t$ for given polytope $P \subseteq \mathbb{R}^n$. We use computational polyhedral geometry and commutative algebra techniques to derive this explicit counting formula. Similar derivations had been done earlier for semi-magic squares [110, Chapter 4].

Let us consider the homogenization C of P, which is the (pointed rational polyhedral) cone of all (continuous) dilations tP, $t \geq 0$:

$$C := \{(x,t) : x \in tP, x, \in \mathbb{R}^n, t \geq 0\}.$$

Then, the lattice points in C are in one-to-one correspondence to the lattice points in tP where t ranges over all non-negative integers. Let H denote the unique inclusion-minimal Hilbert basis of C. Thus, every lattice point in C can be represented as a non-negative integer linear combination of elements from H. Unfortunately, this representation is not unique.

Example 8.2.1 Let us consider the cone of magic 3×3 squares. This cone is defined by the system of equations

$$\begin{array}{rclclclcl}
t & = & x_{11}+x_{12}+x_{13} & & & & & & \text{magic sum } t \\
x_{11}+x_{12}+x_{13} & = & x_{21}+x_{22}+x_{23} & = & x_{31}+x_{32}+x_{33} & & & & \text{row sums} \\
x_{11}+x_{12}+x_{13} & = & x_{11}+x_{21}+x_{31} & = & x_{12}+x_{22}+x_{32} & = & x_{13}+x_{23}+x_{33} & & \text{column sums} \\
x_{11}+x_{12}+x_{13} & = & x_{11}+x_{22}+x_{33} & = & x_{31}+x_{22}+x_{13} & & & & \text{diagonal sums}
\end{array}$$

and the inequalities $x_{ij}, t \geq 0$. This cone C of magic 3×3 squares lives in $\mathbb{R}^{10}$ and is of dimension 3, it is a cone based on a quadrilateral, thus it has 4 rays (see Figure 8.3). The 5 elements in the Hilbert basis of C correspond to those magic 3×3 squares that cannot be written as a non-trivial sum of two other magic 3×3 squares.

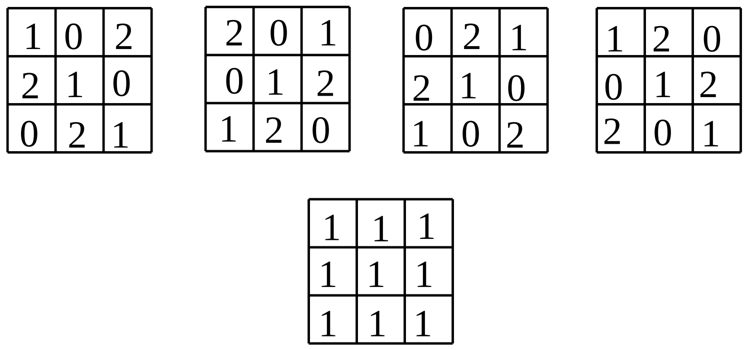


Figure 8.3: The Hilbert basis for the cone of 3×3 magic squares projected down to the x-components living in $\mathbb{R}^9$. The top four squares are the rays of the cone.

We can already recover from the Hilbert basis the fact that there exist only magic 3×3 squares of magic sums that are divisible by 3, since every Hilbert basis element has magic sum 3.

To compute the above Hilbert bases, we use `4ti2` as follows. First, we specify the problem matrix in a file, say "3x3":

```
8 10
1 1 1  0  0  0  0  0  0 -1
1 1 1 -1 -1 -1  0  0  0  0
1 1 1  0  0  0 -1 -1 -1  0
0 1 1 -1  0  0 -1  0  0  0
1 0 1  0 -1  0  0 -1  0  0
1 1 0  0  0 -1  0  0 -1  0
0 1 1  0 -1  0  0  0 -1  0
1 1 0  0 -1  0 -1  0  0  0
```

and then call "hilbert 3x3" which produces a file named "3x3.hil":

```
5 10
1 0 2 2 1 0 0 2 1 3
2 0 1 0 1 2 1 2 0 3
0 2 1 2 1 0 1 0 2 3
1 2 0 0 1 2 2 0 1 3
1 1 1 1 1 1 1 1 1 3
```

containing the desired 5 Hilbert basis elements. □

Having a Hilbert basis of C available, let us show how it can be used to find $\sum_{t=0}^{\infty} i_P(t)z^t$ as a rational expression in z.

With any d-dimensional rational pointed polyhedral cone $C = \{Ax = 0, x \geq 0\}$ and a field k we associate a *semigroup* $S_C = (C \cap \mathbb{Z}^n, +)$ and a *semigroup ring* $R_C = k[y^a : a \in S_C]$, where there is one monomial $y^a = y_1^{a_1} y_2^{a_2} \dots y_n^{a_n}$ in the ring for each element $a = (a_1, \dots, a_n)$ of the semigroup S_C. By the definition of a Hilbert basis we know that every element of S_C can be written as a finite linear combination $\sum \mu_i h_i$ where the μ_i are non-negative integers. Thus R_C is in fact a finitely generated k-algebra, with one generator per element of a Hilbert basis. Therefore R_C can be written as the quotient $k[x_1, x_2, \dots, x_N]/I_C$: Once we have the Hilbert basis $H = \{h_1, \dots, h_N\}$ for the cone C, I_C is simply the kernel of the polynomial map $\phi : k[x_1, x_2, \dots, x_N] \longrightarrow k[y_1, y_2, \dots y_n]$, where $\phi(x_i) = y^{h_i}$ and for $h_i = (a_1, a_2, \dots, a_n)$ we set $y^{h_i} = y_1^{a_1} y_2^{a_2} \dots y_n^{a_n}$. There are standard techniques for computing this kind of kernel. It is in fact a toric ideal computation, see for example Chapter 7.

Example 8.2.1 cont. Let us illustrate this algebraic construction for magic 3×3 squares, where x_5 corresponds to the matrix with all entries one, at the bottom of Figure 8.3, and the other 4 variables x_1, x_2, x_3, x_4 correspond to the magic squares on top of Figure 8.3, as they appear from left to right. The ideal I_C is the toric ideal defined by the matrix whose columns are the 5 Hilbert basis elements:

$$\begin{pmatrix} 1 & 2 & 0 & 1 & 1 \\ 0 & 0 & 2 & 2 & 1 \\ 2 & 1 & 1 & 0 & 1 \\ 2 & 0 & 2 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 2 & 0 & 2 & 1 \\ 0 & 1 & 1 & 2 & 1 \\ 2 & 2 & 0 & 0 & 1 \\ 1 & 0 & 2 & 1 & 1 \\ 3 & 3 & 3 & 3 & 3 \end{pmatrix}.$$

Using `4ti2`, we see that I_C is generated by the two relations $x_5^2 - x_1x_4, x_2x_3 - x_1x_4$. The first relation means, for example, that the sum of magic square 1 with magic square 4 is the same as twice the magic square 5.

In order to compute the two relations $x_5^2 - x_1x_4, x_2x_3 - x_1x_4$ with `4ti2`, store the above matrix into a file, say again "3x3":

```
5 10
1 2 0 1 1
0 0 2 2 1
2 1 1 0 1
2 0 2 0 1
1 1 1 1 1
0 2 0 2 1
0 1 1 2 1
2 2 0 0 1
1 0 2 1 1
3 3 3 3 3
```

and then type "groebner 3x3" which creates the file "3x3.gro":

```
2 5
-1 0 0 -1 2
-1 1 1 -1 0
```

containing the desired two vectors that encode the two binomials stated above. □

It is important to observe that we can give a natural grading to R_C. A lattice point in C can be thought of as a monomial in the ring and its degree will be t, the dilation factor.

Example 8.2.1 cont. All elements of the Hilbert basis of magic 3×3 squares have degree 3. □

Once we have a graded k-algebra we can talk about its decomposition into the direct sum of its graded components $R_C = \bigoplus R_C(i)$, where each $R_C(i)$ collects all elements of degree i and it is a

k-vector space (where $R_C(0) = k$). The function $H(R_C, i) = \dim_k(R_C(i))$ is the *Hilbert function* of R_C. Similarly one can construct the *Hilbert-Poincaré series* of R_C, $H_{R_C}(t) = \sum_{i=0}^{\infty} H(R_C, i)t^i$.

Lemma 8.2.2 *Let C be a pointed rational cone, with Hilbert basis $H = \{h_1, \ldots, h_N\}$. Let the degree of a variable x_i in the ring $k[x_1, \ldots, x_N]$ be given by the t-component of h_i. Let R_C be the (graded) semigroup ring obtained from the minimal Hilbert basis of a cone C. Then the number of distinct lattice points in C with last component s equals the value of the Hilbert function $H(R_C, s)$.*

Proof. By the definition of a Hilbert basis we have that every lattice point in the cone C can be written as a linear integer combination of the elements of the Hilbert basis. The elements of $H = \{h_1, h_2, \ldots, h_N\}$ are not affinely independent therefore there are different combinations that produce the same lattice point. We have some dependencies of the form $\sum a_i h_i = \sum a_j h_j$ where the sums run over some subsets of $\{1, \ldots, N\}$. We consider such identities as giving a single lattice point. The dependencies are precisely the elements of the toric ideal I_C, that give $R_C = k[x_1, x_2, \ldots, x_N]/I_C$. Every such dependence is a linear combination of generators of any Gröbner basis of the ideal I_C. Thus, if we encode a lattice point $X \in C$ as a monomial in variables $x_1, \ldots, x_N$ whose exponents are the coefficients of the corresponding Hilbert basis elements that add to X, we are counting the equivalence classes modulo I_C. These are called *standard monomials.* Finally, it is known that the number of standard monomials of graded degree i equals the dimension of $R_C(i)$ as a k-vector space [36, Chapter 9]. □

It is known that the Hilbert-Poincaré series of R_C can be expressed as a rational function of the form

$$H_{R_C}(t) = \frac{p(t)}{\prod_{i=1}^{r}(1 - t^{\delta_i})},$$

where δ_i can be read from the rays of the cone C; they correspond to the denominators of the vertices of the polytope of stochastic arrays whose dilations give the cone C (see Theorem 4.6.25 [110] and Theorem 2.3 in [111]). To compute the Hilbert-Poincaré series we relied on the computer algebra package `CoCoA` [30], that has implementations for different algorithms of Hilbert series computations [20]. The basic idea comes from the theory of Gröbner bases (see [36, Chapter 9]). It is known that the initial ideal of I_C with respect to any monomial order gives a monomial ideal J and the Hilbert functions of $k[x_1, x_2, \ldots, x_N]/I_C$ and $k[x_1, x_2, \ldots, x_N]/J$ are equal. Computing the Hilbert function of the monomial ideal J is a combinatorial problem which can be solved by an inclusion-exclusion type procedure [20] that eliminates variables at each iteration.

Example 8.2.1 cont. The `CoCoA` commands that compute the Hilbert-Poincaré series for the ideal I_C are

```
L:=[3,3,3,3,3];
Use S::=Q[x[1..5]],Weights(L);
I:=Ideal(x[1]*x[4]-x[5]^2,x[1]*x[4]-x[2]*x[3]);
Poincare(S/I);
---  Non-simplified HilbertPoincare' Series  ---
(1 - 2x[1]^6 + x[1]^12) / ( (1-x[1]^3) (1-x[1]^3) (1-x[1]^3) (1-x[1]^3) (1-x[1]^3) )
```

In simplified form this reads

$$\sum_{t=0}^{\infty} i_P(t) z^t = \frac{z^6 + 2z^3 + 1}{(1 - z^3)^3}.$$

Note that in order to carry out the computation it is necessary to specify weights for the variables. In our case the weights are simply the magic sums of the arrays.

It is known that from a rational representation like this one can directly recover a quasi-polynomial (see [110, Chapter 4]).

$$i_{P_{3\times3}}(t) = \begin{cases} \frac{2}{9}t^2 + \frac{2}{3}t + 1 & \text{if } 3|t, \\ 0 & \text{otherwise.} \end{cases}$$

□

It remains to clarify how we can get an explicit formula for the quasi-polynomial $i_P(t)$ from $\sum_{t=0}^{\infty} i_P(t)z^t$. Here, the following lemma due to Ehrhart tells us bounds both for the degree and for the period of the quasi-polynomial $i_P(t)$.

Lemma 8.2.3 ([54]) *For a rational k-polytope P embedded in $\mathbb{R}^n$, the counting function $i_P(t)$ is a quasi-polynomial in t whose degree equals k and whose period is less than or equal to the least common multiple of the denominators of the vertices of P.*

With this information, we only have to find the Taylor series expansion of $\sum_{t=0}^{\infty} i_P(t)z^t$ at $t = 0$ to a sufficiently high degree and then interpolate $i_P(t)$ from the coefficients (function values of $i_P(t)$).

For example, for magic 3×3 squares of magic sum 1, the vertices of the two-dimensional polytope $P_{3\times3}$ are obtained by dividing the first 4 magic squares in Figure 8.3 by 3. Thus, the degree of $i_{P_{3\times3}}(t)$ is two and the periodicity of the quasi-polynomial should be a divisor of 3. In this case, the periodicity of the quasi-polynomial $i_{P_{3\times3}}(t)$ equals indeed the given bound of 3. Although in all our computations the period of the quasi-polynomial turned out to be equal to the least common multiple of the denominators of the vertices of P, this is not true in general (see Example 4.6.27 in [110]).

8.3 Families of magic arrays.

Let us now apply the above method to derive counting formulas for some families of magic and semi-magic arrays.

8.3.1 Magic 4×4 squares

In [18], the authors presented a counting formula for magic 4×4 squares, whose constant term was corrected in [3] via the approach presented above. In this example, the Hilbert basis consists of 20 elements, 8 of magic sum 1 and 12 of magic sum 2. They lead to the following counting formula

$$\begin{aligned}\sum_{t=0}^{\infty} i_{P_{4\times4}}(t)z^t = &= \frac{z^8+4z^7+18z^6+36z^5+50z^4+36z^3+18z^2+4z+1}{(1-z)^4(1-z^2)^4}\\ &= 1+8z+48z^2+200z^3+675z^4+1904z^5+4736z^6+10608z^7+\ldots,\end{aligned}$$

or more explicitly

$$i_{P_{4\times4}}(t) = \begin{cases} \frac{1}{480}t^7+\frac{7}{240}t^6+\frac{89}{480}t^5+\frac{11}{16}t^4+\frac{49}{30}t^3+\frac{38}{15}t^2+\frac{71}{30}t+1 & \text{if } 2|t,\\ \\ \frac{1}{480}t^7+\frac{7}{240}t^6+\frac{89}{480}t^5+\frac{11}{16}t^4+\frac{779}{480}t^3+\frac{593}{240}t^2+\frac{1051}{480}t+\frac{13}{16} & \text{otherwise.}\end{cases}$$

8.3.2 Magic 5×5 squares

The Hilbert basis for magic 5×5 squares consists of 4828 elements of magic sums ranging from 1 up to 9. The subsequent toric ideal computation in 4828 variables, however, could not be done by CoCoA, Macaulay2, or `4ti2`. Thus, although we were able to compute the Hilbert basis within a few seconds of computation time, we could not produce the counting formula $i_P(t)$ for magic 5×5 squares via this approach.

8.3.3 Pandiagonal magic 4×4 squares

Let $PP_{4\times4}$ denote the polytope of pandiagonal magic 4×4 squares of magic sum 1. The Hilbert basis for pandiagonal magic 4×4 squares consists of 8 elements, all of magic sum 2. From them we obtain the counting formula

$$\begin{aligned}\sum_{t=0}^{\infty} i_{PP_{4\times4}}(t)z^t = &= \frac{z^6+3z^4+3z^2+1}{(1-z^2)^5}\\ &= 1+8z^2+33z^4+96z^6+225z^8+456z^{10}+\ldots,\end{aligned}$$

or more explicitly

$$i_{PP_{4\times4}}(t) = \begin{cases} \frac{1}{48}(t^2+4t+12)(t+2)^2 & \text{if } 2|t,\\ \\ 0 & \text{otherwise.}\end{cases}$$

8.3.4 Pandiagonal magic 5×5 squares

Let $PP_{5\times5}$ denote the polytope of pandiagonal magic 5×5 squares of magic sum 1. The Hilbert basis for pandiagonal magic 5×5 squares consists of 10 elements, all of magic sum 1. From them we obtain

$$\begin{aligned}\sum_{t=0}^{\infty} i_{PP_{5\times5}}(t)z^t = &= \frac{z^4+z^3+z^2++z+1}{(1-z)^9}\\ &= 1+10z+55z^2+220z^3+715z^4+2001z^5+4995z^6+11385z^7+\ldots,\end{aligned}$$

or more explicitly

$$i_{PP_{5\times5}}(t) = \frac{1}{8064}(t+4)(t+3)(t+2)(t+1)(t^2+5t+8)(t^2+5t+42).$$

8.3.5 Magic $3 \times 3 \times 3$ hypercubes

The Hilbert basis for magic $3 \times 3 \times 3$ hypercubes consists of 19 elements all of magic sum 3. From them we obtain

$$\begin{aligned}\sum_{t=0}^{\infty} i_{P_{3\times3\times3}}(t)z^t &= \frac{z^{12}+14z^9+36z^6+14z^3+1}{(1-z^3)^5} \\ &= 1+19z^3+121z^6+439z^9+1171z^{12}+2581z^{15}+\dots,\end{aligned}$$

or more explicitly

$$i_{P_{3\times3\times3}}(t) = \begin{cases} \frac{11}{324}t^4+\frac{11}{54}t^3+\frac{25}{36}t^2+\frac{7}{6}t+1 & \text{if } 3|t, \\ 0 & \text{otherwise.}\end{cases}$$

8.3.6 Semi-magic $3 \times 3 \times 3$ hypercubes

The Hilbert basis for semi-magic $3 \times 3 \times 3$ hypercubes consists of 66 elements, 12 of magic sum 1 and 54 of magic sum 2. Thus, the polytope of stochastic semi-magic $3 \times 3 \times 3$ cubes is not equal to the convex hull of integral semi-magic cubes. This follows because the 54 elements of degree two in the Hilbert basis, when appropriately normalized, give rational stochastic matrices that are all vertices. In other words, the Birkhoff-von Neumann Theorem [106, page 108] about stochastic semi-magic matrices is false for $3 \times 3 \times 3$ stochastic semi-magic cubes.

For semi-magic $3 \times 3 \times 3$ cubes we have the counting formula

$$\begin{aligned}\sum_{t=0}^{\infty} i_{P_{3\times3\times3s}}(t)z^t &= \frac{z^8+5z^7+67z^6+130z^5+242z^4+130z^3+67z^2+5z+1}{(1-z)^9(1+z)^2} \\ &= 1+12z+132z^2+847z^3+3921z^4+14286z^5+43687z^6+116757z^7+\dots\end{aligned}$$

In other words,

$$i_{P_{3\times3\times3s}}(t) = \begin{cases} \frac{9}{2240}t^8+\frac{27}{560}t^7+\frac{87}{320}t^6+\frac{297}{320}t^5+\frac{1341}{640}t^4+\frac{513}{160}t^3+\frac{3653}{1120}t^2+\frac{627}{280}t+1 & \text{if } 2|t, \\ \frac{9}{2240}t^8+\frac{27}{560}t^7+\frac{87}{320}t^6+\frac{297}{320}t^5+\frac{1341}{640}t^4+\frac{513}{160}t^3+\frac{3653}{1120}t^2+\frac{4071}{2240}t+\frac{47}{128} & \text{otherwise.}\end{cases}$$

Chapter 9

Counting via Barvinok's algorithm

9.1 Introduction

In the 1980's, H. Lenstra created an algorithm to *detect* integer points in polyhedra, based on the LLL-algorithm and the idea of short vectors [67, 93]. As a consequence, solving integer programming problems with a fixed number of variables can be done in time polynomial in the size of the input.

In the 1990's, based on work by the geometers Brion, Khovanski, Lawrence, and Pukhlikov, Barvinok created an algorithm to *count* integer points inside polyhedra that runs in polynomial time for fixed dimension (see [13, 14] and the references within). Shortly after Barvinok's breakthrough, Dyer and Kannan [52] modified the original algorithm of Barvinok, which originally relied on Lenstra's result, giving a new proof that integer programming problems with a fixed number of variables can be solved in polynomial time. In this chapter, extending the work initiated in [48], we describe the first ever implementation of Barvinok's algorithm valid for arbitrary rational polytopes; the program `LattE` [41].

In this chapter, we go through the steps of Barvinok's algorithm, showing how they are implemented in `LattE`. Barvinok's algorithm relies on two important new ideas: the use of rational functions as efficient data structures and the signed decompositions of cones into unimodular cones.

Given a polyhedron $P = \{x \in \mathbb{R}^d : Ax \leq b\}$, we encode the lattice points of P in the generating function $f(P; z) = \sum_{a \in P \cap \mathbb{Z}^d} z^a$, where $z^a = z_1^{a_1} z_2^{a_2} \ldots z_d^{a_d}$. Note that when P is a polytope (i.e. a bounded polyhedron), the monomials of $f(P; z)$ are in bijection with the lattice points and thus $f(P; z)$ is a (Laurent) polynomial. Counting the lattice points in P is equivalent to evaluating the expression $f(P;)$ at $z = \mathbf{1}$, the vector with all entries 1.

Example 9.1.1 Consider the interval $P = [0, 1000] \subseteq \mathbb{R}$. Its lattice points $0, 1, \ldots, 1000$ can be encoded into the rather long polynomial $f(P; z) = 1 + z_1 + z_1^2 + \ldots + z_1^{1000}$. $\square$

In this chapter, it is our goal to rewrite $f(P; z)$ as a "short" (polynomial size) sum of rational functions in z from which we can solve feasibility, counting, or even optimization questions, about the lattice points in P.

Example 9.1.1 cont. For the interval above, we obtain the following short rational expression:

$$f(P;z) = \frac{1}{(1-z_1^{1001})} - \frac{z_1}{(1-z_1^{1001})}$$

Indeed, by adding the rational functions we recover the polynomial $1 + z_1 + z_1^2 + \ldots + z_1^{1000}$. □

9.2 Brion's formula

One crucial component of Barvinok's algorithm is the ability to distribute the computation on the vertices of the polytope. Let v be a vertex of P. Then, the *supporting cone* $K(P,v)$ of P at v is $K(P,v) = v + \{u \in \mathbb{R}^d : v + \delta u \in P$ for all sufficiently small $\delta > 0\}$. Then the seminal theorem of Brion [23] states:

Theorem 9.2.1 *[23] Let P be a rational polyhedron and let $V(P)$ be the vertex set of P. Then,*

$$f(P;z) = \sum_{v \in V(P)} f(K(P,v);z).$$

Thus, once we know how to encode the lattice points in the shifted cones $K(P,v)$ efficiently, we are done.

Example 9.2.2 Consider the integral quadrilateral shown in Figure 9.1. The vertices of this quadrilateral are $V_1 = (0,0)$, $V_2 = (5,0)$, $V_3 = (4,2)$, and $V_4 = (0,2)$.

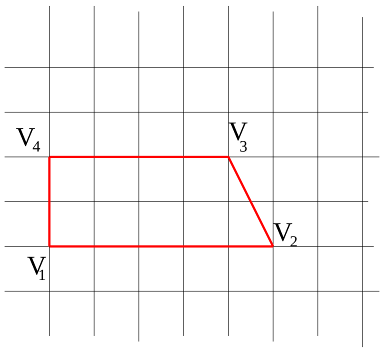


Figure 9.1: A quadrilateral.

We obtain the generation function

$$f(P;z) = z_1^5 + z_1^4 z_2^2 + z_1^4 z_2 + z_1^4 + z_1^3 z_2^2 + z_1^3 z_2 + z_1^3 + z_1^2 z_2^2 + z_1^2 z_2 + z_1^2 + z_1 z_2^2 + z_1 z_2 + z_1 + z_2^2 + z_2 + 1.$$

In terms of supporting cones, this expression can be rewritten as the sum of four rational generation functions (one for each supporting cone) whose formulas are

$$\begin{aligned}
f(K(P,V_1);z) &= \frac{1}{(1-z_1)\,(1-z_2)},\\
f(K(P,V_2);z) &= \frac{(z_1^5+z_1^4z_2)}{(1-z_1^{-1})(1-z_2^2z_1^{-1})},\\
f(K(P,V_3);z) &= \frac{(z_1^4z_2+z_1^4z_2^2)}{(1-z_1^{-1})(1-z_1z_2^{-2})},\\
f(K(P,V_4);z) &= \frac{z_2^2}{(1-z_2^{-1})(1-z_1)}.
\end{aligned}$$

Indeed, the result of adding the rational functions is equal to the polynomial

$$z_1^5+z_1^4z_2^2+z_1^4z_2+z_1^4+z_1^3z_2^2+z_1^3z_2+z_1^3+z_1^2z_2^2+z_1^2z_2+z_1^2+z_1z_2^2+z_1z_2+z_1+z_2^2+z_2+1.$$

□

9.3 Reducing the problem to simplicial cones

In order to use Brion's theorem for counting lattice points in convex polyhedra, we need to know how to compute the rational generating function of the supporting cone $K(P,v)=v+K_v$, where K_v is a rational pointed polyhedral cone attached at v. If the cone K_v is simplicial, this generating function is a rational function whose numerator and denominator have a well-understood geometric meaning (see in [110, Chapter 4], and in [113], Corollary 4.6.8, for a clear explanation).

Let us derive such a formula for $f(K(P,v);z)$ when K_v is simplicial: Let $\{u_1,u_2,\ldots,u_k\}$ be a set of $k\le d$ linearly independent integral vectors of $\mathbb{R}^d$ and let

$$K_v=\mathrm{cone}(u_1,u_2,\ldots,u_k)=\{\lambda_1u_1+\lambda_2u_2+\ldots+\lambda_ku_k:\lambda_i\ge 0,i=1,2,\ldots,k\}.$$

Then we denote by

$$F_v=\{v+\lambda_1u_1+\lambda_2u_2+\ldots+\lambda_ku_k,0\le\lambda_i<1,i=1,2,\ldots,k\}$$

the fundamental parallelepiped of K_v at v. With these definitions, it is well-known [110] that the generating function for the lattice points in K equals

$$\sum_{\beta\in K\cap Z^d} z^\beta=\left(\sum_{\tau\in F_v\cap Z^d} z^\tau\right)\prod_{i=1}^{k}\frac{1}{1-z^{u_i}}. \tag{9.3.1}$$

Thus, to derive a formula for arbitrary pointed cones K_v one could decompose K_v into simplicial cones, via a triangulation, and then apply Formula 9.3.1 above and the inclusion-exclusion principle in [113], Proposition 1.2 to compute $f(K(P,v);z)$.

Example 9.2.2 cont. The four pieces

$$\begin{aligned}
f(K(P,V_1);z) &= \frac{1}{(1-z_1)(1-z_2)},\\
f(K(P,V_2);z) &= \frac{(z_1^5+z_1^4z_2)}{(1-z_1^{-1})(1-z_2^2z_1^{-1})},\\
f(K(P,V_3);z) &= \frac{(z_1^4z_2+z_1^4z_2^2)}{(1-z_1^{-1})(1-z_1z_2^{-2})},\\
f(K(P,V_4);z) &= \frac{z_2^2}{(1-z_2^{-1})(1-z_1)}.
\end{aligned}$$

for the quadrilateral were constructed as described in this section. Note that since all vertices v are integral, we simply have $f(K(P,v);z) = z^v f(K_v;z)$. $\square$

Implementational details. To decompose a cone into simplicial cones the first step is to do a triangulation. (Let us remind the reader here again that a *triangulation* of a cone C in dimension d is a collection of d-dimensional simplicial cones such that their union is C, their interiors are disjoint, and any pair of them intersect in a (possibly empty) common face.)

When the dimension is fixed, there are efficient algorithms to compute such a triangulation (see [10, 92] for details). In `LattE` we use the well-known Delaunay triangulation which we compute via a convex hull calculation. The idea is to "lift" the rays of the cone into a higher dimensional paraboloid by adding a new coordinate which is the sum of the squares of the other coordinates, take the lower convex hull of the lifted points, and then "project" back those simplicial facets. We use Fukuda's implementation in `cdd` [58] of this lift-and-project algorithm. Note that this is not the only choice of triangulation, and definitely not the smallest one.

In principle, one could at this point list the lattice points of the fundamental parallelepiped F_v at v, for example, using a fast Hilbert basis code such as `4ti2` [73] or `NORMALIZ` [24], and then use formula (9.3.1) for a general simplicial cone K_v. Theoretically, this is a bad idea because the number of lattice points in the parallelepiped F_v at v is exponentially large already for fixed dimension. In practice, however, this can often be done and is useful in some situations. $\square$

9.4 Barvinok's idea of signed cone decompositions

Formula 9.3.1 together with Brion's Theorem 9.2.1 already allow us to rewrite the generating function $f(P;z) = \sum_{a\in P\cap\mathbb{Z}^d} z^a$ into a rational generating function

$$f(P;z) = \sum_{i\in I} E_i \frac{z^{u_i}}{\prod_{j=1}^{d}(1-z^{v_{ij}})}.$$

However, as there may be exponentially many lattice points in a fundamental parallelepiped, even when we fix the dimension, this rational generating function $f(P;z)$ is not a representation of the lattice points of P of polynomial size in the encoding length of $P = \{x \in \mathbb{R}^d : Ax \leq b\}$, that is, in the encoding lengths of A and b.

Instead, Barvinok's idea is that it is more efficient to further decompose each simplicial cone into simplicial *unimodular* cones. A *unimodular* cone is a simplicial cone with generators $\{u_1, \ldots, u_k\} \subseteq \mathbb{Z}^d$, $k \leq d$, that form a lattice basis for the lattice $\mathbb{R}\{u_1, \ldots, u_k\} \cap \mathbb{Z}^d$. Note that in this case the numerator of Formula 9.3.1 has a single monomial, since the parallelepiped F_v has only one lattice point.

Unfortunately, not every (simplicial) cone can be written as the sum of (simplicial) unimodular cones. This problem was resolved by Barvinok. Instead of decomposing simplicial cones into sums of simplicial unimodular cones, Barvinok suggested to decompose each simplicial cone into a *signed* sum of simplicial unimodular cones. This, together with Formula 9.3.1 and with Brion's Theorem 9.2.1, allows us to write $f(P; z)$ as a "short" rational generating function of polynomial size representation in the encoding length of P.

To be more formal, for a set $A \subset \mathbb{R}^d$, the indicator function $[A] : \mathbb{R}^d \to \mathbb{R}$ of A is defined as

$$[A](x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases}$$

We want to express the indicator function of a simplicial cone as an integer linear combination of the indicator functions of unimodular simplicial cones. There is a nice valuation from the algebra of indicator functions of polyhedra to the field of rational functions [14], and many of its properties can be used in the calculation. For example, the valuation is zero when the polyhedron contains a line.

Theorem 9.4.1 ([14], Theorem 3.1) *There is a valuation f from the algebra of indicator functions of rational polyhedra into the field of multivariate rational functions such that for any polyhedron P, $f([P]) = \sum_{\alpha \in P \cap \mathbb{Z}^d} x^\alpha$.*

Therefore once we have a unimodular cone decomposition, the rational generating function of the original cone is a signed sum of "unimodular" rational functions. Next we focus on how to decompose a simplicial cone into unimodular cones.

9.5 Performing the signed cone decomposition

Let $u_1, u_2, \ldots, u_d$ be linearly independent integral vectors which generate a simplicial cone K. We denote the *index* of K by $\operatorname{ind}(K)$ which tells us how far K is from being unimodular. That is, $\operatorname{ind}(K) = |\det(u_1|u_2|\ldots|u_d)|$ which is the volume of the parallelepiped spanned by $u_1, u_2, \ldots, u_d$. It is also equal to the number of lattice points inside the half-open parallelepiped spanned by $u_1, u_2, \ldots, u_d$. K is unimodular if and only if the index of K is 1. Now we discuss how we implemented the following key result of Barvinok:

Theorem 9.5.1 ([14],Theorem 4.2) *Fix the dimension d. Then, there exists a polynomial time algorithm with a given rational polyhedral cone $K \subset \mathbb{R}^d$, which computes unimodular cones K_i,*

$i \in I = \{1, 2, \ldots, l\}$, *and numbers* $\epsilon_i \in \{-1, 1\}$ *such that*

$$[K] = \sum_{i \in I} \epsilon_i [K_i].$$

Let K be a rational pointed simplicial cone. Consider the closed parallelepiped

$$\Gamma = \{\alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_d u_d : |\alpha_j| \leq (\mathrm{ind}(K))^{-\frac{1}{d}}, j = 1, 2, \ldots, d\}.$$

Note that this parallelepiped Γ is centrally symmetric and one can show that the volume of Γ is 2^d. Minkowski's First Theorem [106] guarantees that because $\Gamma \subset \mathbb{R}^d$ is a centrally symmetric convex body with volume $\geq 2^d$, there exists a non-zero lattice point w inside of Γ. We will use w to build the decomposition.

However, we need to find such a vector w explicitly. We take essentially the approach suggested by [52]. We require a subroutine that computes the shortest vector in a lattice. For fixed dimension this can be done in polynomial time using lattice basis reduction (this follows trivially from [106], Corollary 6.4b, page 72). It is worth observing that when the dimension is not fixed the problem becomes NP-hard [5]. We use the basis reduction algorithm of Lenstra, Lenstra, and Lovász [67, 106] to find a short vector. Given A, an integral $d \times d$ matrix whose columns generate a lattice, LLL's algorithm outputs A', a new $d \times d$ matrix, spanning the same lattice generated by A. The column vectors of A', $u'_1, u'_2, ..., u'_d$, are short and nearly orthogonal to each other, and each u'_i is an approximation of the shortest vector in the lattice, in terms of Euclidean length. It is well-known [106] that there exists a unique unimodular matrix U such that $AU = A'$.

The method proposed in [52] to find w is the following: Let $A = (u_1|u_2| \ldots |u_d)$, where the u_i are the rays of the simplicial cone we wish to decompose. Compute the reduced basis A' of A^{-1} using the LLL algorithm. Dyer and Kannan observed that we can find the smallest vector with respect to the l^∞ norm by searching over all linear integral combinations of the column vectors of A' with small coefficients. We call this search the *enumeration step*. Let λ be the smallest vector in the lattice spanned by A' with respect to the l^∞ norm. We know that there exists a unique unimodular matrix U such that $A' = A^{-1}U$. Minkowski's theorem for the l^∞ norm implies that for the non-singular matrix A', there exists a non-zero integral vector z such that $\lambda = \|A'z\|_\infty \leq |\det(A')|^{1/d}$, where $\|.\|$ is the infinity norm of the vector space $\mathbb{R}^d$, see statement 23 on page 81 in [106]. We can set

$$\begin{array}{rcccccc} \|\lambda\|_\infty & \leq & |\det(A')|^{1/d} & = & |\det(A^{-1}U)|^{1/d} & = & |\det(A^{-1})\det(U)|^{1/d} \\ & = & |\det(A^{-1})|^{1/d} & = & |\det(A)|^{-1/d} & = & |\mathrm{ind}(K)|^{-1/d}. \end{array}$$

Since A^{-1} and A' span the same lattice, there exists an integral vector $w \in \mathbb{R}^d$ such that $\lambda = A^{-1}w$. Then, we have

$$w = A\lambda.$$

Note that w is a non-zero integral vector which is a linear integer combination of the generators u_i of the cone K *with possibly negative coefficients*, and with coefficients at most $|\mathrm{ind}(K)|^{-1/d}$. Therefore, we have found a non-zero integral vector $w \in \Gamma$.

Implementational details. In `LattE`, we try to avoid the enumeration step because it is very costly. Instead, we choose λ to be the shortest of the columns in A'. This may not be the smallest vector, but for practical purposes, it often decreases $|\mathrm{ind}(K)|$ just like the shortest vector. Experimentally we have observed that we rarely use the enumeration step. $\square$

In the next step of the algorithm, for $i = 1, 2, \ldots, d$, we set

$$K_i = \mathrm{cone}\{u_1, u_2, \ldots, u_{i-1}, w, u_{i+1}, \ldots, u_d\}.$$

Now, we have to show that for each i, $\mathrm{ind}(K_i)$ is smaller than $\mathrm{ind}(K)$. Let $w = \sum_{i=1}^{d} \alpha_i u_i$. Then, since $|\alpha_i| \leq \mathrm{ind}(K)^{-\frac{1}{d}}$, we have

$$\begin{aligned} \mathrm{ind}(K_i) &= |\det(u_1|u_2|\ldots|u_{i-1}|w|u_{i+1}|\ldots|u_d)| \\ &= |\alpha_i||\det(u_1|u_2|\ldots|u_{i-1}|u_i|u_{i+1}|\ldots|u_d)| \\ &= |\alpha_i|\mathrm{ind}(K) \leq (\mathrm{ind}(K))^{\frac{d-1}{d}}. \end{aligned}$$

There is one more technical condition that w needs to satisfy. This is that w and $u_1, \ldots, u_d$ belong to an open half-space [13, Lemma 5.2]. This is easy to achieve as either the w we found or $-w$ satisfies this condition. We can now decompose the original cone K into cones $K_1, \ldots, K_d$ of smaller index, $[K] = \sum \pm[K_i]$. This sum of indicator functions carries signs which depend on the position of w with respect to the interior or exterior of K. We iterate this process until each K_i becomes a unimodular cone for $i = 1, 2, \ldots, d$.

Implementational details. For implementing Barvinok's decomposition of cones, we use the package `NTL` [109] to compute the reduced basis of a cone and to compute with matrices and determinants. All our calculations were done in exact long integer arithmetic using the routines integrated in `NTL`. $\square$

9.6 Pseudo-code of signed cone decomposition algorithm

Algorithm 9.6.1 (Barvinok's Decomposition of a Simplicial Cone)

Input: A simplicial cone $K = \mathrm{cone}\{u_1, u_2, \ldots, u_d\}$ given by its generators.

Output: A list of unimodular cones and numbers ϵ_i as in Theorem 9.5.1.

```
Set two queues Uni := $\emptyset$ and NonUni := $\emptyset$.
if $K$ is unimodular
  then Uni := Uni$\{K\}$
  else NonUni := NonUni $\cup\{K\}$
while NonUni $\neq \emptyset$ do
  Choose a cone $K \in$ NonUni.
  NonUni := NonUni $\setminus\{K\}$
  Set $A := (u_1, \ldots, u_d)$ to be a matrix whose columns are the rays of $K$.
  Compute a smallest vector $\lambda$ w.r.t. $l^\infty$ in the lattice spanned by the column vectors of $A^{-1}$.
  Find a non-zero integral vector $w$ such that $\lambda = A^{-1}w$.
  **if** vectors $w, u_1, \dots, u_d$ lie in an open half-space
    **then** $w := w$
    **else** $w := -w$
  **for** $i = 1, 2, \dots, d$ **do**
    $K_i := \text{cone}\{u_1, \dots, u_{i-1}, w, u_{i+1}, \dots, u_d\}$
    $A_i := (u_1, \dots, u_{i-1}, w, u_{i+1}, \dots, u_d)$
  **for** $i = 1, 2, \dots, d$ **do**
    **if** $\det(A_i)$ and $\det(A)$ have the same sign
      **then** $\epsilon_{K_i} := \epsilon_K$
      **else** $\epsilon_{K_i} := -\epsilon_K$
  **for** $i = 1, 2, \dots, d$ **do**
    **if** $K_i$ is unimodular
      **then** $\text{Uni} := \text{Uni} \cup \{K_i\}$
      **else** $\text{NonUni} := \text{NonUni} \cup \{K_i\}$
**return** Uni
```

9.7 Example of a signed cone decomposition

Here is an example of how we carry out the decomposition. Let K be a cone generated by $(2,7)^T$ and $(1,0)^T$. Let

$$A = \begin{pmatrix} 2 & 1 \\ 7 & 0 \end{pmatrix}.$$

Then, we have $\det(A) = -7$ and

$$A^{-1} = \begin{pmatrix} 0 & 1/7 \\ 1 & -2/7 \end{pmatrix}.$$

The reduced basis A' of A^{-1} and the unimodular matrix U for the transformation from A^{-1} to A' are:

$$A' = \begin{pmatrix} 1/7 & 3/7 \\ -2/7 & 1/7 \end{pmatrix} \quad \text{and} \quad U = \begin{pmatrix} 0 & 1 \\ 1 & 3 \end{pmatrix}.$$

By enumerating the column vectors, we can verify that $(-2/7, 1/7)^\intercal$ is the smallest vector with respect to l^∞ in the lattice generated by the column vectors of A^{-1}. So, we have $w = (1,0)^\intercal$. Then, we have two cones:

$$\text{cone}\left(\begin{pmatrix} 2 \\ 0 \end{pmatrix}, \begin{pmatrix} 7 \\ 1 \end{pmatrix}\right) \text{ and } \text{cone}\left(\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right).$$

The first cone has determinant 2. As $\det(A) < 0$, we assign $\epsilon = -1$ to the first cone. Since the first cone is not unimodular, we have

$$\text{NonUni} = \text{NonUni} \cup \left\{ -\text{cone}\left(\begin{pmatrix} 2 \\ 0 \end{pmatrix}, \begin{pmatrix} 7 \\ 1 \end{pmatrix}\right)\right\}.$$

The second cone is unimodular of with determinant -1 which is the same sign as the determinant of A. Thus, we assign to it $\epsilon = 1$ and put

$$\mathrm{Uni} = \mathrm{Uni} \cup \left\{ + \mathrm{cone}\left(\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right) \right\}.$$

Next we choose

$$\mathrm{cone}\left(\begin{pmatrix} 2 \\ 0 \end{pmatrix}, \begin{pmatrix} 7 \\ 1 \end{pmatrix} \right) \in \mathrm{NonUni}$$

and set

$$A = \begin{pmatrix} 2 & 0 \\ 7 & 1 \end{pmatrix}.$$

Then, we have $\det(A) = 2$ and

$$A^{-1} = \begin{pmatrix} 1/2 & 0 \\ -7/2 & 1 \end{pmatrix}, A' = \begin{pmatrix} 1/2 & 1/2 \\ -1/2 & 1/2 \end{pmatrix}, \quad \text{and} \quad U = \begin{pmatrix} 1 & 1 \\ 3 & 4 \end{pmatrix}.$$

Since $\lambda = (1/2, -1/2)^{\intercal}$ is the smallest vector with respect to l^{∞}, we have $w = (1, 3)^{\intercal}$. So, we get two cones:

$$\mathrm{cone}\left(\begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} 7 \\ 3 \end{pmatrix} \right) \text{ and } \mathrm{cone}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 3 \\ 1 \end{pmatrix} \right).$$

The first cone has a negative determinant which is not the same sign as the determinant of its parent cone K. Since $\epsilon_K = -1$, we assign to the first cone $\epsilon = -\epsilon_K = 1$. The second cone has a positive determinant, so we assign to it $\epsilon = \epsilon_K = -1$. Since both cones are unimodular, we put them into Uni. Since NonUni is empty, we end while loop and print all elements in Uni.

This gives a full decomposition:

$$\mathrm{cone}\left\{ \begin{pmatrix} 2 \\ 7 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\} = \mathrm{cone}\left\{ \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\} \oplus \mathrm{cone}\left\{ \begin{pmatrix} 2 \\ 7 \end{pmatrix}, \begin{pmatrix} 1 \\ 3 \end{pmatrix} \right\} \ominus \mathrm{cone}\left\{ \begin{pmatrix} 1 \\ 3 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\},$$

see Figure 9.2. □

9.8 Brion's polarization trick

It is very important to remark that, in principle, one also needs to keep track of lower dimensional cones present in the decomposition for the purpose of writing the inclusion-exclusion formula of the generating function $f(K)$. For example in Figure 9.3 we have counted a ray twice, and thus it needs to be removed.

But this is actually not necessary, if we remember that the generating function of a polyhedron containing a line is 0 [14, Corollary 2.8]. This fact is crucial for *Brion's polarization trick* [14], Remark 4.3: Let K^* be the dual cone to K. Apply the iterative procedure above to K^* instead of K, ignoring the lower dimensional cones. This can be done because once we polarize the result back, the contribution of the lower dimensional cones is zero with respect to the valuation that assigns to an indicator function its generating function counting the lattice points [14, Corollary 2.8].

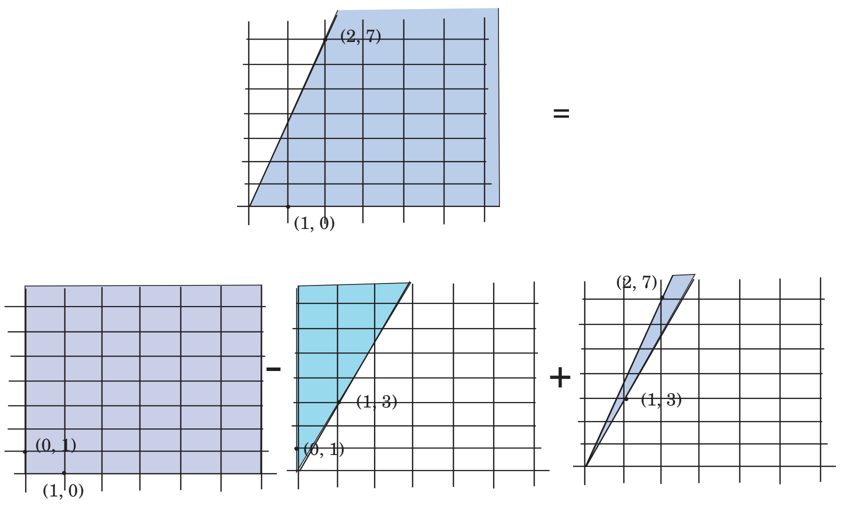


Figure 9.2: Example of Barvinok's decomposition.

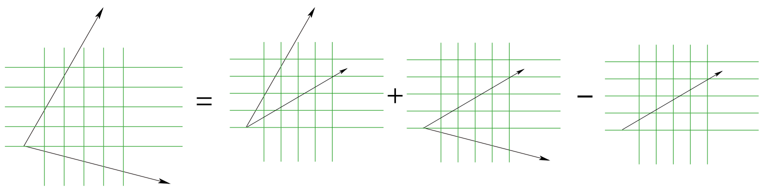


Figure 9.3: Contribution of lower dimensional cones.

9.9 From cones to rational functions

Once we have decomposed all cones into simplicial *unimodular* cones, it is easy to find the generating function of the ith cone K_i attached at the vertex v of the input polytope. We only have to employ a bit of linear algebra.

In the denominator there is a product of binomials of the form $(1 - z^{B_{ij}})$ where B_{ij} is the jth ray of the cone K_i. Thus the denominator is the polynomial $\prod(1 - z^{B_{ij}})$.

How about the numerator? The cone K_i is unimodular, thus it must have a single monomial z^{A_i}, corresponding to the unique lattice point inside the fundamental parallelepiped of K_i at v. If the vertex v has all integer coordinates then $A_i = v$, or else v can be written as a linear combination $\sum \lambda_j B_{ij}$ where all the λ_i are rational numbers and can be found by solving a system of equations (remember the B_{ij} form a vector space basis for $\mathbb{R}^d$). The unique lattice point inside the parallelepiped of the cone K_i at v is simply $\sum \lceil \lambda_j \rceil B_{ij}$ [14, Lemma 4.1].

9.10 Summary of overall algorithm

In the current implementation of `LattE`, we do the following:

1. Find the vertices of the polytope and their defining supporting cones.
2. Compute the polar cone to each of these cones.
3. Apply Barvinok's decomposition to each of the polars.
4. Polarize back all these unimodular cones to obtain a decomposition of the original supporting cones into full-dimensional unimodular cones.
5. Recover the generating function of each cone and, by Brion's theorem, of the whole polytope.

9.11 From rational functions to counts

Brion's theorem says the sum of the rational functions coming from the unimodular cones at the vertices is a polynomial with one monomial per lattice point inside the input polytope. One might think that to compute the number of lattice points inside of a given convex polyhedron, one could directly substitute the value of 1 for each of the variables. Unfortunately, $(1, 1, \ldots, 1)$ is a singularity of all the rational functions. Instead we discuss the method used in `LattE` to compute this value, which is different from that presented by Barvinok [14]. The typical generating function of lattice points inside a unimodular cone forms:

$$E[i]\frac{z^{A_i}}{\prod(1-z^{B_{ij}})},$$

where z^a is monomial in d variables, each A_i (cone vertex) and B_{ij} (a generator of cone i) are integer vectors of length d, i ranges over all cones given, j ranges over the generators of cone i, and $E[i]$ is 1 or -1. Adding these rational functions and simplifying would yield the polynomial function of the lattice point of the polytope. This is practically impossible as the number of monomials is too large. However, calculating the number of monomials in this polynomial is equivalent to evaluating the limit as z_i goes to 1 for all i. We begin by finding an integer vector λ and making the substitution $z_i \rightarrow t^{\lambda_i}$. This is with the intention of obtaining a univariate polynomial. To do this, λ must be picked such that there is no zero denominator in any cone expression, i.e. no dot product of λ with a B_{ij} can be zero. Barvinok showed that such a λ can be picked in polynomial time by choosing points on the moment curve. Unfortunately, this method yields large values in the entries of λ. Instead we try random vectors with small integer entries, allowing small increments if necessary, until we find λ. Since we are essentially trying to avoid a measure zero set, this process terminates very quickly in practice.

After substitution, we have expressions of the form $\pm t^{N_i}/\prod(1-t^{D_{ij}})$, where N_i and D_{ij} are integers. Notice that this substitution followed by summing these expressions yields the same polynomial as would result from first summing and then substituting. This follows from the fact that we can take Laurent series expansions, and the sum of Laurent series is equal to the Laurent series of the sum of the original expressions.

Also, note that we have the following identity:

$$\sum_{\alpha \in P \cap \mathbb{Z}^d} z^\alpha = \sum_{i=1}^{\#\text{ of cones}} E[i] \frac{z^{A_i}}{\prod(1 - z^{B_{ij}})}.$$

After substitution we have the following univariate (Laurent) polynomial such that:

$$\sum_{\alpha \in P \cap \mathbb{Z}^d} t^{\sum_{i=1}^d \lambda_i \alpha_i} = \sum_{i=1}^{\#\text{ of cones}} E[i] \frac{t^{N_i}}{\prod(1 - t^{D_{ij}})}.$$

With the purpose of avoiding large exponents in the numerators, we factor out a power of t, say t^c. Now we need to evaluate the sum of these expressions at $t = 1$, but we cannot evaluate these expressions directly at $t = 1$ because each has a pole there. Consider the Laurent expansion of the sum of these expressions about $t = 1$. The expansion must evaluate at $t = 1$ to the finite number $\sum_{\alpha \in P \cap \mathbb{Z}^d} 1$. It is a Taylor expansion and its value at $t = 1$ is simply the constant coefficient. If we expand each expression about $t = 1$ individually and add them up, it will yield the same result as adding the expressions and then expanding (again the sum of Laurent expansions is the Laurent expansion of the sum of the expressions). Thus, to obtain the constant coefficient of the sum, we add up the constant coefficients of the expansions about $t = 1$ of each summand. Computationally, this is accomplished by substituting $t = s+1$ and expanding about $s = 0$ via a polynomial division. Summing up the constant coefficients with proper accounting for $E[i]$ and proper decimal accuracy yields the desired result: the number of lattice points in the polytope. Before the substitution $t = s + 1$ we rewrite each rational function in the sum (recall t^c was factored to keep exponents small)

$$\sum E[i] \frac{t^{N_i - c}}{\prod(1 - t^{D_{ij}})} = \sum E'[i] \frac{t^{N'_i}}{\prod(t^{D'_{ij}} - 1)},$$

in such a way that $D'_{ij} > 0$ for all i, j. This requires that the powers of t at each numerator to be modified, and the sign $E[i]$ is also adjusted to $E'[i]$. Then the substitution $t = s + 1$ yields

$$\sum E'[i] \frac{(1 + s)^{N'_i}}{\prod((1 + s)^{D'_{ij}} - 1)},$$

where it is evident that, in each summand, the pole $s = 0$ has an order equal to the number of factors in the denominator. This is the same as the number of rays in the corresponding cone and we denote this number by d.

Thus the summand for cone i can be rewritten as $E'[i]s^{-d} P_i(s)/Q_i(s)$ where $P_i(s) = (1 + s)^{N_i}$ and $Q_i(s) = \prod^d((1 + s)^{D'_{ij}} - 1)/s)$. $P_i(s)/Q_i(s)$ is a Taylor polynomial whose s^d coefficient is the contribution we are looking for (after accounting for the sign $E'[i]$ of course). The coefficients of the quotient $P_i(s)/Q_i(s)$ can be obtained recursively as follows: Let $Q_i(s) = b_0 + b_1 s + b_2 s^2 + \ldots$ and $P_i(s) = a_0 + a_1 s + a_2 s^2 + \ldots$ and let $\frac{P_i(s)}{Q_i(s)} = c_0 + c_1 s + c_2 s^2 + \ldots$. Therefore, we want to obtain

c_d which is the coefficient of the constant term of P_i/Q_i. So, how do we obtain c_d from $Q_i(s)$ and $P_i(s)$? We obtain this by the following recurrence relation:

$$\begin{aligned} c_0 &= \frac{a_0}{b_0}, \\ c_k &= \frac{1}{b_0}(a_k - b_1 c_{k-1} - b_2 c_{k-2} - \ldots - b_k c_0) \text{ for } k = 1, 2, \ldots. \end{aligned}$$

In order to obtain c_d, only the coefficients $a_0, a_1, \ldots, a_d$ and $b_0, b_1, \ldots, b_d$ are required.

9.12 Example for limit computation

Let us consider three points in 2 dimensions such that $V_1 = (0,1)$, $V_2 = (1,0)$, and $V_3 = (0,0)$. Then, the convex hull of V_1, V_2, and V_3 is a triangle in 2 dimensions. We want to compute the number of lattice points by using the residue theorem. Let K_i be the vertex cone at V_i for $i = 1, 2, 3$. Then, we have the rational functions:

$$\begin{aligned} f(K_1; x, y) &= \frac{y}{(1-y^{-1})(1-xy^{-1})}, \\ f(K_2; x, y) &= \frac{x}{(1-x^{-1})(1-x^{-1}y)}, \\ f(K_3; x, y) &= \frac{1}{(1-x)(1-y)}. \end{aligned}$$

We choose a vector λ such that the inner products of λ and the generators of K_i are not equal to zero. We choose $\lambda = (1, -1)$ in this example. Then, reduce multivariate to univariate with λ, so that we have:

$$\begin{aligned} f(K_1; t) &= \frac{t^{-1}}{(1-t)(1-t^2)}, \\ f(K_2; t) &= \frac{t}{(1-t^{-1})(1-t^{-2})}, \\ f(K_3; t) &= \frac{1}{(1-t)(1-t^{-1})}. \end{aligned}$$

We want to have all the denominators to have positive exponents. We simplify them in order to eliminate negative exponents in the denominators with simple algebra. Then, we have:

$$\begin{aligned} f(K_1; t) &= \frac{t^{-1}}{(1-t)(1-t^2)}, \\ f(K_2; t) &= \frac{t^4}{(1-t)(1-t^2)}, \\ f(K_3; t) &= \frac{-t}{(1-t)(1-t)}. \end{aligned}$$

We factor out t^{-1} from each rational function, so that we obtain:

$$\begin{aligned} f(K_1;t) &: \frac{1}{(1-t)(1-t^2)}, \\ f(K_2;t) &: \frac{t^5}{(1-t)(1-t^2)}, \\ f(K_3;t) &: \frac{-t^2}{(1-t)(1-t)}. \end{aligned}$$

We substitute $t = s+1$ and simplify them to the form $\frac{P(s)}{s^d Q(s)}$:

$$\begin{aligned} f(K_1;s) &: \frac{1}{s^2(2+s)}, \\ f(K_2;s) &: \frac{1+5s+10s^2+10s^3+5s^4+s^5}{s^2(2+s)}, \\ f(K_3;s) &: \frac{-(1+2s+s^2)}{s^2}. \end{aligned}$$

Now we use the recurrence relation to obtain the coefficient of the constant terms. We have $c_2 = \frac{1}{8}$ for $f(K_1)$, $c_2 = \frac{31}{8}$ for $f(K_2)$, and $c_2 = -1$ for $f(K_3)$. Thus, if we sum up all these coefficients, we have 3, which is the number of lattice points in this triangle.

9.13 What if the polytope P is not full-dimensional?

Before we end our description of `LattE`, we must comment on how we deal with polytopes that are not full-dimensional (e.g. transportation polytopes). Given the lower-dimensional polytope $P = \{x \in \mathbb{R}^n : Ax = a, Bx \leq b\}$ with the $d \times n$ matrix A of full row-rank, we will use the equations to transform P into a polytope $Q = \{x \in \mathbb{R}^{n-d} : Cx \leq c\}$ in fewer variables, whose integer points are in one-to-one correspondence to the integer points of P. This second polytope will be the input to the main part of `LattE`. The main idea of this transformation is to find the general integer solution $x = x_0 + \sum_{i=1}^{n-d} \lambda_i g_i$ to $Ax = a$ and to substitute it into the inequalities $Bx \leq b$, giving a new full-dimensional system $Cx \leq c$ in $n-d$ variables $\lambda_1, \ldots, \lambda_{n-d}$. This can be done using the Hermite normal form of A and is explained in Chapter 3.

9.14 Computing an example with `LattE`

Let us demonstrate how to count with the `LattE`. We wish to count the number of 4×4 (non-negative integer) tables that have pre-described row counts and column counts as given in the following table. In other words, in how many different ways can we replace the question marks by non-negative integers such that we obtain a table with the given row and column counts.

?	?	?	?	220
?	?	?	?	215
?	?	?	?	93
?	?	?	?	64
108	286	71	127	

First we encode the polytope into a file, say "4x4":

```
8 17
220 -1 -1 -1 -1  0  0  0  0  0  0  0  0  0  0  0  0
215  0  0  0  0 -1 -1 -1 -1  0  0  0  0  0  0  0  0
 93  0  0  0  0  0  0  0  0 -1 -1 -1 -1  0  0  0  0
 64  0  0  0  0  0  0  0  0  0  0  0  0 -1 -1 -1 -1
108 -1  0  0  0 -1  0  0  0 -1  0  0  0 -1  0  0  0
286  0 -1  0  0  0 -1  0  0  0 -1  0  0  0 -1  0  0
 71  0  0 -1  0  0  0 -1  0  0  0 -1  0  0  0 -1  0
127  0  0  0 -1  0  0  0 -1  0  0  0 -1  0  0  0 -1
linearity 8 1 2 3 4 5 6 7 8
nonnegative 16 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

Note that a polytope $\{x : Ax \leq b\}$, where A is a $d \times n$ matrix, is encoded as

```
d n+1
b -A
```

similar to the `cdd` [58] input format. To simplify the input, one may specify which of the inequalities are in fact equations and which variables are in fact non-negative. In both cases, we first state the number of equations or non-negative variables and then we list the indices.

Now we call the `count` function of `LattE`:

```
./count 4x4
```

Pretty quickly, `Latte` returns the answer. There are in fact

$$1,225,914,276,768,514$$

such tables. If we wished, instead, to compute the full Ehrhart series, we simply need to run

```
./ehrhart 4x4
```

The (unsimplified) Ehrhart series will be stored into the file "4x4.rat".

9.15 Computational experience

9.15.1 Height of signed decomposition tree

From the example in Section 9.7, we notice that the determinant of each cone gets much smaller in each step. This is not an accident as Theorem 9.5.1 guarantees that the cardinality of the index set I of cones in the decomposition is bounded polynomially in terms of the determinant of the input matrix. We have looked experimentally at how many levels of iteration are necessary to carry out the decomposition. We observed experimentally that it often grows linearly with the dimension. We tested two kinds of instances. We used random square matrices whose entries are between 0 and 9, thinking of their columns as the generators of a cone centered at the origin. We tested from 2×2 matrices all the way to 8×8 matrices, and we tested fifteen random square matrices for each dimension. We show the results in Table 9.1. For computation, we used a 1 GHz Pentium PC machine running Red Hat Linux.

Dimension	Height of tree	# of cones	| determinant|	Time (seconds)
2	1.33	2.5	11.5	0
3	2.87	12.5	55.7	0
4	3.87	65.7	274.7	0.2
5	5.87	859.4	3875.9	0.3
6	7.47	10308.0	19310.4	3.7
7	8.53	91029.4	72986.3	41.6
8	10.67	2482647.5	1133094.7	2554.5

Table 9.1: Averages of 15 random matrices for computational experiences

The second set of examples comes from the Birkhoff polytope B_n of doubly stochastic matrices [106]. Each vertex of the polytope is a permutation matrix which is a 0/1 matrix whose column sums and row sums are all 1 [106]. We decompose the cone with vertex at the origin and whose rays are the $n!$ permutation matrices. The results are reported in Table 9.2.

Dimension	# of vertices	# of unimodular cones at a vertex cone	Time (seconds)
$B_3 = 4$	6	3	0.1
$B_4 = 9$	24	16	0.2
$B_5 = 16$	120	125	0.5
$B_6 = 25$	720	1296	7.8

Table 9.2: The numbers of unimodular cones for the Birkhoff polytopes

9.15.2 Counting with `LattE` – knapsacks and the Frobenius number

At the moment, we have been able to handle polytopes of dimension 30 and several thousands vertices. It is known that the theoretical upper bound of the number of unimodular cones is 2^{dh}, where $h = \lfloor \frac{\log\log 1.9 - \log\log D}{\log(d-1/d)} \rfloor$ and where D is the volume of the fundamental parallelepiped of the input cone [13]. If we fix the dimension this upper bound becomes polynomial time. Unfortunately, if we do not fix the dimension, this upper bound becomes exponential. In practice this might be costly and some families of polytopes have large numbers of unimodular cones. The cross polytope family, for instance, has many unimodular cones and behaves badly. For example, for the cross polytope in 6 dimensions, with cross6.ine input file [58], `LattE` took 147.63 seconds to finish computing. The number of lattice points of this polytope is obviously 13. Also, for the cross polytope in 8 dimensions, with cross8.ine input file [58], `LattE` took 85311.3 seconds to finish computing, even though this polytope has only 16 vertices and the number of lattice points of this polytope is 17. For all computations, we used a 1 GHz Pentium PC machine running Red Hat Linux.

Let us now report on computations with some hard knapsack-type problems. Suppose we have a set of positive relatively prime integers $\{a_1, a_2, \ldots, a_d\}$. Denote by a the vector $(a_1, a_2, \ldots, a_d)$. Consider the following problem: does there exist a non-negative integral vector x satisfying $a^\intercal x = a_0$ for some positive integer a_0? We take several examples from [1] which have been found to be extremely hard to solve by commercial quality branch-and-bound software. This is very surprising since the number of variables is at most 10.

Problem	a										Frobenius #	Time
cuww1	12223	12224	36674	61119	85569						89643481	0.55s
cuww2	12228	36679	36682	48908	61139	73365					89716838	1.78s
cuww3	12137	24269	36405	36407	48545	60683					58925134	1.27s
cuww4	13211	13212	39638	52844	66060	79268	92482				104723595	2.04s
cuww5	13429	26850	26855	40280	40281	53711	53714	67141			45094583	16.05s
prob1	25067	49300	49717	62124	87608	88025	113673	119169			33367335	47.07s
prob2	11948	23330	30635	44197	92754	123389	136951	140745			14215206	60.58s
prob3	39559	61679	79625	99658	133404	137071	159757	173977			58424799	88.30s
prob4	48709	55893	62177	65919	86271	87692	102881	109765			60575665	59.04s
prob5	28637	48198	80330	91980	102221	135518	165564	176049			62442884	101.78s
prob6	20601	40429	40429	45415	53725	61919	64470	69340	78539	95043	22382774	225.86s
prob7	18902	26720	34538	34868	49201	49531	65167	66800	84069	137179	27267751	177.64s
prob8	17035	45529	48317	48506	86120	100178	112464	115819	125128	129688	21733990	509.78s
prob9	3719	20289	29067	60517	64354	65633	76969	102024	106036	119930	13385099	
prob10	45276	70778	86911	92634	97839	125941	134269	141033	147279	153525	106925261	264.67s

Table 9.3: Infeasible knapsack problems.

It is not very difficult to see that if the right-hand-side value a_0 is large enough, the equation will surely have a non-negative integer solution. The *Frobenius number* for a knapsack problem is the largest value a_0 such that the knapsack problem is infeasible. Aardal and Lenstra [1] solved them using the reformulation in [2]. Their method works significantly better than branch-and-bound using `CPLEX` 6.5. Here we demonstrate that our implementation of Barvinok's algorithm is fairly fast and, on the order of seconds, we resolved the first 15 problems in Table 1 of [1] and verified all are infeasible except *prob9*, where there is a mistake. The vector $(3480, 1, 4, 4, 1, 0, 0, 0, 0, 0)$ is a solution to the right-hand-side $13,385,099$. In fact, using `LattE` we know that the exact number of solutions is $838,908,602,000$. For comparison we named the problems exactly as in Table 1 of [1]. We present our results in Table 9.3. To measure the hardness of the Frobenius problem, it is very interesting to know the number of lattice points if we add 1 to the Frobenius number for each problem. In Table 9.4, we find the number of solutions if we add 1 to the Frobenius number on each of the (infeasible) problems. The speed is practically the same as in the previous case.

problem	RHS	# of lattice points.
cuww1	89643482	1
cuww2	89716839	1
cuww3	58925135	2
cuww4	104723596	1
cuww5	45094584	1
prob1	33367336	859202692
prob2	14215207	2047107
prob3	58424800	35534465752
prob4	60575666	63192351
prob5	62442885	21789552314
prob6	22382775	218842
prob7	27267752	4198350819898
prob8	21733991	6743959
prob10	106925262	102401413506276371

Table 9.4: The number of lattice points if we add 1 to the Frobenius number.

9.15.3 Magic 5×5 squares

We were able to use Barvinok's algorithm to compute sufficiently many counts in order to recover all the counting formulas presented in the previous chapter. However, our challenging example of computing a counting formula for magic 5×5 squares was also not tractable via Barvinok's algorithm. Although it was easy to determine the 1940 vertices of $tP_{5\times5}$ and their adjacencies for any given t, we already failed to dualize the supporting tangent cones for some of the vertices with many (~ 500) adjacent vertices. Seemingly, $tP_{5\times5}$ is too degenerate to be attacked directly by Barvinok's algorithm.

9.16 Homogenization of Barvinok's algorithm

We observed in [44] that a major practical bottleneck of the original Barvinok algorithm in [13] is the fact that a polytope may have too many vertices. Since originally one visits each vertex to compute a rational function at each tangent cone, the result can be costly. For example, the well-known polytope of semi-magic $4 \times 4 \times 4$ cubes has over two million vertices, but only 64 linear inequalities describe the polytope. As described above, we encountered another bottleneck in the dualization step due to the degeneracy of the given polytope.

In such cases we propose a homogenization of Barvinok's algorithm working with a single cone. Instead of encoding the lattice points in P, or in tP in general, we encode the lattice points in the homogenization of P:

$$C := \{(x,t) : x \in tP, x \in \mathbb{R}^d, t \geq 0\}.$$

Note that C is a rational polyhedral cone. Thus, we can apply Barvinok's algorithm to find a short rational generating function

$$\sum_{(x,t)\in C\cap\mathbb{Z}^{d+1}} y^x z^t.$$

in order to encode the lattice points of C. If we substitute $y = \mathbf{1}$ into this expression, using for example the analytical methods from Section 9.11 to deal with possible singularities, we obtain

$$\sum_{t=0}^{\infty} i_P(t) z^t,$$

where, as in the previous chapter, $i_P(t) = |tP \cap \mathbb{Z}^d|$. Note that this substitution step can be done in polynomial time by [15]. We call this approach, that takes P and returns the Ehrhart series $\sum_{t=0}^{\infty} i_P(t)z^t$ the *Homogenized Barvinok Algorithm*. Clearly, in fixed dimension d, this algorithm runs in polynomial time in the size of the input data.

Clearly, a major advantage of this simple homogenization idea is that we do not only compute $i_P(t)$ for fixed values of t, but we compute in fact a short rational expression for the Ehrhart series $\sum_{t=0}^{\infty} i_P(t)z^t$, from which we can then extract any particular value of the quasi-polynomial $i_P(t)$.

Another possible advantage is that we can get the extreme rays of the polar cone C^* of C for free, since the normal vectors of the facets of C (which we can easily construct from the facets of P, are exactly the extreme rays of C^*. If the polytope P has far fewer facets then vertices, then the number of rays of the cone C^* is small. As in the end, we polarize back only simplicial unimodular, the polarization steps never cause a bottleneck in the Homogenized Barvinok Algorithm.

9.17 Challenging counting formulas found with `LattE`

We have also implemented the Homogenized Barvinok Algorithm in `LattE`. We called Maple to simplify the sum of rational expressions encoding $\sum_{t=0}^{\infty} i_P(t)z^t$ into a single rational expression in z. Finally, we were able to find a counting formula for magic 5×5 squares. Currently, only the Homogenized Barvinok Algorithm has been able to compute it.

9.17.1 Magic 5×5 squares

The Ehrhart series for magic 5×5 squares is given by

$$\sum_{t=0}^{\infty} i_{P_{5\times 5}}(t) z^t = p(z)/q(z),$$

where

$$\begin{aligned}
p(z) &= z^{76} + 28z^{75} + 639z^{74} + 11050z^{73} + 136266z^{72} + 1255833z^{71} + 9120009z^{70} + \\
&\quad 54389347z^{69} + 274778754z^{68} + 1204206107z^{67} + 4663304831z^{66} + 16193751710z^{65} + \\
&\quad 51030919095z^{64} + 147368813970z^{63} + 393197605792z^{62} + 975980866856z^{61} + \\
&\quad 2266977091533z^{60} + 4952467350549z^{59} + 10220353765317z^{58} + 20000425620982z^{57} + \\
&\quad 37238997469701z^{56} + 66164771134709z^{55} + 112476891429452z^{54} + \\
&\quad 183365550921732z^{53} + 287269293973236z^{52} + 433289919534912z^{51} + \\
&\quad 630230390692834z^{50} + 885291593024017z^{49} + 1202550133880678z^{48} + \\
&\quad 1581424159799051z^{47} + 2015395674628040z^{46} + 2491275358809867z^{45} + \\
&\quad 2989255690350053z^{44} + 3483898479782320z^{43} + 3946056312532923z^{42} + \\
&\quad 4345559454316341z^{41} + 4654344257066635z^{40} + 4849590327731195z^{39} + \\
&\quad 4916398325176454z^{38} + 4849590327731195z^{37} + 4654344257066635z^{36} + \\
&\quad 4345559454316341z^{35} + 3946056312532923z^{34} + 3483898479782320z^{33} + \\
&\quad 2989255690350053z^{32} + 2491275358809867z^{31} + 2015395674628040z^{30} + \\
&\quad 1581424159799051z^{29} + 1202550133880678z^{28} + 885291593024017z^{27} + \\
&\quad 630230390692834z^{26} + 433289919534912z^{25} + 287269293973236z^{24} + \\
&\quad 183365550921732z^{23} + 112476891429452z^{22} + 66164771134709z^{21} + \\
&\quad 37238997469701z^{20} + 20000425620982z^{19} + 10220353765317z^{18} + \\
&\quad 4952467350549z^{17} + 2266977091533z^{16} + 975980866856z^{15} + 393197605792z^{14} + \\
&\quad 147368813970z^{13} + 51030919095z^{12} + 16193751710z^{11} + 4663304831z^{10} + \\
&\quad 1204206107z^{9} + 274778754z^{8} + 54389347z^{7} + 9120009z^{6} + 1255833z^{5} + 136266z^{4} + \\
&\quad 11050z^{3} + 639z^{2} + 28z + 1, \\
q(z) &= \left(z^2-1\right)^{10}\left(z^2+z+1\right)^{7}\left(z^7-1\right)^{2}\left(z^6+z^3+1\right)\left(z^5+z^3+z^2+z+1\right)^{4}(1-z)^{3}\left(z^2+1\right)^{4}.
\end{aligned}$$

9.17.2 Magic $3 \times 3 \times 3 \times 3$ cubes

We were now also able to compute the Ehrhart series for magic $3 \times 3 \times 3 \times 3$ cubes. It is given by

$$\sum_{t=0}^{\infty} i_{P_{3\times3\times3\times3}}(t) z^t = p(z)/q(z),$$

where

$$\begin{aligned}
p(z) &= z^{54} + 150z^{51} + 5837z^{48} + 63127z^{45} + 331124z^{42} + 1056374z^{39} + 2326380z^{36} + \\
&\quad 3842273z^{33} + 5055138z^{30} + 5512456z^{27} + 5055138z^{24} + 3842273z^{21} + 2326380z^{18} + \\
&\quad 1056374z^{15} + 331124z^{12} + 63127z^{9} + 5837z^{6} + 150z^{3} + 1, \\
q(z) &= \left(z^3+1\right)^{4}\left(z^{12}+z^9+z^6+z^3+1\right)\left(1-z^3\right)^{9}\left(z^6+z^3+1\right).
\end{aligned}$$

Chapter 10

Integer linear programming using Barvinok's rational functions

This chapter presents two algebraic-analytic algorithms for solving integer linear programming problems based on the generating function techniques of Barvinok [13] and the recent advances by Barvinok and Woods [16]. These are the *BBS algorithm* based on Barvinok's binary search idea proposed in [14] and the *digging algorithm*, our improvement of Lasserre's heuristic method for solving integer programs [91].

10.1 The BBS algorithm

In 1993, Barvinok gave an algorithm that *counts* lattice points in convex rational polyhedra in polynomial time when the dimension of the polytope is fixed, Chapter 9. Originally, Barvinok's counting algorithm relied on Lenstra's polynomial time algorithm for integer programming in a fixed number of variables [93], but shortly after Barvinok's breakthrough, Dyer and Kannan [52] showed that this step can be replaced by a short-vector computation using the LLL algorithm. Therefore, using binary search, one can turn Barvinok's counting oracle into an algorithm that solves integer programming problems with a fixed number of variables in polynomial time (i.e. by counting the number of lattice points in P that satisfy $c^\intercal x \geq \alpha$, we can narrow the range for the maximum value of $c^\intercal x$, then we iteratively look for the largest α where the count is non-zero). This idea was proposed by Barvinok in [14]. We call this IP algorithm the *BBS algorithm*.

10.2 The digging algorithm

More recently, Lasserre outlined a very easy asymptotic heuristic method for solving integer programs [91], or at least providing an upper bound on the optimal value, which is also based on

Barvinok's rational functions (it comes without complexity guarantees). Unfortunately, Lasserre's criteria, needed to find an optimum value, often fail in practice.

We improve Lasserre's heuristic and give a deterministic IP algorithm based on Barvinok's rational function algorithms, the *digging algorithm*. In this case the algorithm can have an exponential number of steps even for fixed dimension, but performs well in practice.

10.2.1 General setup

We consider the integer programming problem $\max\{c^\intercal x : Ax \leq b, x \geq 0, x \in \mathbb{Z}^d\}$, where $c \in \mathbb{Z}^d$ is arbitrary, and $A \in \mathbb{Z}^{m\times d}$, $b \in \mathbb{Z}^m$ are fixed. We assume that the input system of inequalities $Ax \leq b, x \geq 0$ defines a bounded polytope $P \subset \mathbb{R}^d$, such that $P \cap \mathbb{Z}^d$ is nonempty. As before, all integer points are encoded as a short rational function

$$f(P;z) = \sum_{i\in I} E_i \frac{z^{u_i}}{\prod_{j=1}^d (1-z^{v_{ij}})}. \tag{10.2.1}$$

for P, where the rational function is given in Barvinok's form. Remember that if we were to expand Equation (10.2.1) into monomials (generally a very bad idea!) we would get $f(P;z) = \sum_{\alpha\in P\cap\mathbb{Z}^d} z^\alpha$. For a given $c \in \mathbb{Z}^d$, we make the substitution $z_i = t^{c_i}$, Equation (10.2.1) yields a univariate rational function in t:

$$f(P;t) = \sum_{i\in I} E_i \frac{t^{c^\intercal u_i}}{\prod_{j=1}^d (1-t^{c^\intercal v_{ij}})}. \tag{10.2.2}$$

The key observation is that if we make that substitution directly into the monomial expansion of $f(P;z)$, we have that $z^\alpha \to t^{c^\intercal \alpha}$. Moreover we would obtain the relation

$$f(P;t) = \sum_{\alpha\in P\cap\mathbb{Z}^d} t^{c^\intercal \alpha} = kt^M + (\text{lower degree terms}), \tag{10.2.3}$$

where M is the optimal value of our integer program and where k counts the number of optimal integer solutions. Unfortunately, in practice, M and the number of lattice points in P may be huge and we need to avoid the monomial expansion step altogether. All computations have to be done by manipulating short rational functions.

10.2.2 Lasserre's approach

Lasserre [91] suggested the following approach: For $i \in I$, define sets η_i by

$$\eta_i = \{j \in \{1,\ldots,d\} : c^\intercal v_{ij} > 0\},$$

and define vectors w_i by $w_i = u_i - \sum_{j\in\eta_i} v_{ij}$. Let n_i denote the cardinality of η_i. Now define

$$\begin{aligned} M &= \max\{c^\intercal w_i : i \in I\}, \\ S &= \{i \in I | c^\intercal w_i = M\}, \\ \sigma &= \sum_{i\in S} E_i(-1)^{n_i}. \end{aligned}$$

Note that M simply denotes the highest exponent of t appearing in the expansions of the rational functions defined for each $i \in I$ in (10.2.2). The number σ is in fact the sum of the coefficients of t^M in these expressions, that is, σ is the coefficient of t^M in $f(P;t)$. Now with these definitions and notation we can state the following result proved by Lasserre [91].

Theorem 10.2.1 (Theorem 3.1 in [91]) *If $c^\intercal v_{ij} \neq 0$ for all $i \in I, j \in \{1, \ldots, d\}$, and if $\sigma \neq 0$, then M is the optimal value π of the integer program* $\max\{c^\intercal x : Ax \leq b, x \geq 0, x \in \mathbb{Z}^d\}$.

When the hypotheses of Theorem 10.2.1 are met, from an easy inspection, we could recover the optimal value of an integer program. If we assume that c is random, the first condition is not difficult to obtain. Unfortunately, our computational experiments (see Section 10.3) indicate that the condition $\sigma \neq 0$ is satisfied only occasionally. Thus an improvement on the approach that Lasserre proposed is needed to make the heuristic terminate in all instances. Here we explain the details of an algorithm that *digs* for the coefficient of the next highest appearing exponent of t. For simplicity our explanation assumes the easy-to-achieve condition $c^\intercal v_{ij} \neq 0$.

10.2.3 Digging approach

As before, take Equation (10.2.1) computed via Barvinok's algorithm. Now, for the given $c \in \mathbb{Z}^d$, we make the substitutions $z_k = y_k t^{c_k}$, for $k = 1, \ldots, d$. Then substitution into (10.2.1) yields, for the right-hand side of Equation (10.2.1), a sum of multivariate rational functions in the vector variable y and scalar variable t:

$$g(P; y, t) = \sum_{i \in I} E_i \frac{y^{u_i} t^{c^\intercal u_i}}{\prod_{j=1}^{d} (1 - y^{v_{ij}} t^{c^\intercal v_{ij}})}. \tag{10.2.4}$$

On the other hand, the substitution on the left-side of Equation (10.2.1) gives a sum of monomials, also shown grouped in terms of t.

$$g(P; y, t) = \sum_{\alpha \in P} y^\alpha t^{c^\intercal \alpha} = \sum_{\alpha \in \mathbb{Z}^d, n \in \mathbb{Z}} a_{\alpha,n} y^\alpha t^n. \tag{10.2.5}$$

Both equations, (10.2.4) and (10.2.5), represent the same function $g(P; y, t)$, so the corresponding coefficients of the Laurent series expansions of both expressions must be equal. What Barvinok's algorithm provides us is the right-hand side of Equation (10.2.4) and we need to manipulate it to obtain the coefficient of highest degree in t from Equation (10.2.5) (because P is a polytope, there will be a highest degree monomial, from which we recover the optimal value). The process is the following. Apply the identity

$$\frac{1}{1 - y^{v_{ij}} t^{c^\intercal v_{ij}}} = \frac{-y^{-v_{ij}} t^{-c^\intercal v_{ij}}}{1 - y^{-v_{ij}} t^{-c^\intercal v_{ij}}} \tag{10.2.6}$$

to Equation (10.2.4), so that any v_{ij} such that $c^\intercal v_{ij} > 0$ can be changed in "sign" to be sure that, for all v_{ij} in (10.2.4), $c^\intercal v_{ij} < 0$ is satisfied (we may have to change some of the E_i, u_i and v_{ij} using our identity, but we abuse notation and still refer to the new signs as E_i and the new numerator vectors as u_i and the new denominator vectors as v_{ij}). Then, for each of the rational functions in the sum of Equation (10.2.4) compute a Laurent expansion of the form

$$E_i\, y^{u_i} t^{c^\intercal u_i} \prod_{j=1}^{d} (1 + y^{v_{ij}} t^{c^\intercal v_{ij}} + (y^{v_{ij}} t^{c^\intercal v_{ij}})^2 + \ldots). \tag{10.2.7}$$

Add terms of the same degree in t from the different resultant series. We obtain the coefficients $a_{\alpha,n}$ appearing in the terms of the series (10.2.5). Thus, we have an algorithm to solve integer programs:

Algorithm 10.2.2 (Digging Algorithm)

Input: A, b, c.

Output: optimal value and optimal solution of $\max\{c^\intercal x : Ax \leq b, x \geq 0, x \in \mathbb{Z}^d\}$.

1. Use the identity (10.2.6) as necessary to enforce that all v_{ij} in (10.2.4) satisfy $c^\intercal v_{ij} < 0$.
2. Via the expansion formulas (10.2.7), find (10.2.5) by calculating the terms' coefficients. Then proceed in decreasing order with respect to the degree of t. This can be done because, for each series appearing in the expansion formulas (10.2.7), the terms of the series are given in decreasing order with respect to the degree of t.
3. Continue calculating the terms of the expansion (10.2.5), in decreasing order with respect to the degree of t, until a degree n of t is found such that for some $\alpha \in \mathbb{Z}^d$, the coefficient of $y^\alpha t^n$ is non-zero in the expansion (10.2.5).
4. Return "n" as the optimal value of the integer program and return α as an optimal solution.

We close this section mentioning that we can recover not only the optimal value, but also an explicit optimal solution.

Remark. There is also a variation of the digging algorithm where instead of using Barvinok's rational function for the whole polytope, one uses the Barvinok rational function only of a tangent cone at a chosen vertex (typically the vertex is LP relaxation optimal solution). Then we dig in this cone for a lattice point of the polytope with an optimal objective value. We observed speed ups in practice in some of the cases. □

10.3 Computational experiments

In this section we report our experience solving hard knapsack problems from [1, 35]. See Table 10.1 for the data used here. Their form is $\max\{c^\intercal x : a^\intercal x = b, x \geq 0, x \in \mathbb{Z}^d\}$, where $b \in \mathbb{Z}$ and where $a \in \mathbb{Z}^d$ with $\gcd(a_1, \ldots, a_d) = 1$. For the cost vector c, we choose the first d components of the vector

$(213, -1928, -11111, -2345, 9123, -12834, -123, 122331, 0, 0)$. We compared the digging algorithm and the BBS algorithm, both implemented in `LattE`, with `CPLEX` version 6.6. The computations were done on a 1 GHz Pentium PC running Red Hat Linux. Table 10.2 provides the optimal values and an optimal solution for each problem. As it turns out, there is exactly one optimal solution for each problem.

Problem	a										b
cuww1	12223	12224	36674	61119	85569						89643482
cuww2	12228	36679	36682	48908	61139	73365					89716839
cuww3	12137	24269	36405	36407	48545	60683					58925135
cuww4	13211	13212	39638	52844	66060	79268	92482				104723596
cuww5	13429	26850	26855	40280	40281	53711	53714	67141			45094584
prob1	25067	49300	49717	62124	87608	88025	113673	119169			33367336
prob2	11948	23330	30635	44197	92754	123389	136951	140745			14215207
prob3	39559	61679	79625	99658	133404	137071	159757	173977			58424800
prob4	48709	55893	62177	65919	86271	87692	102881	109765			60575666
prob5	28637	48198	80330	91980	102221	135518	165564	176049			62442885
prob6	20601	40429	40429	45415	53725	61919	64470	69340	78539	95043	22382775
prob7	18902	26720	34538	34868	49201	49531	65167	66800	84069	137179	27267752
prob8	17035	45529	48317	48506	86120	100178	112464	115819	125128	129688	21733991
prob9	3719	20289	29067	60517	64354	65633	76969	102024	106036	119930	13385100
prob10	45276	70778	86911	92634	97839	125941	134269	141033	147279	153525	106925262

Table 10.1: Knapsack problems.

Problem	Value	Solution	Digging	BBS	CPLEX 6.6
cuww1	1562142	[7334 0 0 0 0]	0.4 sec.	414 sec.	> 1.5h
cuww2	-4713321	[3 2445 0 0 0 0]	> 3.5h	6,600 sec.	> 0.75h
cuww3	1034115	[4855 0 0 0 0 0]	1.4 sec.	6,126 sec.	> 0.75h
cuww4	-29355262	[0 0 2642 0 0 0 0]	> 1.5h	38,511 sec.	> 0.75h
cuww5	-3246082	[1 1678 1 0 0 0 0 0]	> 1.5h	> 80h	> 0.75h
prob1	9257735	[966 5 0 0 1 0 0 74]	51.4 sec.	> 3h	> 1h
prob2	3471390	[853 2 0 4 0 0 0 27]	24.8 sec.	> 10h	> 0.75h
prob3	21291722	[708 0 2 0 0 0 1 173]	48.2 sec.	> 12h	> 1.5h
prob4	6765166	[1113 0 7 0 0 0 0 54]	34.2 sec.	> 5h	> 1.5h
prob5	12903963	[1540 1 2 0 0 0 0 103]	34.5 sec.	> 5h	> 1.5h
prob6	2645069	[1012 1 0 1 0 1 0 20 0 0]	143.2 sec.	> 4h	> 2h
prob7	22915859	[782 1 0 1 0 0 0 186 0 0]	142.3 sec.	> 4h	> 1h
prob8	3546296	[1 385 0 1 1 0 0 35 0 0]	469.9 sec.	> 3.5h	> 2.5h
prob9	15507976	[31 11 1 1 0 0 0 127 0 0]	1,408.2 sec.	> 11h	4.7 sec.
prob10	47946931	[0 705 0 1 1 0 0 403 0 0]	250.6 sec.	> 11h	> 1h

Table 10.2: Optimal values, optimal solutions, and running times for each problem.

With one exception, CPLEX 6.6. could not solve the given problems. Note that whenever the digging algorithm found the optimal value, it did so much faster than the BBS algorithm. This is interesting, as the worst-case complexity for the digging algorithm is exponential even for fixed dimension, while BBS has polynomial complexity in fixed dimension. The digging algorithm fails to find a solution for problems prob2, prob3, and prob5. What happens is that the expansion step becomes costly when more coefficients have to be computed. In these three examples, we computed coefficients for more than 2,500,000, 400,000, and 100,000 powers of t; all turning out to be 0.

The Digging algorithm is slower than CPLEX in problem prob9 because during the execution of Barvinok's unimodular cone decomposition more than 160,000 cones are generated, leading to an enormous rational function for $f(P;t)$. Moreover, for prob9 more than 3,500 coefficients turned out to be 0, before a non-zero leading coefficient was detected. Finally, in problems cuww1, cuww3, prob2, prob3, prob4, prob6, and prob8, no digging was necessary at all, that is, Lasserre's condition did not fail here. For all other problems, Lasserre's condition did fail and digging steps were necessary to find the first non-vanishing coefficient in the expansion of $f(P;t)$.

Chapter 11

Integer polynomial optimization in fixed dimension

Mixed integer non-linear programs combine the hardness of combinatorial explosion with the non-convexity of non-linear functions. For example, the well-known optimality conditions developed for differentiable objective functions have no meaning when the variables are discrete. Thus, it is perhaps not surprising that already linear integer programming with general quadratic constraints is undecidable [83]. Nevertheless, when the number of variables is fixed discrete optimization problems often become tractable and efficient polynomial algorithms exist (e.g. [13, 86, 93]). It is thus natural to ask *what is the complexity of integer non-linear optimization assuming that the number of variables is fixed?* We study the problem

$$\text{maximize } f(x_1, \ldots, x_d) \text{ subject to } g_i(x_1, \ldots, x_d) \geq 0, x \in \mathbb{Z}^d. \tag{11.0.1}$$

Here f, g_i are polynomials with integral coefficients. Note that all throughout this chapter we assume that the number of variables is fixed. Here are our two contributions to the theory:

(1) We give a classification of the computational complexity of Problem (11.0.1) according to special cases. Section 11.1 presents the details, but the reader can see the classification in Table 11.1. New results are marked with letters, known results are marked with asterisks, arrows indicate implications:

(2) For problem (a), that of optimizing an arbitrary integral polynomial over the lattice points of a convex rational polytope with fixed number of variables, we present an algorithm to compute a sequence of upper and lower bounds for its optimal value. Our bounds can be used, for instance, in a branch-and-bound search for the optimum. We use Barvinok's algebraic encoding of the lattice points of polytopes via rational functions [14]. In Section 11.2 we prove:

Theorem 11.0.1 *Let the number of variables d be fixed. Let $f(x_1, \ldots, x_d)$ be a polynomial of maximum total degree D with integer coefficients, and let P be a convex rational polytope defined*

	Type of objective function				
Type of constraints	linear		convex polynomial		arbitrary polynomial
Linear constraints, integer variables	polytime $(*)$	$\Leftarrow$	polytime $(**)$		NP-hard (a)
	$\Uparrow$		$\Uparrow$		$\Downarrow$
Convex semialgebraic constraints, integer variables	polytime $(**)$	$\Leftarrow$	polytime $(**)$		NP-hard (c)
Arbitrary polynomial constraints, integer variables	undecidable (b)	$\Rightarrow$	undecidable (d)	$\Rightarrow$	undecidable (e)

Table 11.1: Computational complexity of Problem (11.0.1) in fixed dimension.

by linear inequalities in d variables. We obtain an increasing sequence of lower bounds $\{L_k\}$ and a decreasing sequence of upper bounds $\{U_k\}$ to the optimal value

$$f^* = \max\ f(x_1, x_2, \ldots, x_d) \text{ subject to } x \in P \cap \mathbb{Z}^d. \tag{11.0.2}$$

The bounds L_k, U_k can be computed in time polynomial in k, the input size of P and f, and the maximum total degree D. If f is non-negative over the polytope (i.e. $f(x) \geq 0$ for all $x \in P$), they satisfy the inequality $U_k - L_k \leq f^* \cdot (\sqrt[k]{|P \cap \mathbb{Z}^d|} - 1)$.

More strongly, if f is non-negative over the polytope, there exists a fully polynomial-time approximation scheme (FPTAS) for the optimization problem (11.0.2).

11.1 Computational complexity bounds

All the results we present refer to the complexity model where the number of operations is given in terms of the input size measured in the standard binary encoding. The results of Lenstra Jr. [93] imply the entry of Table 11.1 marked with $(*)$, i.e. solving linear integer programming problems with a fixed number of variables can be done in time polynomial in the size of the input. More recently, Khachiyan and Porkolab [86] have proved that in fixed dimension, the problem of minimizing a convex polynomial objective function over the integers, subject to polynomial constraints that define a convex body, can be solved in polynomial time in the encoding length of the input. Thus, they settled all entries marked by $(**)$. By the natural containment exhibited by these complexity classes, to show the validity of the remaining entries of Table 11.1 is enough to prove the following lemma:

Lemma 11.1.1 *1. The problem of minimizing a degree four polynomial over the lattice points of a convex polygon is NP-hard (entry* (*a*) *in Table 11.1).*

2. The problem of minimizing a linear form over polynomial constraints in at most 10 integer variables is not computable by a recursive function (entry (*b*) *in Table 11.1).*

Proof. (1) We use the NP-complete problem AN1 on page 249 of [62]. This problem states it is NP-complete to decide whether, given three positive integers a, b, c, there exists a positive integer

$x < c$ such that x^2 is congruent with a modulo b. This problem is clearly equivalent to asking whether the minimum of the quartic polynomial function $(x^2 - a - by)^2$ over the lattice points of the rectangle $\{(x, y) : 1 \le x \le c - 1, \frac{1-a}{b} \le y \le \frac{(c-1)^2-a}{b}\}$ is zero or not. This settles part (1).

(2) In 1973, Jeroslow [83] proved a similar result without fixing the number of variables. We follow his idea, but resorting to a stronger lemma. More precisely our proof relies on a 1982 result [85] which states that there is no recursive function that, given an integer polynomial f with *nine* variables, can determine whether f has a non-negative integer zero, in the sense that it finds an explicit zero or returns null otherwise. Jones paper is a strengthening of the original solution of Hilbert's tenth problem [95]. Now to each polynomial f in $\mathbb{Z}[x_1, x_2 \ldots, x_9]$ associate the ten-dimensional minimization problem

$$\text{minimize } y \text{ subject to } \quad (1 - y)f(x_1, x_2, \ldots, x_9) = 0, \ (y, x_1, \ldots, x_9) \in \mathbb{Z}^{10}_{\ge 0}. \tag{11.1.1}$$

The minimum attained by y is either zero or one depending on whether f has an integer non-negative solution or not. Thus part (2) is settled. □

11.2 FPTAS for optimizing non-negative polynomials over integer points of polytopes

Consider now a polynomial function $f \in \mathbb{Z}[x_1, x_2, \ldots, x_d]$ of maximum total degree D and a convex polytope $P = \{x|Ax \le b\}$ where A is an $m \times d$ integral matrix and b is an integral m-vector. The purpose of this section is to present an algorithm to generate lower and upper bounds L_k, U_k to the integer global optimum value of

$$\text{maximize } f(x_1, \ldots, x_d) \text{ subject to } (x_1, \ldots, x_d) \in P \cap \mathbb{Z}^d. \tag{11.2.1}$$

We should also remark that in our algorithm the polynomial objective function f can be arbitrary (e.g. non-convex). As we have seen, the optimization problem is NP-hard already for two integer variables and polynomials of degree four. Nevertheless we will see that, in fixed dimension and when $f(x) \ge 0$ for all $x \in P$, the algorithm gives a fully polynomial time approximation scheme or FPTAS. This means that, in polynomial time on the input and $(1/\epsilon)$, one can compute a $(1 - \epsilon)$-approximation to the maximum. The algorithm we present is based on Barvinok's theory for encoding all the lattice points of a polyhedron in terms of short rational functions. The set of lattice points is represented by a Laurent polynomial: $g_P(z) = \sum_{\alpha \in P \cap \mathbb{Z}^d} z^\alpha$. From Barvinok's theory this exponentially-large sum of monomials $g_P(z)$ can instead be written as a polynomial-size sum of rational functions (assuming the dimension d is fixed) of the form:

$$g_P(z) = \sum_{i \in I} E_i \frac{z^{u_i}}{\prod_{j=1}^{d} (1 - z^{v_{ij}})}, \tag{11.2.2}$$

where I is a polynomial-size indexing set, and where $E_i \in \{1, -1\}$ and $u_i, v_{ij} \in \mathbb{Z}^d$ for all i and j.

We need a way to encode via rational functions the values of the polynomial f over all the lattice points in a polytope. The key idea, first introduced in Lemma 9 of [40] and generalized in [81], is that differential operators associated to f can be used to compute a rational function representation of $\sum_{a\in P\cap\mathbb{Z}^d} f(a)z^a$. The following Lemma recently appeared in [16]. We are truly grateful to Alexander Barvinok who communicated to us that Lemma 11.2.1 was true for variable D and thus we had indeed obtained an FPTAS from the construction of the upper and lower bounds.

Lemma 11.2.1 *Let $g_P(z)$ be the Barvinok representation of the generating function of the lattice points of P. Let f be a polynomial in $\mathbb{Z}[x_1,\ldots,x_d]$ of maximum total degree D. We can compute, in time polynomial on D and the size of the input data, a Barvinok rational function representation $g_{P,f}(z)$ for the generating function $\sum_{a\in P\cap\mathbb{Z}^d} f(a)z^a$.*

Proof. We give here the authors' original proof the lemma for D fixed. The first proof without this assumption was recently given by Barvinok in [16].

We begin assuming $f(z) = z_r$, the general case will follow from it: Consider the action of the differential operator $z_r\frac{\partial}{\partial z_r}$ in the generating function $g_P(z)$ and on its Barvinok representation. On one hand, for the generating function

$$z_r\frac{\partial}{\partial z_r}\cdot g_P(z) = \sum_{\alpha\in P\cap\mathbb{Z}^d} z_r\frac{\partial}{\partial z_r}z^\alpha = \sum_{\alpha\in P\cap\mathbb{Z}^d}\alpha_r z^\alpha.$$

On the other hand, by linearity of the operator, we have that in terms of rational functions

$$z_r\frac{\partial}{\partial z_r}\cdot g_P(z) = \sum_{i\in I} E_i z_r\frac{\partial}{\partial z_r}\cdot\left(\frac{z^{u_i}}{\prod\limits_{j=1}^{d}(1-z^{v_{ij}})}\right).$$

Thus it is enough to prove that the summands of the expression above can be written in terms of rational functions computable in polynomial time. The standard quotient rule for derivatives says that

$$\frac{\partial}{\partial z_r}\left(\frac{z^{u_i}}{\prod\limits_{j=1}^{d}(1-z^{v_{ij}})}\right) = \frac{(\frac{\partial z^{u_i}}{\partial z_r})\prod_{j=1}^d(1-z^{v_{ij}}) - z^{u_i}(\frac{\partial}{\partial z_r}\prod_{j=1}^d(1-z^{v_{ij}}))}{\prod_{j=1}^d(1-z^{v_{ij}})^2}.$$

We can expand the numerator as a sum of no more than 2^d monomials. This is a constant number because d, the number of variables, is assumed to be a constant. This argument completes the proof of our lemma when $f(z) = z_r$.

For the case when $f(z)$ is a general monomial, i.e. $f(z) = c\cdot z_1^{\beta_1}\cdot\ldots\cdot z_d^{\beta_d}$, then we can compute again a rational function representation of $g_{P,f}(z)$ by repeated application of basic differential operators:

$$c\left(z_1\frac{\partial}{\partial z_1}\right)^{\beta_1}\cdot\ldots\cdot\left(z_d\frac{\partial}{\partial z_d}\right)^{\beta_d}\cdot g_P(z) = \sum_{\alpha\in P\cap\mathbb{Z}^d} c\cdot\alpha^\beta z^\alpha.$$

Thus we require no more than $O(D^d)$ repetitions of the single-variable case.

Finally, if we deal with a polynomial f of many monomial terms, we compute and add up all such expressions that we get for each term of $f(x)$ and obtain our desired short rational function representation for the generating function for $\sum_{\alpha \in P \cap \mathbb{Z}^d} f(\alpha) z^\alpha$. Note that only polynomially many steps are needed because d is fixed and the largest number of possible monomials in f of degree s is $\binom{d+s-1}{d-1}$, thus for fixed d we will do no more than $O(D^d)$ repetitions of the monomial case. $\square$

Now we are ready to present our algorithm to obtain bounds U_k, L_k that reach the optimum. Step 1 of preprocessing is necessary because we rely on the elementary fact that, for a collection $S = \{s_1, \ldots, s_r\}$ of non-negative real numbers, maximum$\{s_i | s_i \in S\}$ equals $\lim_{k \to \infty} \sqrt[k]{\sum_{j=1}^r s_j^k}$.

Algorithm 11.2.2 (Bounding Algorithm)

Input: A rational convex polytope $P \subset \mathbb{R}^d$, a polynomial objective $f \in \mathbb{Z}[x_1, \ldots, x_d]$ of maximum total degree D.

Output: An increasing sequence of lower bounds L_k, and a decreasing sequence of upper bounds U_k reaching the maximal function value f^* of f over all lattice points of P.

1. If f is known to be non-negative in all points of P, then go directly to Step 2. Else, solving $2d$ linear programs over P, we find lower and upper integer bounds for each of the variables $x_1, \ldots, x_d$. Let M be the maximum of the absolute values of these $2d$ numbers. Thus $|x_i| \leq M$ for all i. Let C be the maximum of the absolute values of all coefficients, and r be the number of monomials of $f(x)$. Then

$$L := -rCM^D \leq f(x) \leq rCM^D =: U,$$

   as we can bound the absolute value of each monomial of $f(x)$ by CM^D. Replace f by $\overline{f}(x) = f(x) - L \leq U - L$, a non-negative polynomial over P. Go to Steps 2, 3, etc. and return the optimal value of $\overline{f}$. Trivially, if we find the optimal value of $\overline{f}$ over P we can extract the optimal value for f.

2. Via Barvinok's algorithm, compute a short rational function expression for the generating function $g_P(z) = \sum_{\alpha \in P \cap \mathbb{Z}^d} z^\alpha$. From $g_P(z)$ compute the number $|P \cap \mathbb{Z}^d| = g_P(1)$ of lattice points in P in polynomial time.

3. From the rational function representation $g_P(z)$ of the generating function $\sum\limits_{\alpha \in P \cap \mathbb{Z}^d} z^\alpha$ compute the rational function representation of $g_{P,f^k}(z)$ of $\sum_{\alpha \in P \cap \mathbb{Z}^d} f^k(\alpha) z^\alpha$ in polynomial time by application of Lemma 11.2.1. We define

$$L_k := \sqrt[k]{g_{P,f^k}(1)/g_{P,f^0}(1)} \quad \text{and} \quad U_k := \sqrt[k]{g_{P,f^k}(1)}.$$

   When $\lfloor U_k \rfloor - \lceil L_k \rceil < 1$ stop and return $\lceil L_k \rceil = \lfloor U_k \rfloor$ as the optimal value.

Lemma 11.2.3 *The algorithm is correct.*

Proof. Using the fact that the arithmetic mean of a finite set of non-negative values is at most as big as the maximum value, which in turn is at most as big as the sum of all values, we obtain the sequences of lower and upper bounds, L_k and U_k, for the maximum:

$$L_k = \sqrt[k]{\frac{\sum\limits_{\alpha \in P \cap \mathbb{Z}^d} f(\alpha)^k}{|P \cap \mathbb{Z}^d|}} \leq \max\{f(\alpha) : \alpha \in P \cap \mathbb{Z}^d\} \leq \sqrt[k]{\sum_{\alpha \in P \cap \mathbb{Z}^d} f(\alpha)^k} = U_k.$$

Note that as $s \to \infty$, L_k and U_k approach this maximum value monotonously (from below and above, respectively). Trivially, if the difference between (rounded) upper and lower bounds becomes strictly less than 1, we have determined the value $\max\{f(x) : x \in P \cap \mathbb{Z}^d\} = \lceil L_k \rceil$. Thus the algorithm terminates with the correct answer. □

Theorem 11.0.1 will follow from the next lemma:

Lemma 11.2.4 *Let f be a polynomial with integer coefficients and maximum total degree D that is non-negative over the polytope P. When the dimension d is fixed,*

1. *the bounds L_k, U_k can be computed in time polynomial in k, the input size of P and f, and the total degree D. The bounds satisfy the following inequality:*

$$U_k - L_k \leq f^* \cdot \left(\sqrt[k]{|P \cap \mathbb{Z}^d|} - 1 \right).$$

2. *In addition, for $k = (1 + 1/\epsilon)\log(|P \cap \mathbb{Z}^d|)$, L_k is a $(1-\epsilon)$-approximation to the optimal value f^* and it can be computed in time polynomial in the input size, the total degree D, and $1/\epsilon$. Similarly, U_k gives a $(1+\epsilon)$-approximation to f^*. Moreover, with the same complexity, one can also find a feasible lattice point that approximates an optimal solution with similar quality.*

Proof. Part (i). From Lemma 11.2.1 on fixed dimension d, we can compute $g_{P,f} = \sum_{\alpha \in P \cap \mathbb{Z}^d} f(\alpha) z^\alpha$ as a rational function in time polynomial in D, the total degree of f, and the input size of P. Thus, because f^k has total degree of Dk and the encoding length for the coefficients of f^k is bounded by $k\log(kC)$ (with C the largest coefficient in f), we can also compute $g_{P,f^k} = \sum_{\alpha \in P \cap \mathbb{Z}^d} f^k(\alpha) z^\alpha$ in time polynomial in k, the total degree D, and the input size of P. Note that using residue techniques, see [16] or Section 9.11, we can evaluate $g_{P,f^k}(1)$ in polynomial time. Finally observe

$$\begin{aligned} U_k - L_k &= \sqrt[k]{\sum_{\alpha \in P \cap \mathbb{Z}^d} f^k(\alpha)} - \sqrt[k]{\frac{\sum_{\alpha \in P \cap \mathbb{Z}^d} f^k(\alpha)}{|P \cap \mathbb{Z}^d|}} = \sqrt[k]{\frac{\sum_{\alpha \in P \cap \mathbb{Z}^d} f^k(\alpha)}{|P \cap \mathbb{Z}^d|}} \left(\sqrt[k]{|P \cap \mathbb{Z}^d|} - 1 \right) \\ &= L_k \left(\sqrt[k]{|P \cap \mathbb{Z}^d|} - 1 \right) \leq f^* \left(\sqrt[k]{|P \cap \mathbb{Z}^d|} - 1 \right). \end{aligned}$$

Part (ii). Note that if $\left(\sqrt[k]{|P \cap \mathbb{Z}^d|} - 1 \right) \leq \epsilon$ then L_k is indeed a $(1-\epsilon)$-approximation because

$$f^* \leq U_k = L_k + (U_k - L_k) \leq L_k + f^* \left(\sqrt[k]{|P \cap \mathbb{Z}^d|} - 1 \right) \leq L_k + f^* \epsilon.$$

Observe that $\phi(\epsilon) := (1+1/\epsilon)/(1/\log(1+\epsilon))$ is an increasing function for $\epsilon < 1$ and $\lim_{\epsilon\to 0}\phi(\epsilon) = 1$, thus $\phi(\epsilon) \geq 1$ for $0 < \epsilon \leq 1$. Hence, for all

$$k \geq \log(|P \cap \mathbb{Z}^d|) + \log(|P \cap \mathbb{Z}^d|)/\epsilon \geq \log(|P \cap \mathbb{Z}^d|)/\log(1+\epsilon),$$

we have indeed $\left(\sqrt[k]{|P \cap \mathbb{Z}^d|} - 1\right) \leq \epsilon$. Finally, from Lemma 11.2.1, the calculation of L_k for $k = \log(|P \cap \mathbb{Z}^d|) + \log(|P \cap \mathbb{Z}^d|)/\epsilon$ would require a number of steps polynomial in the input size and $1/\epsilon$. A very similar argument can be written for U_k but we omit it here.

To complete the proof of part (ii) it remains to show that not only we approximate the optimal value f^* but we can also efficiently find a lattice point α with $f(\alpha)$ giving that quality approximation of f^*. Let $k = (1+1/\epsilon)\log(|P \cap \mathbb{Z}^d|)$, thus, by the above discussion, L_k is an $(1-\epsilon)$-approximation to f^*. Let $Q_0 := [-M, M]^d$ denote the box computed in Step 1 of the algorithm such that $P \subseteq Q_0$. By bisecting Q_0, we obtain two boxes Q_1' and Q_1''. By applying the algorithm separately to the polyhedra $P \cap Q_1'$ and $P \cap Q_1''$, we compute lower bounds L_k' and L_k'' for the optimization problems restricted to Q_1' and Q_1'', respectively. Because L_k^k is the arithmetic mean of $f^k(\alpha)$ for $\alpha \in P \cap \mathbb{Z}^d$, clearly

$$\min\{L_k', L_k''\} \leq L_k \leq \max\{L_k', L_k''\}.$$

Without loss of generality, let $L_k' \geq L_k''$. We now apply the bisection procedure iteratively on Q_k'. After $d \log M$ bisection steps, we obtain a box Q_k' that contains a single lattice point $\alpha \in P \cap Q_k' \cap Z^d$, which has an objective value $f(\alpha) = L_k' \geq L_k \geq (1-\epsilon)f^*$. □

We remark that if we need to apply the construction of Step 1 of the algorithm because f takes negative values on P, then we can only obtain an $(1-\epsilon)$-approximation (and $(1+\epsilon)$-approximation, respectively) for the modified function $\overline{f}$ in polynomial time, but not the original function f. We also emphasize that, although our algorithm requires the computation of $\sum_{\alpha\in P} f^q(\alpha)$ for different powers of f, these numbers are obtained without explicitly listing all lattice points (a hard task), nor we assume any knowledge of the individual values $f(\alpha)$. We can access the power means $\sum_{\alpha\in P} f^q(\alpha)$ indirectly via rational functions. Here are two small examples:

Example 11.2.5 (Monomial optimization over a quadrilateral) The problem we consider is that of maximizing the value of the monomial x^3y over the lattice points of the quadrilateral

$$\{(x,y)|3991 \leq 3996\,x - 4\,y \leq 3993,\ 1/2 \leq x \leq 5/2\}.$$

It contains only 2 lattice points. The sum of rational functions encoding the lattice points is

$$\frac{x^2y^{1000}}{(1-(xy^{999})^{-1})\,(1-y^{-1})} + \frac{xy}{(1-xy^{999})\,(1-y^{-1})} + \frac{xy}{(1-xy^{999})\,(1-y)} + \frac{x^2y^{1000}}{(1-(xy^{999})^{-1})\,(1-y)}.$$

In the first iteration, we get $L_1 = 4000.50$ while $U_1 = 8001$. After thirty iterations, we see that $L_{30} = 7817.279750$ while $U_{30} = 8000$, the true optimal value. □

Example 11.2.6 (nvs04 from MINLPLIB) A somewhat more complicated example, from a well-known library of test examples (see `http://www.gamsworld.org/minlp/`), is the problem given by

$$\begin{aligned} \min \quad & 100\left(\frac{1}{2}+i_2-\left(\frac{3}{5}+i_1\right)^2\right)^2+\left(\frac{2}{5}-i_1\right)^2 \\ \text{s.t.} \quad & i_1, i_2 \in [0,200]\cap\mathbb{Z}. \end{aligned} \tag{11.2.3}$$

Its optimal solution as given in MINLPLIB is $i_1 = 1$, $i_2 = 2$ with an objective value of 0.72. Clearly, to apply our algorithm from page 145 literally, the objective function needs to be multiplied by a factor of 100 to obtain an integer valued polynomial.

Using the bounds on i_1 and i_2 we obtain an upper bound of $165 \cdot 10^9$ for the objective function, which allows us to convert the problem into an equivalent maximization problem, where all feasible points have a non-negative objective value. The new optimal objective value is 164999999999.28. Expanding the new objective function and translating it into a differential operator yields

$$\frac{4124999999947}{25}\mathrm{Id} - 28z_2\frac{\partial}{\partial z_2} + \frac{172}{5}z_1\frac{\partial}{\partial z_1} - 117\left(z_1\frac{\partial}{\partial z_1}\right)^{(2)} - 100\left(z_2\frac{\partial}{\partial z_2}\right)^{(2)}$$

$$+240\left(z_2\frac{\partial}{\partial z_2}\right)\left(z_1\frac{\partial}{\partial z_1}\right)+200\left(z_2\frac{\partial}{\partial z_2}\right)\left(z_1\frac{\partial}{\partial z_1}\right)^{(2)}-240\left(z_1\frac{\partial}{\partial z_1}\right)^{(3)}-100\left(z_1\frac{\partial}{\partial z_1}\right)^{(4)}.$$

The short generating function can be written as

$$g(z_1,z_2)=\left(\frac{1}{1-z_1}-\frac{z_1^{201}}{1-z_1}\right)\left(\frac{1}{1-z_2}-\frac{z_2^{201}}{1-z_2}\right).$$

In this example, the number of lattice points is $|P\cap\mathbb{Z}^2| = 40401$. As the first bounds we get $L_1 = 139463892042.292155534$, $U_1 = 28032242300500.723262442$. After 30 iterations the bounds become $L_{30} = 164999998845.993553019$ and $U_{30} = 165000000475.892451381$. □

Chapter 12

FPTAS for mixed-integer polynomial optimization in fixed dimension

A well-known result by Lenstra Jr. states that *linear* mixed integer programming problems with fixed number of variables can be solved in polynomial time on the input size [93]. It is a natural question to ask what is the computational complexity, when the number of variables is fixed, of the *non-linear* mixed integer problem

$$\begin{aligned} \max \quad & f(x_1, \ldots, x_{d_1}, z_1, \ldots, z_{d_2}) && && (12.0.1a) \\ \text{s.t.} \quad & A\mathbf{x} + B\mathbf{z} \leq \mathbf{b} && && (12.0.1b) \\ & x_i \in \mathbb{R} && \text{for } i = 1, \ldots, d_1, && (12.0.1c) \\ & z_i \in \mathbb{Z} && \text{for } i = 1, \ldots, d_2, && (12.0.1d) \end{aligned}$$

where f is a polynomial function of maximum total degree D with rational coefficients, and where $A \in \mathbb{Z}^{p \times d_1}$, $B \in \mathbb{Z}^{p \times d_2}$, $\mathbf{b} \in \mathbb{Z}^p$ (here we assume that $A\mathbf{x} + B\mathbf{z} \leq \mathbf{b}$ describes a convex polytope, which we denote by P).

It was well-known that continuous polynomial optimization over polytopes, without fixed dimension, is NP-hard and that an FPTAS is not possible. Indeed the max-cut problem can be modeled as minimizing a quadratic form over the cube $[-1, 1]^d$ [68]. More strongly, it turns out that, even for dimension two and total degree of f four, problem (12.0.1) is an NP-hard problem too [42]. Thus the best we can hope for, even for fixed dimension, is an approximation result similar to that for the pure integer situation. In this chapter, we present the best possible such result:

Theorem 12.0.1 *Let the dimension $d = d_1 + d_2$ be fixed.*

(a) *There exists a fully polynomial time approximation scheme (FPTAS) for the optimization problem* (12.0.1) *for all polynomial functions $f \in \mathbb{Q}[x_1, \ldots, x_{d_1}, z_1, \ldots, z_{d_2}]$ that are non-negative* *on the feasible region* (12.0.1b–12.0.1d). *(We assume the encoding length of f is at least as large as its maximum total degree.)*

(b) *Moreover, the restriction to non-negative polynomials is necessary, as there does not even exist a polynomial time approximation scheme (PTAS) for the maximization of* arbitrary *polynomials over mixed-integer sets in polytopes, even for fixed dimension $d \geq 2$.*

The proof of Theorem 12.0.1 is presented in Section 12.3. As we will see, Theorem 12.0.1 is a non-trivial consequence of the existence of an FPTAS for the problem of maximizing a non-negative polynomial with integer coefficients over the lattice points of a convex rational polytope; see the previous Chapter for such an algorithm.

Our arguments for the mixed-integer situation, however, are independent of which FPTAS is used in the integral case. Our results come to complement other approximation schemes investigated for continuous variables and fixed degree (see [39] and references therein).

Our main approach is to use grid refinements in order to approximate the mixed-integer optimal value via auxiliary pure integer problems. One of the difficulties on constructing approximations is the fact that not every sequence of grids whose widths converge to zero leads to a convergent sequence of optimal solutions of grid optimization problems. This difficulty is addressed in Section 12.1. In Section 12.2 we develop techniques for bounding differences of polynomial function values. Finally, in Section 12.3 we combine the constructions from Sections 12.1 and 12.2 and give a proof of our main theorem, Theorem 12.0.1.

12.1 Grid approximation results

An important step in the development of an FPTAS for the mixed-integer optimization problem is the reduction of the mixed-integer problem (12.0.1) to an auxiliary optimization problem over a lattice $\frac{1}{m}\mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2}$. To this end, we consider the *grid problem*

$$\begin{aligned} \max \quad & f(x_1, \dots, x_{d_1}, z_1, \dots, z_{d_2}) \\ \text{s.t.} \quad & A\mathbf{x} + B\mathbf{z} \leq \mathbf{b} \\ & x_i \in \tfrac{1}{m}\mathbb{Z} \qquad \text{for } i = 1, \dots, d_1, \\ & z_i \in \mathbb{Z} \qquad \text{for } i = 1, \dots, d_2. \end{aligned} \tag{12.1.1}$$

We can solve this problem approximately using the integer FPTAS (Lemma 11.2.4):

Corollary 12.1.1 *For fixed dimension $d = d_1 + d_2$ there exists an algorithm with running time polynomial in* $\log m$*, in the encoding length of f and of P, in the maximum total degree D of f, and in $\frac{1}{\epsilon}$ for computing a feasible solution $(\mathbf{x}^m_\epsilon, \mathbf{z}^m_\epsilon) \in P \cap \left(\frac{1}{m}\mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2}\right)$ to the grid problem* (12.1.1)*, where f is non-negative on the feasible region, with*

$$f(\mathbf{x}^m_\epsilon, \mathbf{z}^m_\epsilon) \geq (1 - \epsilon) f(\mathbf{x}^m, \mathbf{z}^m), \tag{12.1.2}$$

where $(\mathbf{x}^m, \mathbf{z}^m) \in P \cap \left(\frac{1}{m}\mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2}\right)$ is an optimal solution to (12.1.1).

Proof. We apply Lemma 11.2.4 to the pure integer optimization problem:

$$\begin{aligned} \max \quad & \tilde f(\tilde{\mathbf{x}}, \mathbf{z}) \\ \text{s.t.} \quad & A\tilde{\mathbf{x}} + mB\mathbf{z} \le m\mathbf{b} \\ & \tilde x_i \in \mathbb{Z} && \text{for } i = 1, \dots, d_1, \\ & z_i \in \mathbb{Z} && \text{for } i = 1, \dots, d_2, \end{aligned} \tag{12.1.3}$$

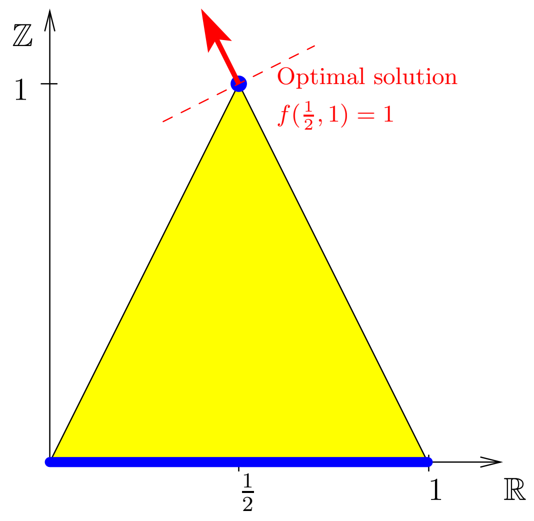
where $\tilde f(\tilde{\mathbf{x}}, \mathbf{z}) := m^D f(\frac{1}{m}\tilde{\mathbf{x}}, \mathbf{z})$ is a polynomial function with integer coefficients. Clearly the binary encoding length of the coefficients of $\tilde f$ increases by at most $\lceil D \log m \rceil$, compared to the coefficients of f. Likewise, the encoding length of the coefficients of mB and $m\mathbf{b}$ increases by at most $\lceil \log m \rceil$. By Theorem 1.1 of [42], there exists an algorithm with running time polynomial in the encoding length of $\tilde f$ and of $A\mathbf{x} + mB\mathbf{z} \le m\mathbf{b}$, the maximum total degree D, and $\frac{1}{\epsilon}$ for computing a feasible solution $(\mathbf{x}^m_\epsilon, \mathbf{z}^m_\epsilon) \in P \cap \left(\frac{1}{m}\mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2}\right)$ such that $\tilde f(\mathbf{x}^m_\epsilon, \mathbf{z}^m_\epsilon) \ge (1-\epsilon)\tilde f(\mathbf{x}^m, \mathbf{z}^m)$, which implies the estimate (12.1.2). □

One might be tempted to think that for large-enough choice of m, we immediately obtain an approximation to the mixed-integer optimum with arbitrary precision. However, this is not true, as the following example demonstrates.

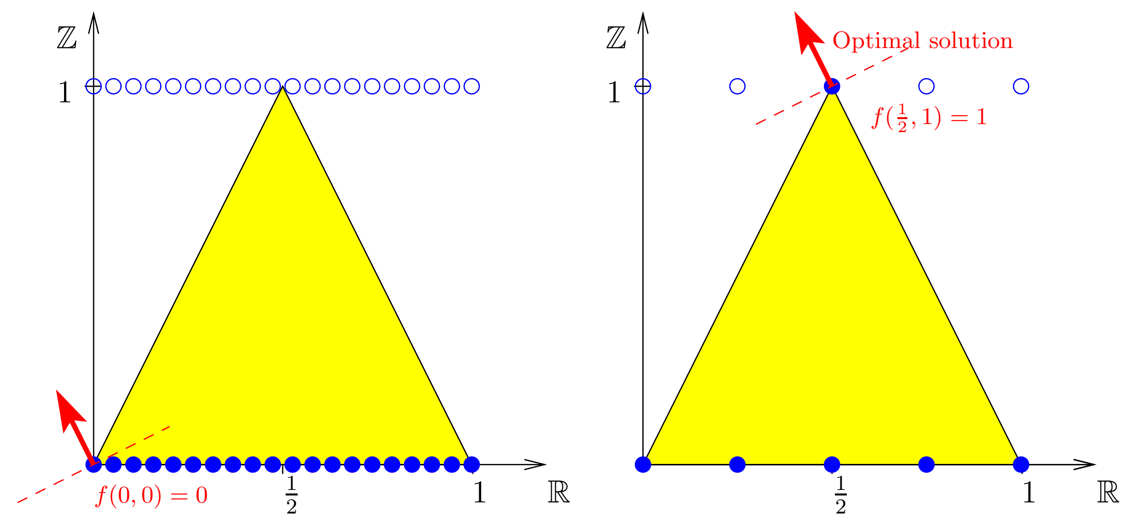
Example 12.1.2 Consider the mixed-integer optimization problem

$$\begin{aligned} \max \quad & 2z - x \\ \text{s.t.} \quad & z \le 2x \\ & z \le 2(1-x) \\ & x \in \mathbb{R}_+, \ z \in \{0, 1\}, \end{aligned} \tag{12.1.4}$$

whose feasible region consists of the point $(\frac{1}{2}, 1)$ and the segment $\{(x, 0) : x \in [0, 1]\}$.



The unique optimal solution to (12.1.4) is $x = \frac{1}{2}$, $z = 1$. Now consider the sequence of grid approximations of (12.1.4) where $x \in \frac{1}{m}\mathbb{Z}_+$.



For even m, the unique optimal solution to the grid approximation is $x = \frac{1}{2}$, $z = 1$. However, for odd m, the unique optimal solution is $x = 0$, $z = 0$. Thus the full sequence of the optimal solutions to the grid approximations does not converge since it has two limit points. □

However we can prove that it is possible to construct, in polynomial time, a subsequence of finer and finer grids that contain a lattice point $(\mathbf{x}, \mathbf{z}^*)$ that is arbitrarily close to the mixed-integer optimum $(\mathbf{x}^*, \mathbf{z}^*)$. This is the central statement of this section and a basic building block of the approximation result.

Theorem 12.1.3 (Grid Approximation) *Let d_1 be fixed. Let*

$$P = \{ (\mathbf{x}, \mathbf{z}) \in \mathbb{R}^{d_1+d_2} : A\mathbf{x} + B\mathbf{z} \le \mathbf{b} \},$$

where $A \in \mathbb{Z}^{p\times d_1}$, $B \in \mathbb{Z}^{p\times d_2}$. Let $M \in \mathbb{R}$ be given such that

$$P \subseteq \{ (\mathbf{x}, \mathbf{z}) \in \mathbb{R}^{d_1+d_2} : |x_i| \le M \text{ for } i = 1, \dots, d_1 \}.$$

There exists a polynomial-time algorithm to compute a number Δ such that for every mixed-integer point $(\mathbf{x}^*, \mathbf{z}^*) \in P \cap (\mathbb{R}^{d_1} \times \mathbb{Z}^{d_2})$ and $\delta > 0$ the following property holds:

> *Every lattice $\frac{1}{m}\mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2}$ for $m = k\Delta$ and $k \ge \frac{1}{\delta}(d_1 + 1)M$ contains a lattice point $(\mathbf{x}, \mathbf{z}^*) \in P \cap \left(\frac{1}{m}\mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2}\right)$ with $\|\mathbf{x} - \mathbf{x}^*\| \le \delta$.*

Theorem 12.1.3 follows directly from the next two lemmas.

Lemma 12.1.4 (Integral Scaling Lemma) *Let $P = \{ (\mathbf{x}, \mathbf{z}) \in \mathbb{R}^{d_1+d_2} : A\mathbf{x} + B\mathbf{z} \leq \mathbf{b} \}$, where $A \in \mathbb{Z}^{p\times d_1}$, $B \in \mathbb{Z}^{p\times d_2}$. For fixed d_1, there exists a polynomial time algorithm to compute a number $\Delta \in \mathbb{Z}_{>0}$ such that for every $\mathbf{z} \in \mathbb{Z}^{d_2}$ the polyhedron*

$$\Delta P_{\mathbf{z}} = \left\{ \Delta\mathbf{x} : (\mathbf{x}, \mathbf{z}) \in P \right\}$$

is integral. In particular, the number Δ has an encoding length that is bounded by a polynomial in the encoding length of P.

Proof. Because the dimension d_1 is fixed, there exist only polynomially many simplex bases of the inequality system $A\mathbf{x} \leq \mathbf{b} - B\mathbf{z}$, and they can be enumerated in polynomial time. The determinant of each simplex basis can be computed in polynomial time. Then Δ can be chosen as the least common multiple of all these determinants. □

Lemma 12.1.5 *Let $Q \subset \mathbb{R}^d$ be an integral polytope, i.e., all vertices have integer coordinates. Let $M \in \mathbb{R}$ be such that $Q \subseteq \{ \mathbf{x} \in \mathbb{R}^d : |x_i| \leq M \text{ for } i = 1, \dots, d \}$. Let $\mathbf{x}^* \in Q$ and let $\delta > 0$. Then every lattice $\frac{1}{k}\mathbb{Z}^d$ for $k \geq \frac{1}{\delta}(d+1)M$ contains a lattice point $\mathbf{x} \in Q \cap \frac{1}{k}\mathbb{Z}^d$ with $\|\mathbf{x} - \mathbf{x}^*\|_\infty \leq \delta$.*

Proof. By Carathéodory's Theorem, there exist $d+1$ vertices $\mathbf{x}^0, \dots, \mathbf{x}^d \in \mathbb{Z}^d$ of Q and convex multipliers $\lambda_0, \dots, \lambda_d$ such that $\mathbf{x}^* = \sum_{i=0}^d \lambda_i \mathbf{x}^i$. Let $\lambda_i' := \frac{1}{k}\lfloor k\lambda_i \rfloor \geq 0$ for $i = 1, \dots, d$ and $\lambda_0' := 1 - \sum_{i=1}^d \lambda_i' \geq 0$. Then $\mathbf{x} := \sum_{i=0}^d \lambda_i'\mathbf{x}^i \in Q \cap \frac{1}{k}\mathbb{Z}^d$, and we have

$$\|\mathbf{x} - \mathbf{x}^*\|_\infty \leq \sum_{i=0}^d (\lambda_i' - \lambda_i)\|\mathbf{x}^i\|_\infty \leq (d+1)\frac{1}{k}M \leq \delta,$$

which proves the lemma. □

12.2 Bounding techniques

Using the results of Section 12.1 we are now able to approximate the mixed-integer optimal point by a point of a suitably fine lattice. The question arises how we can use the geometric distance of these two points to estimate the difference in objective function values. We prove Lemma 12.2.1 that provides us with a local Lipschitz constant for the polynomial to be maximized.

Lemma 12.2.1 (Local Lipschitz constant) *Let f be a polynomial in d variables with maximum total degree D. Let C denote the largest absolute value of a coefficient of f. Then there exists a Lipschitz constant L such that $|f(\mathbf{x}) - f(\mathbf{y})| \leq L\|\mathbf{x} - \mathbf{y}\|_\infty$ for all $|x_i|, |y_i| \leq M$. The constant L is $O(D^{d+1}CM^D)$.*

Proof. Using the usual multi-index notation, let $f(\mathbf{x}) = \sum_{\boldsymbol{\alpha}\in\mathcal{D}} c_{\boldsymbol{\alpha}}\mathbf{x}^{\boldsymbol{\alpha}}$. Let $r = |\mathcal{D}|$ be the number of monomials of f. Then we have

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq \sum_{\boldsymbol{\alpha}\neq\mathbf{0}} |c_{\boldsymbol{\alpha}}|\, |\mathbf{x}^{\boldsymbol{\alpha}} - \mathbf{y}^{\boldsymbol{\alpha}}|.$$

We estimate all summands separately. Let $\boldsymbol{\alpha} \neq \mathbf{0}$ be an exponent vector with $n := \sum_{i=1}^{d} \alpha_i \leq D$. Let

$$\boldsymbol{\alpha} = \boldsymbol{\alpha}^0 \geq \boldsymbol{\alpha}^1 \geq \cdots \geq \boldsymbol{\alpha}^n = \mathbf{0}$$

be a decreasing chain of exponent vectors with $\boldsymbol{\alpha}^{i-1} - \boldsymbol{\alpha}^i = \mathbf{e}^{j_i}$ for $i = 1, \dots, n$. Let $\boldsymbol{\beta}^i := \boldsymbol{\alpha} - \boldsymbol{\alpha}^i$ for $i = 0, \dots, n$. Then $\mathbf{x}^{\boldsymbol{\alpha}} - \mathbf{y}^{\boldsymbol{\alpha}}$ can be expressed as the "telescope sum"

$$\begin{aligned}
\mathbf{x}^{\boldsymbol{\alpha}} - \mathbf{y}^{\boldsymbol{\alpha}} &= \mathbf{x}^{\boldsymbol{\alpha}^0}\mathbf{y}^{\boldsymbol{\beta}^0} - \mathbf{x}^{\boldsymbol{\alpha}^1}\mathbf{y}^{\boldsymbol{\beta}^1} + \mathbf{x}^{\boldsymbol{\alpha}^1}\mathbf{y}^{\boldsymbol{\beta}^1} - \mathbf{x}^{\boldsymbol{\alpha}^2}\mathbf{y}^{\boldsymbol{\beta}^2} \\
&\quad + - \cdots - \mathbf{x}^{\boldsymbol{\alpha}^n}\mathbf{y}^{\boldsymbol{\beta}^n} \\
&= \sum_{i=1}^{n} \left(\mathbf{x}^{\boldsymbol{\alpha}^{i-1}}\mathbf{y}^{\boldsymbol{\beta}^{i-1}} - \mathbf{x}^{\boldsymbol{\alpha}^i}\mathbf{y}^{\boldsymbol{\beta}^i} \right) \\
&= \sum_{i=1}^{n} \left((x_{j_i} - y_{j_i})\mathbf{x}^{\boldsymbol{\alpha}^i}\mathbf{y}^{\boldsymbol{\beta}^{i-1}} \right).
\end{aligned}$$

Since $\left|\mathbf{x}^{\boldsymbol{\alpha}^i}\mathbf{y}^{\boldsymbol{\beta}^{i-1}}\right| \leq M^{n-1}$ and $n \leq D$, we obtain

$$|\mathbf{x}^{\boldsymbol{\alpha}} - \mathbf{y}^{\boldsymbol{\alpha}}| \leq D \cdot \|\mathbf{x} - \mathbf{y}\|_\infty \cdot M^{n-1},$$

thus

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq CrDM^{D-1}\|\mathbf{x} - \mathbf{y}\|_\infty.$$

Let $L := CrDM^{D-1}$. Now, since $r = O(D^d)$, we have $L = O(D^{d+1}CM^D)$. □

Moreover, in order to obtain an FPTAS, we need to put differences of function values in relation to the maximum function value. To do this, we need to deal with the special case of polynomials that are constant on the feasible region; here trivially every feasible solution is optimal. For non-constant polynomials, we can prove a lower bound on the maximum function value. The technique is to bound the difference of the minimum and the maximum function value on the mixed-integer set from below; if the polynomial is non-constant, this implies, for a non-negative polynomial, a lower bound on the maximum function value. We will need a simple fact about the roots of multivariate polynomials.

Lemma 12.2.2 *Let $f \in \mathbb{Q}[x_1, \dots, x_d]$ be a polynomial and let D be the largest power of any variable that appears in f. Then $f = 0$ if and only if f vanishes on the set $\{0, \dots, D\}^d$.*

Proof. This is a simple consequence of the Fundamental Theorem of Algebra. See, for instance, [36, Chapter 1, §1, Exercise 6 b]. □

Lemma 12.2.3 *Let $f \in \mathbb{Q}[x_1, \dots, x_d]$ be a polynomial with maximum total degree D. Let $Q \subset \mathbb{R}^d$ be an integral polytope of dimension $d' \leq d$. Let $k \geq D\,d'$. Then f is constant on Q if and only if f is constant on $Q \cap \frac{1}{k}\mathbb{Z}^d$.*

Proof. Let $\mathbf{x}^0 \in Q \cap \mathbb{Z}^d$ be an arbitrary vertex of Q. There exist vertices $\mathbf{x}^1, \dots, \mathbf{x}^{d'} \in Q \cap \mathbb{Z}^d$ such that the vectors $\mathbf{x}^1 - \mathbf{x}^0, \dots, \mathbf{x}^{d'} - \mathbf{x}^0 \in \mathbb{Z}^d$ are linearly independent. By convexity, Q contains the parallelepiped

$$S := \left\{ \mathbf{x}^0 + \textstyle\sum_{i=1}^{d'} \lambda_i(\mathbf{x}^i - \mathbf{x}^0) : \lambda_i \in [0, \frac{1}{d'}] \text{ for } i = 1, \dots, d' \right\}.$$

We consider the set

$$S_k = \tfrac{1}{k}\mathbb{Z}^d \cap S \supseteq \left\{ \mathbf{x}^0 + \textstyle\sum_{i=1}^{d'} \frac{n_i}{k}(\mathbf{x}^i - \mathbf{x}^0) : n_i \in \{0, 1, \dots, D\} \text{ for } i = 1, \dots, d' \right\}.$$

Now if there exists a $c \in \mathbb{R}$ with $f(\mathbf{x}) = c$ for $\mathbf{x} \in Q \cap \frac{1}{k}\mathbb{Z}^d$, then all the points in S_k are roots of the polynomial $f - c$, which has only maximum total degree D. By Lemma 12.2.2 (after an affine transformation), $f - c$ is zero on the affine hull of S_k; hence f is constant on the polytope Q. □

Theorem 12.2.4 *Let $f \in \mathbb{Z}[x_1, \dots, x_{d_1}, z_1, \dots, z_{d_2}]$. Let P be a rational convex polytope, and let Δ be the number from Lemma 12.1.4. Let $m = k\Delta$ with $k \geq D\,d_1$, $k \in \mathbb{Z}$. Then f is constant on the feasible region $P \cap (\mathbb{R}^{d_1} \times \mathbb{Z}^{d_2})$ if and only if f is constant on $P \cap (\frac{1}{m}\mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2})$. If f is not constant, then*

$$\left| f(\mathbf{x}_{\max}, \mathbf{z}_{\max}) - f(\mathbf{x}_{\min}, \mathbf{z}_{\min}) \right| \geq m^{-D}, \tag{12.2.1}$$

where $(\mathbf{x}_{\max}, \mathbf{z}_{\max})$ is an optimal solution to the maximization problem over the feasible region $P \cap (\mathbb{R}^{d_1} \times \mathbb{Z}^{d_2})$ and $(\mathbf{x}_{\min}, \mathbf{z}_{\min})$ is an optimal solution to the minimization problem.

Proof. Let f be constant on $P \cap (\frac{1}{m}\mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2})$. For fixed integer part $\mathbf{z} \in \mathbb{Z}^{d_2}$, we consider the polytope $\Delta P_{\mathbf{z}} = \{ \Delta\mathbf{x} : (\mathbf{x}, \mathbf{z}) \in P \}$, which is a slice of P scaled to become an integral polytope. By applying Lemma 12.2.3 with $k = (D+1)d$ on every polytope $\Delta P_{\mathbf{z}}$, we obtain that f is constant on every slice $P_{\mathbf{z}}$. Because f is also constant on the set $P \cap (\frac{1}{m}\mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2})$, which contains a point of every non-empty slice $P_{\mathbf{z}}$, it follows that f is constant on P.

If f is not constant, there exist $(\mathbf{x}^1, \mathbf{z}^1)$, $(\mathbf{x}^2, \mathbf{z}^2) \in P \cap (\frac{1}{m}\mathbb{Z}^{d_1} \times \mathbb{Z}^{d_2})$ with $f(\mathbf{x}^1, \mathbf{z}^1) \neq f(\mathbf{x}^2, \mathbf{z}^2)$. By the integrality of all coefficients of f, we obtain the estimate

$$|f(\mathbf{x}^1, \mathbf{z}^1) - f(\mathbf{x}^2, \mathbf{z}^2)| \geq m^{-D}.$$

Because $(\mathbf{x}^1, \mathbf{z}^1)$, $(\mathbf{x}^2, \mathbf{z}^2)$ are both feasible solutions to the maximization problem and the minimization problem, this implies (12.2.1). □

12.3 Proof of Theorem 12.0.1

Now we are in the position to prove the main result.

Part (a). Let $(\mathbf{x}^*, \mathbf{z}^*)$ denote an optimal solution to the mixed-integer problem (12.0.1). Let $\epsilon > 0$. We show that, in time polynomial in the input length, the maximum total degree, and $\frac{1}{\epsilon}$, we can compute a point $(\mathbf{x}, \mathbf{z})$ that satisfies (12.0.1b–12.0.1d) such that

$$|f(\mathbf{x}, \mathbf{z}) - f(\mathbf{x}^*, \mathbf{z}^*)| \leq \epsilon f(\mathbf{x}^*, \mathbf{z}^*). \tag{12.3.1}$$

First we note that we can restrict ourselves to the case of polynomials with integer coefficients, simply by multiplying f with the least common multiple of all denominators of the coefficients. We next establish a lower bound on $f(\mathbf{x}^*,\mathbf{z}^*)$. To this end, let Δ be the integer from Lemma 12.1.4, which can be computed in polynomial time. By Theorem 12.2.4 with $m = D\,d_1\Delta$, either f is constant on the feasible region, or

$$f(\mathbf{x}^*,\mathbf{z}^*) \geq (D\,d_1\Delta)^{-D}, \tag{12.3.2}$$

where D is the maximum total degree of f. Now let

$$\delta := \frac{\epsilon}{2(Dd_1\Delta)^D L(C,D,M)} \tag{12.3.3}$$

and let

$$m := \Delta\left\lceil \frac{2}{\epsilon}(Dd_1\Delta)^D L(C,D,M)(d_1+1)M \right\rceil, \tag{12.3.4}$$

where $L(C,D,M)$ is the Lipschitz constant from Lemma 12.2.1. Then we have $m \geq \Delta\frac{1}{\delta}(d_1+1)M$, so by Theorem 12.1.3, there is a point $(\lfloor\mathbf{x}^*\rceil_\delta,\mathbf{z}^*) \in P \cap \left(\frac{1}{m}\mathbb{Z}^{d_1}\times\mathbb{Z}^{d_2}\right)$ with $\left\|\lfloor\mathbf{x}^*\rceil_\delta - \mathbf{x}^*\right\|_\infty \leq \delta$. Let $(\mathbf{x}^m,\mathbf{z}^m)$ denote an optimal solution to the grid problem (12.1.1). Because $(\lfloor\mathbf{x}^*\rceil_\delta,\mathbf{z}^*)$ is a feasible solution to the grid problem (12.1.1), we have

$$f(\lfloor\mathbf{x}^*\rceil_\delta,\mathbf{z}^*) \leq f(\mathbf{x}^m,\mathbf{z}^m) \leq f(\mathbf{x}^*,\mathbf{z}^*). \tag{12.3.5}$$

Now we can estimate

$$\begin{aligned}
\left|f(\mathbf{x}^*,\mathbf{z}^*) - f(\mathbf{x}^m,\mathbf{z}^m)\right| &\leq \left|f(\mathbf{x}^*,\mathbf{z}^*) - f(\lfloor\mathbf{x}^*\rceil_\delta,\mathbf{z}^*)\right| \\
&\leq L(C,D,M)\left\|\mathbf{x}^* - \lfloor\mathbf{x}^*\rceil_\delta\right\|_\infty \\
&\leq L(C,D,M)\,\delta \\
&= \frac{\epsilon}{2}(D\,d_1\Delta)^{-D} \\
&\leq \frac{\epsilon}{2}f(\mathbf{x}^*,\mathbf{z}^*),
\end{aligned} \tag{12.3.6}$$

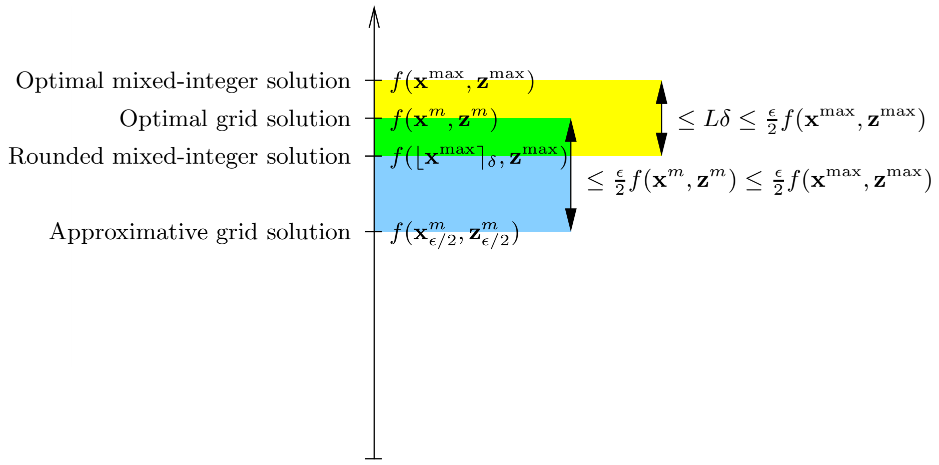
where the last estimate is given by (12.3.2) in the case that f is not constant on the feasible region. On the other hand, if f is constant, the estimate (12.3.6) holds trivially.

By Corollary 12.1.1 we can compute a point $(\mathbf{x}^m_{\epsilon/2},\mathbf{z}^m_{\epsilon/2}) \in P \cap \left(\frac{1}{m}\mathbb{Z}^{d_1}\times\mathbb{Z}^{d_2}\right)$ such that

$$(1-\tfrac{\epsilon}{2})f(\mathbf{x}^m,\mathbf{z}^m) \leq f(\mathbf{x}^m_{\epsilon/2},\mathbf{z}^m_{\epsilon/2}) \leq f(\mathbf{x}^m,\mathbf{z}^m) \tag{12.3.7}$$

in time polynomial in $\log m$, the encoding length of f and P, the maximum total degree D, and $1/\epsilon$. Here $\log m$ is bounded by a polynomial in $\log M$, D and $\log C$, so we can compute $(\mathbf{x}^m_{\epsilon/2},\mathbf{z}^m_{\epsilon/2})$ in time polynomial in the input size, the maximum total degree D, and $1/\epsilon$. Now we can estimate, using (12.3.7) and (12.3.6),

$$\begin{aligned}
&f(\mathbf{x}^*,\mathbf{z}^*) - f(\mathbf{x}^m_{\epsilon/2},\mathbf{z}^m_{\epsilon/2}) \\
&\quad\leq f(\mathbf{x}^*,\mathbf{z}^*) - (1-\tfrac{\epsilon}{2})f(\mathbf{x}^m,\mathbf{z}^m) \\
&\quad= \tfrac{\epsilon}{2}f(\mathbf{x}^*,\mathbf{z}^*) + (1-\tfrac{\epsilon}{2})\big(f(\mathbf{x}^*,\mathbf{z}^*) - f(\mathbf{x}^m,\mathbf{z}^m)\big) \\
&\quad\leq \tfrac{\epsilon}{2}f(\mathbf{x}^*,\mathbf{z}^*) + \tfrac{\epsilon}{2}f(\mathbf{x}^*,\mathbf{z}^*) \\
&\quad= \epsilon f(\mathbf{x}^*,\mathbf{z}^*).
\end{aligned}$$


Optimal mixed-integer solution
$f(\mathbf{x}^{\max}, \mathbf{z}^{\max})$
Optimal grid solution
$f(\mathbf{x}^{m}, \mathbf{z}^{m})$
Rounded mixed-integer solution
$f(\lfloor \mathbf{x}^{\max} \rceil_\delta, \mathbf{z}^{\max})$
Approximative grid solution
$f(\mathbf{x}^{m}_{\epsilon/2}, \mathbf{z}^{m}_{\epsilon/2})$
$\leq L\delta \leq \frac{\epsilon}{2} f(\mathbf{x}^{\max}, \mathbf{z}^{\max})$
$\leq \frac{\epsilon}{2} f(\mathbf{x}^{m}, \mathbf{z}^{m}) \leq \frac{\epsilon}{2} f(\mathbf{x}^{\max}, \mathbf{z}^{\max})$

Hence $f(\mathbf{x}^m_{\epsilon/2}, \mathbf{z}^m_{\epsilon/2}) \geq (1-\epsilon) f(\mathbf{x}^*, \mathbf{z}^*)$.

Part (b). Let the dimension $d \geq 2$ be fixed. We prove that there does not exist a PTAS for the maximization of arbitrary polynomials over mixed-integer sets of polytopes. We use the NP-complete problem AN1 on page 249 of [62]. This is to decide whether, given three positive integers a, b, c, there exists a positive integer $x < c$ such that $x^2 \equiv a \pmod{b}$. This problem is equivalent to asking whether the maximum of the quartic polynomial function $f(x,y) = -(x^2 - a - by)^2$ over the lattice points of the rectangle

$$P = \left\{ (x,y) : 1 \leq x \leq c-1,\ \frac{1-a}{b} \leq y \leq \frac{(c-1)^2 - a}{b} \right\}$$

is zero or not. If there existed a PTAS for the maximization of arbitrary polynomials over mixed-integer sets of polytopes, we could, for any fixed $0 < \epsilon < 1$, compute in polynomial time a solution $(x_\epsilon, y_\epsilon) \in P \cap \mathbb{Z}^2$ with $\left|f(x_\epsilon, y_\epsilon) - f(x^*, y^*)\right| \leq \epsilon \left|f(x^*, y^*)\right|$, where (x^*, y^*) denotes an optimal solution. Thus, we have $f(x_\epsilon, y_\epsilon) = 0$ if and only if $f(x^*, y^*) = 0$; this means we could solve the problem AN1 in polynomial time. $\square$

Chapter 13

Short rational functions for toric algebra

In this chapter we present polynomial-time algorithms for computing with toric ideals. Our results are a direct application of recent results by Barvinok and Woods [15] on short encodings of rational generating functions.

Let $A \in \mathbb{Z}^{d\times n}$ and $b \in \mathbb{Z}^d$ such that the convex polyhedron $P = \{u \in \mathbb{R}^n : Au = b, u \geq 0\}$ is bounded. Barvinok gave an algorithm for counting the lattice points in P in polynomial time when $n - d$ is a constant, see Chapter 9 for a detailed description. The input for Barvinok's algorithm is the binary encoding of the integers a_{ij} and b_i, and the output is a formula for the multivariate generating function $f(P) = \sum_{a\in P\cap\mathbb{Z}^n} x^a$. This long polynomial with exponentially many monomials is encoded as a much shorter sum of rational functions of the form

$$f(P;z) = \sum_{i\in I} \pm \frac{x^{u_i}}{(1-x^{c_{1,i}})(1-x^{c_{2,i}})\dots(1-x^{c_{n-d,i}})}. \tag{13.0.1}$$

Barvinok and Woods [15] developed a set of powerful manipulation rules for using these short rational functions in Boolean constructions on various sets of lattice points. In this chapter we apply their techniques to problems in combinatorial commutative algebra. Our first theorem concerns the computation of the *toric ideal* I_A of the matrix A. Remember that this ideal is generated by all binomials $x^u - x^v$ such that $Au = Av$. In what follows, we encode any set of binomials $x^u - x^v$ in n variables as the formal sum of the corresponding monomials $y^u z^v$ in $2n$ variables $y_1, \dots, y_n, z_1, \dots, z_n$.

Theorem 13.0.1 *Let $A \in \mathbb{Z}^{d\times n}$. Assuming that n and d are fixed, there is a polynomial time algorithm to compute a short rational function G which represents the reduced Gröbner basis of the toric ideal I_A with respect to any given term order $\prec$. Given G and any monomial x^a, the following tasks can be performed in polynomial time:*

1. *Decide whether x^a is in normal form with respect to G.*

2. *Perform one step of the division algorithm modulo G.*

3. *Compute the normal form of x^a modulo the Gröbner basis G.*

The proof of Theorem 13.0.1 will be given in Section 13.1. Special attention will be paid to the Projection Theorem [15, Theorem 1.7] since projection of short rational functions is the most difficult step to implement. Its practical efficiency has yet to be investigated. Our proof of Theorem 13.0.1 does use the Projection Theorem, but our Proposition 13.1.5 in Section 13.1 shows that a *non-reduced* Gröbner basis can be computed in polynomial time without using the Projection Theorem.

13.1 Encoding Gröbner bases of toric ideals

Barvinok and Woods [15] showed:

Lemma 13.1.1 (Theorem 3.6 in [15]) *Let S_1, S_2 be finite subsets of $\mathbb{Z}^n$, for n fixed. Let $f(S_1; z)$ and $f(S_2; z)$ be their generating functions, given as short rational functions with at most k binomials in each denominator. Then there exist a polynomial time algorithm, which, given $f(S_i; z)$, computes*

$$f(S_1 \cap S_2; z) = \sum_{i \in I} \gamma_i \cdot \frac{z^{u_i}}{(1 - z^{v_{i1}}) \dots (1 - z^{v_{is}})}$$

with $s \leq 2k$, where the γ_i are rational numbers, and u_i, v_{ij} nonzero integers.

We will use this *Intersection Lemma* to extract special monomials present in the expansion of a generating function. The essential step in the intersection algorithm is the use of the *Hadamard product* [15, Definition 3.2] and a special monomial substitution. The Hadamard product is a bilinear operation on rational functions (we denote it by $*$). The computation is carried out for pairs of summands as in (13.0.1). Note that the Hadamard product $m_1 * m_2$ of two monomials m_1, m_2 is zero unless $m_1 = m_2$. We present an example of computing intersections.

Example 13.1.2 Let $S_i = \{x \in \mathbb{R} : i - 2 \leq x \leq i\} \cap \mathbb{Z}$ for $i = 1, 2$. We rewrite their rational generating functions as in the proof of Theorem 3.6 in [15]:

$$\begin{aligned} f(S_1; z) &= \frac{z^{-1}}{(1-z)} + \frac{z}{(1-z^{-1})} = \frac{-z^{-2}}{(1-z^{-1})} + \frac{z}{(1-z^{-1})} = g_{11} + g_{12}, \text{ and} \\ f(S_2; z) &= \frac{1}{(1-z)} + \frac{z^2}{(1-z^{-1})} = \frac{-z^{-1}}{(1-z^{-1})} + \frac{z^2}{(1-z^{-1})} = g_{21} + g_{22}. \end{aligned}$$

We need to compute four Hadamard products between rational functions whose denominators are products of binomials and denominators are monomials. Lemma 3.4 in [15] says that, for our example, these Hadamard products are essentially the same as computing the functions (13.0.1) of the auxiliary polyhedron $\{(\epsilon_1, \epsilon_2) : p_1 + a_1\epsilon_1 = p_2 + a_2\epsilon_2, \epsilon_i \geq 0\}$ where p_1, p_2 are the exponents of numerators of the g_{ij}'s involved and a_1, a_2 are the exponents of the binomial denominators. For example, the Hadamard product $g_{11} * g_{22}$ corresponds to the polyhedron

$$\{(\epsilon_1, \epsilon_2) : \epsilon_2 = 4 + \epsilon_1, \epsilon_i \geq 0\}.$$

The contribution of this half line is $-\frac{z^{-2}}{(1-z^{-1})}$. We find

$$\begin{aligned} f(S_1; z) * f(S_2; z) &= g_{11} * g_{21} + g_{12} * g_{22} + g_{12} * g_{21} + g_{11} * g_{22} \\ &= \frac{z^{-2}}{(1-z^{-1})} + \frac{z}{(1-z^{-1})} - \frac{z^{-1}}{(1-z^{-1})} - \frac{z^{-2}}{(1-z^{-1})} \\ &= \frac{z - z^{-1}}{1 - z^{-1}} = 1 + z = f(S_1 \cap S_2; z). \end{aligned}$$

□

Another key subroutine introduced by Barvinok and Woods is the following *Projection Theorem*. In both Lemmas 13.1.1 and 13.1.3, the dimension n is assumed to be fixed.

Lemma 13.1.3 (Theorem 1.7 in [15]) *Assume the dimension n is a fixed constant. Consider a rational polytope $P \subset \mathbb{R}^n$ and a linear map $T : \mathbb{Z}^n \to \mathbb{Z}^k$. There is a polynomial time algorithm which computes a short representation of the generating function $f(T(P \cap \mathbb{Z}^n); z)$.*

We represent a term order $\prec$ on monomials in $x_1, \ldots, x_n$ by an integral $n \times n$-matrix W as in [96]. Two monomials satisfy $x^\alpha \prec x^\beta$ if and only if $W\alpha$ is lexicographically smaller than $W\beta$. In other words, if $w_1, \ldots, w_n$ denote the rows of W, there is some $j \in \{1, \ldots, n\}$ such that $w_i^\intercal \alpha = w_i^\intercal \beta$ for $i < j$, and $w_j^\intercal \alpha < w_j^\intercal \beta$. For example, $W = I_n$ describes the lexicographic term ordering. In what follows, we will denote by $\prec_W$ the term ordering defined by W.

Lemma 13.1.4 *Let $S \subset \mathbb{Z}_+^n$ be finite. Suppose the polynomial $f(S; z) = \sum_{\beta \in S} z^\beta$ is represented as a short rational function and let $\prec_W$ be a term order. We can extract the (unique) leading monomial of $f(S; z)$ with respect to $\prec_W$, in polynomial time.*

Proof. The term order $\prec_W$ is represented by an integer matrix W. For each of the rows w_j of W we perform a monomial substitution $z_i := z_i' t_j^{w_{ji}}$. Such a monomial substitution can be computed in polynomial time by [15, Theorem 2.6]. The effect is that the polynomial $f(S; z)$ gets replaced by a polynomial in the t and the $z's$. After each substitution we determine the degree in t. This is done as follows: We want to do calculations in univariate polynomials since this is faster so we consider the polynomial $g(t) = f(S, \mathbf{1}, t)$, where all variables except t are set to the constant one. Clearly the degree of $g(t)$ in t is the same as the degree of $f(S; z', t)$. We create the *interval polynomial* $i_{[p,q]}(t) = \sum_{i=p}^{q} t^i$ which obviously has a short rational function representation. Compute the Hadamard product of and $i_{[p,q]}$ with $g(t)$. This yields those monomials whose degree in the variable t lies between p and q. We will keep shrinking the interval $[p, q]$ until we find the degree. We need a bound for the degree in t of $g(t)$ to start a binary search. A polynomial upper bound U can be found via the estimate in Theorem 3.1 of [91] by easy manipulation of the numerator and denominator of the fractions in $g(t)$. In no more than $\log(U)$ steps one can determine the degree in t of $f(S; z, t)$ by using a standard binary search algorithm.

Once the degree r in t is known, we compute the Hadamard product of $f(S; z, t)$ and $i_{[r,r]}$, and then compute the limit as t approaches 1. This can be done in polynomial time using residue techniques. The limit represents the subseries $H(S; z) = \sum_{\beta : \beta^\intercal w_j = r} z^\beta$. Repeat the monomial and highest degree search for the row w_{j+1}, w_{j+2}, etc. Since $\prec_W$ is a term order, after doing this n times we will have only one single monomial left, the desired leading monomial. □

Proposition 13.1.5 *Let $A \in \mathbb{Z}^{d\times n}$, $W \in \mathbb{Z}^{n\times n}$ specifying a term order $\prec_W$, and assume that d and n are fixed.*

1) There is a polynomial time algorithm to compute a short rational function G which represents a universal Gröbner basis of I_A.

2) Given the term order $\prec_W$ and a short rational function encoding a (possibly infinite) set of binomials $\sum y^u z^v$, one can compute in polynomial time a short rational function encoding only those binomials such that $x^v \prec_W x^u$.

3) Suppose we are given a sum of short rational functions $f(z)$ which is identical, in a monomial expansion, to a single monomial z^a. Then in polynomial time we can recover the (unique) exponent vector a.

Proof. 1) Denote by w_i the i-th row of the matrix W which specifies the term order. Now set $M = (d+1)(n-d)D(A)$ where $D(A)$ is the largest absolute value of any $d \times d$-subdeterminant of A. Using Barvinok's algorithm, we compute the following generating function in $2n$ variables:

$$G(y, z) = \sum \{y^u z^v : Au = Av, 0 \le u_i, v_i \le M\}.$$

This is the sum over all lattice points in a rational polytope. Lemma 4.1 and Theorem 4.7 in Chapter 4 of [114] imply that the toric ideal I_A is generated by the finite set of binomials $x^u - x^v$ corresponding to the terms $y^u z^v$ in $G(y, z)$. Moreover, these binomials are a universal Gröbner basis of I_A.

2) Suppose we are given a short rational generating function $G_0(y, z) = \sum y^u z^v$ representing a set of binomials $x^u - x^v$ in I_A, for instance $G_0 = G$ in part (1). In the following steps, we will alter the series so that a term $y^u z^v$ gets removed whenever u is not bigger than v in the term order $\prec_W$. Starting with $H_0 = G_0$, we perform Hadamard products with short rational functions $f(S; y, z)$ for $S \subset \mathbb{Z}^{2n}$.

Set $H_i = H_{i-1} * f(\{(u, v) : w_i^\intercal u = w_i^\intercal v\})$, and $G_i = H_{i-1} * f(\{(u, v) : w_i^\intercal u \ge w_i^\intercal v + 1\})$. All monomials $y^u z^v \in G_j$ have the property that $w_i^\intercal u = w_i^\intercal v$ for $i < j$, $w_j^\intercal u > w_j^\intercal v$, and thus $v \prec_W u$. On the other hand, if $v \prec_W u$ then there is some j such that $w_i^\intercal u = w_i^\intercal v$ for $i < j$, $w_j^\intercal u > w_j^\intercal v$, and we can conclude that $y^u z^v \in G_j$. This proves that $G = G_1 \cup G_2 \cup \ldots \cup G_n$ encodes exactly those binomials in G_0 that are correctly ordered with respect to $\prec_W$. We have proved our claim since all of the above constructions can be done in polynomial time.

3) Given $f(z)$ we can compute in polynomial time the partial derivative $\partial f(z)/\partial z_i$. This puts the exponent of z_i as a coefficient of the unique monomial. To compute the derivative can be done in polynomial time by the quotient and product derivative rules. Each time we differentiate a short rational function of the form

$$\frac{z^{b_i}}{(1-z^{c_{1,i}})(1-z^{c_{2,i}})\dots(1-z^{c_{d,i}})}$$

we add polynomially many (binomial) factors to the numerator. The factors in the numerators should be expanded into monomials to have again summands in short rational canonical form $\frac{z^{b_i}}{(1-z^{c_{1,i}})(1-z^{c_{2,i}})\dots(1-z^{c_{d,i}})}$. Note that at most 2^d monomials appear (a constant) each time. Finally, if we take the limit when all variables z_i go to one we will get the desired exponent. □

13.2 Proof of Theorem 13.0.1

Proposition 13.1.5 gives a Gröbner basis for the toric ideal I_A in polynomial time. We now show how to get the reduced Gröbner basis.

Step 1. Compute the generating function which encodes all binomials in I_A:

$$f(y,z) = \sum \{y^u z^v : Au = Av, u, v \geq 0\}.$$

This computation is similar to part 1 of Proposition 13.1.5 except that there is no upper bound M. Next we wish to remove from $f(y,z)$ all incorrectly ordered binomials (i.e. those monomials $y^u z^v$ with $u \prec_W v$ instead of the other way around). We do this following part 2 of Proposition 13.1.5. Abusing notation let us still call $f(y,z)$ the resulting sum of rational functions. Let now $g(y)$ be the projection of $f(y,z)$ onto the first group of variables. Thus $g(y)$ is the sum over all non-standard monomials, and it can be computed in polynomial time by Lemma 13.1.3.

Step 2. Write $\frac{1}{1-y} = \prod_{i=1}^{n} \frac{1}{1-y_i}$ for the generating function of all y-monomials. We compute the following Hadamard product of n rational functions in y:

$$\left(\frac{1}{1-y} - y_1 \cdot g(y)\right) * \left(\frac{1}{1-y} - y_2 \cdot g(y)\right) * \dots * \left(\frac{1}{1-y} - y_n \cdot g(y)\right).$$

This is the generating function over those monomials all of whose proper factors are standard monomials modulo the toric ideal I_A.

Step 3. Let $h(y,z)$ denote the ordinary product of the result of Step 2 with

$$\frac{1}{1-z} - g(z) = \sum \{z^v : v \text{ standard monomial modulo } I_A\}.$$

Thus $h(y,z)$ is the sum of all monomials $y^u z^v$ such that x^v is standard and x^u is minimally non-standard. Compute the Hadamard product $G(y,z) := f(y,z) * h(y,z)$. This is a short rational representation of a polynomial, namely, it is the sum over all monomials $y^u z^v$ such that the binomial $x^u - x^v$ is in the reduced Gröbner basis of I_A with respect to W.

We next prove claims 1 and 2. Let $G(y,z)$ be the reduced Gröbner basis of I_A encoded by the rational function above, and let M be the degree bound of Proposition 13.1.5. Let $b(y,z)$ be the rational function representing $\{(u,v) : 0 \leq u \leq a, 0 \leq v \leq M\}$. The Hadamard product $\overline{G}(y,z) = G(y,z) * b(y,z)$ is computable in polynomial time and encodes exactly those binomials in G that can reduce x^a. If $\overline{G}$ is empty then x^a is in normal form already, otherwise we use Lemma 13.1.4 to find an element $(u,v) \in \overline{G}$ and reduce x^a to x^{a-u+v}.

It is worth noting that analytic calculations may now be used as part of algebraic algorithms: Suppose again we wish to decide whether x^a is in reduced normal form or not. Take $G(y,z)$ as before and compute $F(y) = G(y,\mathbf{1})$. This can be done using monomial substitution [15]. Next compute the integral

$$\frac{1}{(2\pi i)^n} \int_{|y_1|=\epsilon_1} \cdots \int_{|y_d|=\epsilon_d} \frac{(y_1^{-a_1} \cdots y_n^{-a_n})F(y)}{(1-y_1)\cdots(1-y_n)}\, \mathrm{d}y.$$

Here $0 < \epsilon_1, \ldots, \epsilon_d < 1$ are different numbers such that we can expand all the $\frac{1}{1-y_k}$ into the power series about 0. It is possible to do a partial fraction decomposition of the integrand into a sum of simple fractions. The integral is a non-negative integer: it is the number of ways that the monomial x^a can be written in terms of the leading monomials of the Gröbner bases G.

We now present the algorithm for claim 3 in Theorem 13.0.1. A curious byproduct of representing Gröbner bases with short rational functions is that the reduction to normal form need not be done by dividing several times anymore:

Step 4. Let $f(y,z)$ and $g(y)$ as above and compute the Hadamard product

$$H(y,z) := f(y,z) * \left(\left(\frac{1}{1-y} \right) \cdot \left(\frac{1}{1-z} - g(z) \right) \right).$$

This is the sum over all monomials $y^u z^v$ where x^v is the normal form of x^u.

Step 5. We use $H(y,z)$ as one would use a traditional Gröbner basis of the ideal I_A. The normal form of a monomial x^a is computed by forming the Hadamard product

$$H(y,z) * \frac{y^a}{1-z}.$$

Since this is strictly speaking a sum of rational functions equal to a single monomial, applying Part 3 of Proposition 13.1.5 concludes the proof of Theorem 13.0.1. $\square$

Chapter 14

Conclusions

In this thesis we dealt with explicit and implicit representations of lattice point sets, with the computation of these representations and with some applications. This thesis demonstrates the enormous algorithmic and theoretical progress in the computation and the applicability of Hilbert bases of rational polyhedral cones, of Graver bases of lattices, of Gröbner and Markov bases of lattice ideals, and of short rational generating functions.

These algorithms have been efficiently implemented by several people into the two software packages `4ti2` and `LattE`. Both codes have been already successfully tested in recent research projects, such as in algebraic statistics, computational biology of combinatorics.

Some impressive results are the computation of the following objects:

- Hilbert basis of magic 6×6 squares: $522,347$ elements
- Homogeneous primitive partition identities for $n = 20$: $1,254,767$ elements
- Markov basis of the toric ideal of $4 \times 4 \times 4$ contingency tables: $145,512$ elements
- Markov bases of toric ideals of phylogenetic trees with $2,048$ variables
- Counting formula for magic 5×5 squares

From a theoretical perspective, we were able to extend the applicability of both Hilbert bases and short rational generating functions to certain classes of nonlinear (mixed-) integer optimization problems over polyhedra. Both results were a bit unexpected as usually linearity of constraints and objective function were used in proofs.

Further practically interesting problem classes, not considered in this thesis, for a suitable explicit or implicit representation of all (mixed-) integer points in polyhedra are for example:

- Design of molecules with specific pharmaceutical properties,
- Decomposition of chemical/biological reactions into elementary reactions,

- Reachability in Petri-nets,
- Decomposition von of networks and network flows.

It is to be expected that the theoretical and algorithmic achievements for the computation and applicability

- of Hilbert bases of rational polyhedral cones,
- of Graver bases of lattices,
- of Gröbner bases and Markov bases of lattice ideals, and
- of short rational generating functions,

will lead to significant speed-ups in the solution of problems arising in research *and* in practice in the near future.

However, since the sizes of these objects increase exponentially with the dimension, one probably needs to incorporate these objects into already practically successful algorithms to achieve a faster solution.

Bibliography

[1] K. Aardal, A. K. Lenstra, and H. W. Lenstra Jr. Hard equality constrained integer knapsacks. In: *Integer Programming and Combinatorial Optimization: 9th International IPCO Conference*, W. J. Cook and A. S. Schulz (eds.), LNCS **2337**, Springer-Verlag, 2002, 350–366.

[2] K. Aardal, C. A. J. Hurkens, and A. K. Lenstra. Solving a linear diophantine equations with lower and upper bounds on the variables. In: *Integer Programming and Combinatorial Optimization, 6th International IPCO conference*, R. E. Bixby, E. A. Boyd, R. Z. Rios-Mercado (eds.), LNCS **1412**, Springer Verlag, 1998, 229–242.

[3] M. Ahmed, J. A. De Loera, and R. Hemmecke. Polyhedral cones of magic cubes and squares. In: *New Directions in Combinatorial Geometry, The Goodman-Pollack Festschrift*, Aronov et al. (eds.), Springer Verlag, 2003, 25–41.

[4] F. Ajili and E. Contejean. Avoiding slack variables in the solving of linear Diophantine equations and inequations. *Theoretical Computer Science* **173** (1997), 183–208.

[5] M. Ajtai. Generating hard instances of lattice problems. In: *Proc. of 28th Annual ACM Symp. on Theory of Computing*, ACM, 1996, 99–108.

[6] D. Alvis and M. Kinyon. Birkhoff's theorem for Panstochastic matrices. *Amer. Math. Monthly* **108** (2001), 28–37.

[7] W. S. Andrews. *Magic squares and cubes.* Second edition, Dover Publications, Inc., New York, 1960.

[8] S. Aoki and A. Takemura. Minimal basis for connected Markov chain over $3\times3\times K$ contingency tables with fixed two-dimensional marginals. *Austr. New Zeal. J. Stat.* **45** (2003), 229–249.

[9] S. Aoki and A. Takemura. The list of indispensable moves of the unique minimal Markov basis for $3 \times 4 \times K$ and $4 \times 4 \times 4$ contingency tables with fixed two-dimensional marginals. METR Technical Report 03-38, 2003.

[10] F. Aurenhammer and R. Klein. Voronoi diagrams. In: *Handbook of Computational Geometry*, J.-R. Sack and J. Urrutia (eds.), Amsterdam, Netherlands: North-Holland, 2000, 201–290.

[11] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics* **65** (1996), 21–46.

[12] M. L. Balinski and F. J. Rispoli. Signature classes of transportation polytopes. *Math. Prog. Ser. A* **60** (1993), 127–144.

[13] A. I. Barvinok. Polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. *Math. of Operations Research* **19** (1994), 769–779.

[14] A. I. Barvinok and J. Pommersheim. An algorithmic theory of lattice points in polyhedra. In: *New Perspectives in Algebraic Combinatorics* (Berkeley, CA, 1996-1997), MSRI Publ. **38**, Cambridge Univ. Press, Cambridge, 1999, 91–147.

[15] A. I. Barvinok and K. Woods. Short rational generating functions for lattice point problems. *J. Amer. Math. Soc.* **16** (2003), 957–979.

[16] A. I. Barvinok. Computing the Ehrhart quasi-polynomial of a rational simplex. e-print, available at `arXiv: math.CO/0504444`, 2005.

[17] M. Beck. Counting lattice points by means of the residue theorem. *Ramanujan Journal* **4** (2000), 299–310.

[18] M. Beck, M. Cohen, J. Cuomo, and P. Gribelyuk. The number of "magic squares" and hypercubes. *American Mathematical Monthly* **110** (2003), 707–717.

[19] D. Bertsimas and R. Weismantel. *Optimization over Integers*. Dynamic Ideas, Belmont Mass, 2005.

[20] A. M. Bigatti. Computation of Hilbert-Poincaré Series. *J. Pure Appl. Algebra* **119** (1997), 237–253.

[21] A. M. Bigatti, R. LaScala, and L. Robbiano. Computing toric ideals. *J. of Symbolic Computation* **27** (1999), 351–365.

[22] R. Bouyekhf and A. E. Moudni. On the analysis of some structural properties of Petri nets. *IEEE Transactions on Systems, Man and Cybernetics* **35** (2005), 784–794.

[23] M. Brion. Points entiers dans les polyèdres convexes. *Ann. Sci. École Norm. Sup.* **21** (1988), 653–663.

[24] W. Bruns and R. Koch. `NORMALIZ`, computing normalizations of affine semigroups. Available from `ftp://ftp.mathematik.uni-osnabrueck.de/pub/osm/kommalg/software/`.

[25] B. Buchberger. A criterion for detecting unnecessary reductions in the construction of Gröbner bases. In: *Proceedings EUROSAM 79*, LNCS **72**, Springer-Verlag, 1979, 3–21.

[26] B. Buchberger. Gröbner bases: an algorithmic method in polynomial ideal theory. In *Recent Trends in Multidimensional Systems Theory*, N. K. Bose (ed.), D. Reidel Publishing Company, Dordrecht, 1985, 184–232.

[27] B. Buchberger. History and basic features of the critical-pair/completion procedure. *J. of Symbolic Computation* **2** (1987), 3–38.

[28] R. E. Burkard, E. Çela, P. M. Pardalos and L. Pitsoulis. The quadratic assignment problem. In: *Handbook of Combinatorial Optimization*, P. P. Pardalos and M. G. C. Resende (eds.), Kluwer Academic Publishers, Dordrecht, 1998, 241–238.

[29] M. Caboara, M. Kreuzer and L. Robbiano. Efficiently computing minimal sets of critical pairs. *J. of Symbolic Computation* **38** (2004), 1169–1190.

[30] `CoCoA` Team. `CoCoA`: a system for doing Computations in Commutative Algebra. Available at `http://cocoa.dima.unige.it`.

[31] C. S. Chan and D. P. Robbins. On the volume of the polytope of doubly stochastic matrices. *Experimental Math.* **8** (1999), 291–300.

[32] S. Collart, M .Kalkbrener, and D. Mall. Converting Bases with the Gröbner walk. *J. of Symbolic Computation* **24** (1997), 456–469.

[33] E. Contejean and H. Devie. An efficient incremental algorithm for solving systems of linear Diophantine equations. *Information and Computation* **113** (1994), 143–172.

[34] P. Conti and C. Traverso. Buchberger algorithm and integer programming. In: *Proceedings AAECC-9, (New Orleans)*, LNCS **539**, Springer-Verlag, 1991, 130–139.

[35] G. Cornuejols, R. Urbaniak, R. Weismantel, and L. Wolsey. Decomposition of integer programs and of generating sets. LNCS **1284**, Springer-Verlag, 1997, 92–103.

[36] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, Algorithms.* Springer-Verlag, 1992.

[37] L. H. Cox. Bounds on entries in 3-dimensional contingency tables. In: *Inference Control in Statistical Databases - From Theory to Practice*. LNCS **2316**, Springer, New York, 2002, 21–33.

[38] L. H. Cox. On properties of multi-dimensional statistical tables. *J. Stat. Plan. Infer.* **117** (2003), 251–273.

[39] E. de Klerk, M. Laurent, and P. A. Parrilo. A PTAS for the minimization of polynomials of fixed degree over the simplex. Manuscript 2004, to appear in *Theoretical Computer Science*.

[40] J. A. De Loera, D. Haws, R. Hemmecke, P. Huggins, B. Sturmfels, and R. Yoshida. Short rational functions for toric algebra and applications. *J. of Symbolic Computation* **38** (2004), 959–973.

[41] J. A. De Loera, D. Haws, R. Hemmecke, P. Huggins, J. Tauzer, and R. Yoshida. A User's Guide for `LattE` v1.1., 2003. Software package `LattE` is available at `http://www.math.ucdavis.edu/∼latte/`

[42] J. A. De Loera, R. Hemmecke, M. Köppe, and R. Weismantel. Integer polynomial optimization in fixed dimension. *Mathematics of Operations Research* **31** (2006), 147–153.

[43] J. A. De Loera, R. Hemmecke, M. Köppe, and R. Weismantel. FPTAS for Mixed-Integer Polynomial Optimization with a Fixed Number of Variables. In: *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms, Miami, FL, January 22-24*, 2006, 743–748.

[44] J. A. De Loera, R. Hemmecke, J. Tauzer, and R. Yoshida: Effective lattice point counting in rational convex polytopes. *J. of Symbolic Computation* **38** (2004), 1273–1302.

[45] J. A. De Loera and S. Onn. The complexity of three-way statistical tables. *SIAM J. Comp.* **33** (2004), 819–836.

[46] J. A. De Loera and S. Onn. All rational polytopes are transportation polytopes and all polytopal integer sets are contingency tables. In: *Proc. 10th Ann. Math. Prog. Soc. Symp. Integ. Prog. Combin. Optim.* (Columbia University, New York), LNCS **3064**, Springer, New York, 2004, 338–351.

[47] J. A. De Loera and S. Onn. Markov bases of three-way tables are arbitrarily complicated. *J. Symbolic Computation* **41** (2006), 173–181.

[48] J. A. De Loera and B. Sturmfels. Algebraic unimodular counting. *Mathematical Programming Ser. B* **96** (2003) 183–203.

[49] P. Diaconis and A. Gangolli. Rectangular arrays with fixed margins. In *Discrete probability and algorithms*, D. Aldous et al. (eds.), IMA Volumes on Mathematics and its Applications **72**, Springer-Verlag, New York, 1995, 15–41.

[50] P. Diaconis and B. Sturmfels. Algebraic algorithms for sampling from conditional distributions. *Annals of Statistics* **26** (1998), 363–397.

[51] G. T. Duncan, S. E. Fienberg, R. Krishnan, R. Padman, and S. F. Roehrig. Disclosure limitation methods and information loss for tabular data. In: *Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies*, P. Doyle et al. (eds.), North-Holland, 2001, 111–134.

[52] M. Dyer and R. Kannan. On Barvinok's algorithm for counting lattice points in fixed dimension. *Math. of Operations Research* **22** (1997), 545–549.

[53] E. Ehrhart. *Polynomes arithmétiques et methode des polyédres en combinatoire*. International Series of Numerical mathematics **35**, Birkhäuser, Basel, 1977.

[54] E. Ehrhart. Sur un probléme de géométrie diophantienne linéaire II. *J. Reine Angew. Math.* **227** (1967), 25–49.

[55] E: Ehrhart. Figures magiques et methode des polyedres. *J. Reine Angew. Math.* **299/300** (1978), 51–63.

[56] N. Eriksson. Toric ideals of homogeneous phylogenetic models. In: *Proc. of the 2004 International Symposium on Symbolic and Algebraic Computation*, ACM Press, 2004, 149–154.

[57] S. E. Fienberg, U. E. Makov, M. M. Meyer, and R. J. Steele. Computing the exact conditional distribution for a multi-way contingency table conditional on its marginal totals. In: *Data Analysis From Statistical Foundations*, A. K. Md. E. Saleh (ed.), Nova Science Publishers, Huntington, NY, 2001, 145–166.

[58] K. Fukuda. `cdd` and `cdd+`. Implemtation of the double description method, available via `http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html`.

[59] K. Fukuda, A. N. Jensen, N. Lauritzen, and R. Thomas. The generic Grobner walk. e-print, available at `arXiv: math.AC/0501345`, 2005.

[60] K. Fukuda and A. Prodon. Double description method revisited. In: *Combinatorics and Computer Science*, LNCS **1120**, Springer-Verlag, 1996, 91–111.

[61] M. Gardner. *Martin Gardner's New mathematical Diversions from Scientific American*. Simon and Schuster, New York, 1966, 162–172.

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

[63] R. Gebauer and H. M. Möller. On an installation of Buchberger's algorithm. *J. of Symbolic Computation* **6** (1988), 275–286.

[64] F. R. Giles and W. R. Pulleyblank. Total dual integrality and integer polyhedra. *Linear Algebra and its Applications* **25** (1979), 191–196.

[65] J. E. Graver. On the foundation of linear and integer programming I. *Mathematical Programming* **9** (1975), 207–226.

[66] G.-M. Greuel, G. Pfister, and H. Schönemann. `Singular` 3.0. A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra, University of Kaiserslautern (2005). `http://www.singular.uni-kl.de`.

[67] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Second edition. Algorithms and Combinatorics **2**, Springer-Verlag, Berlin, 1993.

[68] J. Håstad. Some optimal inapproximability results. In: *Proceedings of the 29th Symposium on the Theory of Computing (STOC)*, ACM, 1997, 1–10.

[69] U. U. Haus, M. Köppe, and R. Weismantel. The integral basis method for integer programming. *Mathematical Methods of Operations Research* **53** (2001), 281–307.

[70] R. Hemmecke, R. Hemmecke, and P. Malkin. `4ti2` Version 1.2–Computation of Hilbert bases, Graver bases, toric Gröbner bases, and more. Available at `http://www.4ti2.de/`, 2005.

[71] R. Hemmecke. *On the Decomposition of Test Sets: Building Blocks, Connection Sets, and Algorithms*. Ph.D. thesis, Universität Duisburg, 2001.

[72] R. Hemmecke. On the positive sum property and the computation of Graver test sets. *Mathematical Programming* **96** (2003), 247–269.

[73] R. Hemmecke. On the computation of Hilbert bases of cones. In: *Mathematical Software, ICMS 2002*, A. M. Cohen, X.-S. Gao, N. Takayama (eds.), World Scientific, 2002, 307–317.

[74] R. Hemmecke. On the computation of Hilbert bases and extreme rays of cones. e-print, available at `arXiv: math.CO/0203105`, 2002.

[75] R. Hemmecke and P. Malkin. Computation of generating sets of lattice ideals. e-print, available at `arXiv: math.CO/0508359`, 2005.

[76] R. Hemmecke. Exploiting Symmetries in the Computation of Graver Bases. e-print, available at `arXiv: math.CO/0410334`, 2004.

[77] S. Hoşten and J. Shapiro. Primary decomposition of lattice basis ideals. *J. of Symbolic Computation* **29** (2000), 625–639.

[78] S. Hoşten and B. Sturmfels. `GRIN`: An implementation of Gröbner bases for integer programming. In *Integer programming and combinatorial optimization*, E. Balas and J. Clausen (eds.), LNCS **920**, Springer-Verlag, 1995, 267–276.

[79] S. Hoşten and S. Sullivant. Finiteness theorems for Markov bases of hierarchical models. Manuscript 2004, to appear in Journal of Combinatorial Theory: Series A.

[80] S. Hoşten and R. Thomas. Gomory integer programs. In *Mathematical Programming Ser. B* **96** (2002), 271–272.

[81] P. M. Huggins. *Lattice point enumeration via rational functions and applications to optimization and statistics*. Senior undergraduate thesis, Department of mathematics, University of California, Davis, 2004.

[82] R. Irving and M. R. Jerrum. Three-dimensional statistical data security problems. *SIAM J. Comput.* **23** (1994), 170–184.

[83] R. G. Jeroslow. There cannot be any algorithm for integer programming with quadratic constraints. *Operations Research* **(**21) (1973), 221–224.

[84] R. G. Jeroslow. Some basis theorems for integral monoids. *Mathematics of Operations Research* **3** (1978), 145–154.

[85] J. P. Jones. Universal Diophantine equation. *J. of Symbolic Logic* **47** (1982), 403–410.

[86] L. Khachiyan and L. Porkolab. Integer optimization in convex semialgebraic sets. *Discrete Comput. Geom.* **23** (2000), 207–224.

[87] A. N. Kirillov. Ubiquity of Kostka Polynomials. In: *Physics and Combinatorics, Proceedings Nagoya 1999*, A.N. Kirillov, A. Tsuchiya and H. Umemura (eds.), World Scientific, 2001.

[88] V. Klee, and C. Witzgall. Facets and vertices of transportation polytopes. In: *Mathematics of the Decision Sciences, Part I*, (Stanford, CA, 1967), AMS, Providence, RI, 1968, 257–282.

[89] K. Kovács, B. Vizvári, M. Riedel, and J. Tóth: Decomposition of the permanganate/oxalic acid overall reation to elementary steps based on algebraic recipes for integer programming theory. *Physical Chemistry, Chemical Physics* **6** (2004), 1236–1242.

[90] T. Y. Lam. *The algebraic Theory of Quadratic Forms*. Reading, Mass., W. A. Benjamin, 1973.

[91] J. B. Lasserre. Integer programming, Barvinok's counting algorithm and Gomory relaxations. *Operations Research Letters* **32** (2003), 133–137.

[92] C. W. Lee. Subdivisions and triangulations of polytopes. In: *Handbook of Discrete and Computational Geometry*, J. E. Goodman and J. O'Rourke (eds.), CRC Press, New York, 1997, 271–290.

[93] H. W. Lenstra Jr. Integer Programming with a fixed number of variables. *Mathematics of Operations Research* **8** (1983), 538–548.

[94] P. A. MacMahon. *Combinatorial Analysis*. Chelsea, 1960, volumes I and II, reprint of 1917 edition.

[95] Y. Matiyasevich. *Hilbert's tenth problem*. The MIT Press, Cambridge, London, 1993.

[96] T. Mora and L. Robbiano. The Gröbner fan of an ideal. *J. Symbolic Computation* **6** (1988), 183–208.

[97] K. Murota, H. Saito, and R. Weismantel. Optimality criterion for a class of nonlinear integer programs. *Operations Research Letters* **32** (2004), 468–472.

[98] A. Nijehuis and H. Wilf. Representations of integers by linear forms in nonnegative integers. *J. Number Theory* **4** (1972), 98–106.

[99] S. Onn and U.G. Rothblum. Convex combinatorial optimization. *Discr. Comp. Geom.* **32** (2004), 549–566.

[100] D. V. Pasechnik. Computing the Hilbert base via the Elliott-MacMahon algorithm. *Theoretical Computer Science* **263** (2001), 37–46.

[101] P. C. Pasles. The lost squares of Dr. Franklin. *Amer. Math. Monthly* **108** (2001), 489–511.

[102] L. Pottier. Euclide's algorithm in dimension n. Research report, ISSAC **96**, ACM Press, 1996.

[103] L. Pottier. Minimal solutions of linear Diophantine systems: bounds and algorithms. In: *Rewriting techniques and applications (Como, 1991)*, LNCS **488**, Springer, Berlin, 1991, 162–173.

[104] M. Queyranne and F. C. R. Spieksma. Approximation algorithms for multi-index transportation problems with decomposable costs. *Discr. App. Math.* **76** (1997), 239–253.

[105] F. Santos and B. Sturmfels. Higher Lawrence configurations. *J. Comb. Theory Ser. A* **103** (2003), 151–164.

[106] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, Chichester, 1986.

[107] J. R. Schmidt and A. Bincer. The Kostant partition function for simple Lie algebras. In: *J. Mathematical Physics* **25** (1984), 2367–2373.

[108] A. Schulz and R. Weismantel. An oracle-polynomial time augmentation algorithm for integer programming. In: *Proc. of the 10th ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, 1999, 967–968.

[109] V. Shoup. `NTL`, A library for doing Number Theory. Available via `http://shoup.net/ntl/`.

[110] R. P. Stanley. *Enumerative Combinatorics*. Volume I, Cambridge, 1997.

[111] R. P. Stanley. *Combinatorics and commutative algebra*. Progress in Mathematics **41**, Birkhäuser Boston, MA, 1983.

[112] R. P. Stanley. *Combinatorics and commutative algebra*. Second edition, Progress in Mathematics **41**, Birkhäuser, Boston, 1996.

[113] R. P. Stanley. Decompositions of rational convex polytopes. *Annals of Discrete Math.* **6** (1980), 333–342.

[114] B. Sturmfels. *Gröbner Bases and Convex Polytopes*. American Mathematical Society, Providence, Rhode Island, 1995.

[115] B. Sturmfels, R. Weismantel, and G. M. Ziegler. Gröbner bases of lattices, corner polyhedra, and integer programming. *Beiträge zur Algebra und Geometrie/Contributions to Algebra and Geometry* **36** (1995), 282–298.

[116] R. R. Thomas. A geometric Buchberger algorithm for integer programming. *Mathematics of Operations Research* **20** (1995), 864–884.

[117] C. Traverso. Hilbert Functions and the Buchberger Algorithm. *J. of Symbolic Computation* **22** (1997), 355–376.

[118] R. Urbaniak, R. Weismantel, and G. M. Ziegler. A variant of the Buchberger algorithm for integer programming. *SIAM J. Discrete Mathematics* **10** (1997), 96–108.

[119] M. Vlach. Conditions for the existence of solutions of the three-dimensional planar transportation problem. *Discr. App. Math.* **13** (1986), 61–78.

[120] R. Weismantel. Test sets of integer programs. *Mathematical Methods of Operations Research* **47** (1998), 1–37.

[121] H. P. Williams. Fourier-Motzkin elimination extension to integer programming problems. *Journal of Combinatorial Theory (A)* **21** (1976), 118–123.

[122] H. P. Williams. A characterization of all feasible solutions to an integer program. *Discrete Applied Mathematics* **5** (1983), 147–155.

[123] H. P. Williams. A duality theorem for linear congruences. *Discrete Applied Mathematics* **7** (1984), 93–103.

[124] R. T. Wilson. Knots with infinitely many incompressible Seifert surfaces. e-print, available at `arXiv: math.GT/0604001`, 2006.

[125] V.A. Yemelichev, M.M. Kovalev, and M.K. Kravtsov. *Polytopes, Graphs and Optimisation*. Cambridge University Press, Cambridge, 1984.

Erklärung

Hiermit versichere ich, Dr. Raymond Hemmecke, daß ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Magdeburg, den 24. April 2006 Raymond Hemmecke