The Maximum Feasible Subsystem Problem and Vertex-Facet Incidences of Polyhedra

vorgelegt von
Dipl.-Math. Marc E. Pfetsch
aus Heidelberg

Von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Berichter: Prof. Dr. Günter M. Ziegler
Prof. Dr. Rolf H. Möhring

Tag der wissenschaftlichen Aussprache: 21. Oktober 2002

Berlin 2002
D 83

Acknowledgements

Thanks go out to many people. First of all, I thank Günter M. Ziegler for making this work possible by offering me a position and giving me the freedom to work on the things I have been interested in. Working with him, both in research and in teaching, has always been a pleasure and I have learned a lot in these four years.

Work on this thesis actually began during my stay at Cornell. I want to thank Lesie E. Trotter, Jr. for his help regarding my "lost" application. I thank him and Edoardo Amaldi for introducing me to the maximum feasible subsystem problem and the fruitful joint work; the result is Chapter 1, parts of Chapter 2, and the motivation for most investigations in this thesis. Thanks to Edoardo for his invitation to Milan and coming to Berlin.

Thanks also go to my Berlin-coauthors: Michael Joswig, Volker Kaibel, and Günter M. Ziegler. The results of Chapter 3 are an outcome of the fruitful atmosphere in the Discrete Geometry Group at the TU Berlin. This also influenced the work on Chapter 4, which is joint work with Volker Kaibel (Mr. Pulitzer). The discussions with him have helped a lot to improve the work and presentation of this thesis; his door and mind have always been open. Michael Joswig actually raised the question of the "right" complexity of the problem considered in Chapter 4 and showed me the work of Ganter (and lots of other things). Thanks!

I also thank Prof. Rolf H. Möhring for his willingness to take the second assessment for this thesis.

Thanks to Niko Witte for his enthusiasm in implementing and testing the algorithms discussed in Chapter 4. Thanks also to Michael Joswig and Ewgenij (Eugen) Gawrilow for `polymake` and to Ewgenij for "`Eugenplot`" and technical support ("purify"!).

I am very grateful to Edoardo Amaldi, Volker Kaibel, Carsten E.M.C. Lange, Julian Pfeifle, Marc "Vincent" Uetz, and Arnold Waßmer for their careful reading of parts of this thesis.

I also want to thank Mr. Neighbor Christoph Eyrich ("the hardest working man in show-biz") for his LaTeX and style/layout support.

And of course, thanks to all the nerds (former and current) of the "Discrete Geometry" and "Combinatorial Optimization and Graph Algorithms" groups for the lively and stimulating atmosphere. Special thanks to all the basketball players (including Lukas). No "Meta-level" in this thesis – I promise.

Berlin, August 2002 Marc Pfetsch

Contents

Introduction

The *maximum feasible subsystem problem*, Max FS for short, was introduced to me in summer 1997. I had just arrived at Cornell University, where I met Edoardo Amaldi and Prof. Leslie Trotter, Jr. Max FS can be stated as follows: Given an infeasible linear inequality system, find a maximum cardinality feasible subsystem.

One reason to take a closer look at this problem is that it has many interesting applications. The two most prominent ones are in linear programming and in machine learning. In the first, one wishes to analyze the infeasibility of a given linear program and resolve it. This often occurs in practice if one uses a modeling software tool and errors occur in the model or data. The idea is that when we remove the complement of a maximum feasible subsystem, the resulting system is feasible. Hence, one hopes for a small subset of constraints that account for the infeasibility.

In the second application, one looks for a linear classifier that distinguishes between two classes of data. Given a test set of points in $\mathbb{R}^d$, each belonging to one of the two classes, the goal is to find a hyperplane separating the two classes as good as possible, i.e., misclassifying as few points as possible. This hyperplane then is likely to classify new data points correctly. In most cases one cannot find a hyperplane which separates the two classes exactly and the problem can easily be viewed as a special case of Max FS. In fact, Max FS has many more interesting applications and the list is still growing.

As a consequence, one wishes to efficiently solve Max FS or find a solution that is as good as possible. However, Max FS is $\mathcal{NP}$-hard (Chakravarti [41]), which makes it unlikely that there exists a polynomial-time algorithm to solve it. In fact, Max FS is hard to approximate (in polynomial time). More precisely, it can be approximated within a factor 2 but it does not admit a polynomial-time approximation scheme unless $\mathcal{P} = \mathcal{NP}$ (Amaldi and Kann [6]).

Nevertheless, one wants to solve Max FS in practice. Indeed, several heuristics have been developed which seem to work well on special instances. Moreover, Parker and Ryan [95] introduced an exact algorithm, which worked successfully on a small test set. This algorithm iteratively solves set covering problems using integer programming methods. The structure of this algo-

rithm suggests a branch-and-cut approach to MAX FS, which is currently the method of choice to solve hard combinatorial optimization problems to optimality. The main idea of a branch-and-cut approach is to study the polytope corresponding to the problem one wants to solve, i.e., the convex hull of all incidence vectors for feasible points of the problem. Then one uses linear programming relaxations of this polytope with additional cutting planes and combines them with a branch-and-bound process.

The following overview of the genesis of this thesis explains its structure.

As mentioned above, it is fundamental to study the 0/1 polytope P_{FS}, which is the convex hull of all incidence vectors of feasible subsystems of a given infeasible linear inequality system Σ. Edoardo Amaldi, Leslie Trotter, and I continued the polyhedral study started by Parker [94]. As every subset of a feasible subsystem Σ is feasible again, the set of feasible subsystems forms a so-called independence system. Many polyhedral results about the corresponding 0/1 polytope P_{IS} of (general) independence systems exist. These results carry over to P_{FS}, since it just a special case. Hence, one strategy for the investigation of P_{FS} is to see what these results yield for P_{FS}.

It turns out that, in order to get an integer programming formulation for MAX FS, one needs to find *Irreducible Inconsistent Subsystems* (IIS), i.e., infeasible subsystems such that every proper subsystem is feasible. It thus makes sense to look at IISs more closely. We found a new geometric characterization of IISs and proved that the problem of finding a minimum cardinality IIS is $\mathcal{NP}$-hard and hard to approximate. We gave a new proof that IISs give rise to so-called rank facets of P_{FS}. Furthermore, we proved that the corresponding separation problem is $\mathcal{NP}$-hard. We also started to tackle the question of recognizing IIS-hypergraphs, i.e., hypergraphs with each node corresponds to an inequality and each hyper-edge corresponds to an IIS. In the beginning, the idea was to construct difficult instances of MAX FS by choosing the "combinatorics" of the IISs first and then trying to find an infeasible system which "realizes" these IISs.

Then, in Summer 1998, after my one year stay in Cornell was over, I moved to Berlin to join the group of Günter M. Ziegler. Here, the connection between infeasible linear inequality systems and polyhedra became clearer. It is provided by a theorem of Gleeson and Ryan [65], which is fundamental for most parts of this thesis. It states that the IISs of an infeasible system are in one-to-one correspondence to the supports of vertices of an alternative polyhedron. Hence, one can use tools for studying polyhedra in order to get results for infeasible linear inequality systems.

In a joint work with my colleagues Volker Kaibel and Michael Joswig, and with Günter M. Ziegler, we could settle the following problem: Given

vertex-facet incidences of a polyhedron P, decide whether P is unbounded or bounded (i.e., is a polytope). The answer can be obtained by computing the Euler characteristic of (the order complex of) a special poset. In particular, this result proves that no unbounded polyhedron can have the same vertex-facet incidences as a polytope and conversely. This problem arose when proving $\mathcal{NP}$-hardness of the above mentioned IIS-hypergraph recognition problem, i.e., the problem to decide whether a given hypergraph is an IIS-hypergraph.

In connection with these results, it also became clear that, in general, one cannot reconstruct the face lattice of a polyhedron, i.e., its set of faces ordered by inclusion, from its vertex-facet incidences. The reason is that we are missing information about unbounded edges, i.e., about the "situation at infinity". Another result is that the structure of polyhedra which have circulant matrices as vertex-facet incidences can be described. We proved that this is equivalent to the following: Every pointed d-polyhedron which is simple (each vertex lies in d facets) and simplicial (each facet contains d vertices) is a simplex or a polygon, if $d \geq 2$. Such polyhedra arise in the context of so-called generalized antiwebs, a large class of structures which induce facets of P_{IS}. This result leads to a characterization of such facets for the special case of P_{FS} and shows that only two very limited types of generalized antiwebs can occur in this context.

After this, together with Volker Kaibel, we found an algorithm that computes (the Hasse diagram of) the face lattice of a polytope P, given its vertex-facet incidences, in time $\mathcal{O}(\min\{n, m\} \cdot \alpha \cdot \varphi)$, where n is the number of vertices, m is the number of facets, α is the number of vertex-facet incidences, and φ is the total number of faces of P. In fact, this algorithm works for arbitrary atomic and coatomic lattices, if the atom-coatom incidences are given. This makes it possible to generate the above mentioned poset for deciding whether a polyhedron is bounded or unbounded. This poset can be turned into an atomic and coatomic lattice. Then its Euler characteristic can be obtained by computing the Möbius function of this lattice.

Besides all this, I implemented a branch-and-cut algorithm for MAX FS. It provides a first step towards a relatively efficient way of obtaining optimal solutions of MAX FS. The focus lies on the separation of facet-defining inequalities arising from IISs and on two other types of cutting planes. One class is a special case of inequalities introduced by Balas and Ng. Furthermore, Gomory cuts were implemented. So far I could neither find other facet-defining inequalities for P_{IS} for which the separation problem could be solved in polynomial time, nor special cutting planes for P_{FS}. These lines have to be left open for future research.

Looking back at the results collected in this thesis, one should add that MAX FS is not only interesting because of its many applications, but also because of its connections to other areas, as pointed out above. In particular, MAX FS provides the motivation to study problems for polyhedra that are interesting on their own right. Moreover, the independence system of feasible subsystems may be an important special case of general independence systems, besides, for instance, matroids or feasible solutions to knapsack problems.

In the above overview, I indicated that the connections between the different parts of the thesis and the tools used arise naturally when studying the MAX FS problem. Moreover, throughout the text a rather broad range of themes is touched, including classical polyhedral approaches in combinatorial optimization, computational complexity theory, polytope and polyhedral theory, combinatorial topology, computational geometry, practical programming issues, and some more. Additionally, the used techniques and tools range from basic observations and constructions to the application of rather advanced theorems, e.g., the universality theorem about 4-polytopes of Richter-Gebert (see Section 2.5). I hope that the unifying theme is nonetheless clear and that this thesis shows the connection between its beautiful results in different areas.

The structure of this thesis reflects the "historical" process I have described above. Roughly put, the first chapter studies MAX FS, giving the motivation for the other chapters, in return incorporating results of these chapters. A more detailed outline of this thesis is as follows:

CHAPTER 1: This chapter deals with the MAX FS problem itself and with irreducible inconsistent subsystems (IISs). We give basic definitions and describe applications and solution methods for MAX FS. We then study IISs, providing an overview of the known results, before establishing a new geometric characterization of IISs. It is then proved that the problem to find an IIS of smallest cardinality is $\mathcal{NP}$-hard and hard to approximate. After that we turn to the P_{FS} polytope. A new geometric proof of the fact that IISs give rise to facets of P_{FS} is given. Then generalized antiwebs are studied and it is characterized when these structures yield facets of P_{FS}, applying results of Chapter 3.

CHAPTER 2: Here we study IIS-hypergraphs. We give an overview of known results and then discuss how to generate an IIS-hypergraph or an IIS-transversal hypergraph if an infeasible linear inequality system is given. In the next section we develop tools to translate between IIS-hypergraphs and vertex-facet incidence hypergraphs of polyhedra. Then the IIS-hypergraph

recognition problem is shown to be $\mathcal{NP}$-hard and finally it is proved that no finite excluded minor characterization of (partial) IIS-hypergraphs can exist.

CHAPTER 3: This chapter investigates vertex-facet incidences of unbounded polyhedra. We study under which conditions one can reconstruct the face lattice of the polyhedron from its vertex-facet incidences and show that in general this is impossible, even if the vertex-facet incidences satisfy several natural restrictions. Nevertheless, we show that it is always possible to distinguish between vertex-facet incidences coming from polytopes and incidences coming from unbounded polyhedra. This can be done by determining the Euler characteristic of an appropriate complex; we then discuss the computational complexity of computing it. Finally, we show that no unbounded simple and simplicial polyhedra exist.

CHAPTER 4: We present an algorithm which can compute the Hasse diagram of the face lattice of a polytope from its vertex-facet incidences in time polynomial in the size of the input and only linearly in the size of the output. We discuss specializations of this algorithm for simple and simplicial polytopes, to compute the k-skeleton, to compute the face lattice without the edges of the Hasse diagram using less working space, and the application to oriented matroids. Finally, we provide computational results of an implementation of this algorithm.

CHAPTER 5: A branch-and-cut implementation for the MAX FS problem is presented. We discuss heuristics to find IISs and the types of cutting planes that are used. We then give computational results on three different sets of test instances: infeasible systems collected in the Netlib library, random inequality systems, and instances arising from classification problems in machine learning.

Throughout this thesis we assume that the reader is familiar with the foundations of polyhedral theory, computational complexity theory, and linear and combinatorial optimization. For polyhedral/polytope theory we refer to the book of Ziegler [115]. For computational complexity theory we refer to the classical book by Garey and Johnson [62] and to Papadimitriou [93]. For linear and combinatorial optimization we refer to the books by Grötschel, Lovász, and Schrijver [69], Nemhauser and Wolsey [90], and Schrijver [107]. We furthermore assume basic knowledge about simplicial complexes, see Ziegler [115, Chapter 8] or Björner [29].

Most pictures of polytopes in this thesis were produced using the software tools `polymake` [63, 64] and `JavaView` [97, 98].

Chapter 1

The Maximum Feasible Subsystem Problem and Irreducible Inconsistent Subsystems

This chapter provides a study of basic properties related to the maximum feasible subsystem problem (Max FS), which is a combinatorial optimization problem and can be stated as follows: Given an infeasible linear inequality system, find a maximum cardinality feasible subsystem.

After introducing basic notions related to Max FS, we give an overview of its applications, solution methods, and provide corresponding references (Section 1.1). We then study Irreducible Inconsistent Subsystems (IIS) of an infeasible linear inequality system, i.e., infeasible subsystems such that every proper subsystem is feasible. IISs are of importance for Max FS, since one has to remove at least one inequality out of each IIS to get a feasible subsystem. In Section 1.2, we first summarize known results about IISs, in particular, the very fundamental one-to-one correspondence between the IISs of an infeasible system and the vertices of an alternative polyhedron (Section 1.2.1). This result provides the basic connection between questions arising in this chapter and the results of Chapter 3, in which we study vertex-facet incidences of polyhedra. Then, in Section 1.2.2, we give a new geometric characterization of IISs ("simplex decomposition theorem") and deal with the algorithmic question of finding a minimum cardinality IIS (Section 1.2.3). We prove that this problem is $\mathcal{NP}$-hard and hard to approximate.

In Section 1.3, the 0/1 polytope P_{FS} corresponding to Max FS is studied, which is the convex hull of all incidence vectors of feasible subsystems of a given infeasible linear inequality system. First, P_{FS} is introduced in the context of independence system polytopes. We then give a new geometric proof of the fact that inequalities arising from IISs are facet-defining and show that the corresponding separation problem is $\mathcal{NP}$-hard (Section 1.3.2). In Section 1.3.3, we consider the specialization of generalized antiweb inequalities, a class of facet-defining inequalities for general independence system polytopes. We show that only very special types of these inequalities can actually arise for P_{FS} and characterize the cases in which they define facets.

This chapter is joint work with Edoardo Amaldi and Leslie E. Trotter, Jr. Most parts appear in [9].

1.1 INTRODUCTION

In this section, we give the basic definitions needed for the first two chapters. We then briefly review many interesting applications of the MAX FS problem. After that, the current knowledge on the computational complexity of MAX FS as well as exact solution methods and heuristics are discussed.

1.1.1 Basic Definitions

Throughout this chapter, let $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ be an *infeasible* linear inequality system, with $A \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$.[1]

We study the following combinatorial optimization problem, which is called the *maximum feasible subsystem* problem.

MAX FS: Given an infeasible system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$, find a feasible subsystem of Σ containing as many inequalities as possible.

A subsystem of Σ is feasible if and only if it does not contain a minimal infeasible subsystem or irreducible inconsistent subsystem:

Definition 1.1. A subsystem Σ' of an infeasible system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ is an *irreducible inconsistent subsystem (IIS)*, if Σ' is infeasible and all of its proper subsystems are feasible.

Denote the ith row of the matrix A by $\boldsymbol{a}^i \in \mathbb{R}^n$, $1 \leq i \leq m$. Throughout the text, we identify the ith inequality of the system Σ (i.e., $\boldsymbol{a}^i\boldsymbol{x} \leq b_i$) with the index i. Hence, an IIS is just a subset of $[m] := \{1, \ldots, m\}$. We denote by $\mathcal{C}(\Sigma) \subset 2^{[m]}$ the set of all IISs of Σ. The number of IISs can grow exponentially with m and n (see Chakravarti [41])[2]. We are also interested in the following problem.

MIN IIS: Given an infeasible system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$, find a minimum cardinality IIS of Σ.

We need the following definition:

Definition 1.2. Let T be a subset of the inequalities of an infeasible system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$. Then T is an *IIS-transversal* or *IIS-cover*, if $T \cap C \neq \varnothing$ for every IIS C of Σ.

[1] To be able to treat such systems in a computer and for the following problems to fit into (classical) complexity theory, we implicitly assume that the coefficients of A and $\boldsymbol{b}$ are finitely represented wherever it is necessary, e.g. that they are rational or real algebraic numbers; the later assumption is no restriction of generality (see Chapter 2, page 38).

[2] This can also be seen from Theorem 1.12, Lemma 1.10, and the fact that there exist polytopes with exponentially many vertices in the number of facets and the dimension (see Section 2.2).

A complementary version of Max FS is: Given Σ as above, delete as few inequalities from Σ such that the resulting system is feasible. Since we have to remove at least one inequality from every IIS to make the system feasible, this amounts to removing the inequalities of an IIS-transversal from Σ. Hence, this complementary version can be formulated as follows:

Min IIS Cover: Given an infeasible system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$, find an IIS-cover of minimum cardinality.

Min IIS Cover has the following integer programming formulation as a set covering problem:

$$\text{Min IIS Cover} \qquad \begin{array}{lll} \text{minimize} & \displaystyle\sum_{i=1}^{m} y_i & \\ \text{s.t.} & \displaystyle\sum_{i \in C} y_i \geq 1 & \forall\, C \in \mathcal{C}(\Sigma) \\ & y_i \in \{0,1\} & 1 \leq i \leq m. \end{array} \tag{1.1}$$

Of course, weighted versions of Max FS and Min IIS Cover are also important. Most of the solution methods presented in Section 1.1.3 below can handle these cases directly or can be modified to do so. The branch-and-cut algorithm discussed in Chapter 5 needs only slight changes to solve weighted Min IIS Cover instances.

Moreover, in view of solving Max FS or Min IIS Cover we can without loss of generality assume that Σ is in inequality form $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$. Inequalities of type "$\geq$" can be reversed easily and strict inequalities can be treated with some care. Bounds on variables (e.g., nonnegativity constraints) can be converted to general inequalities. Moreover, any equation $\boldsymbol{a}\boldsymbol{x} = \beta$ can be substituted by the pair of inequalities $\boldsymbol{a}\boldsymbol{x} \leq \beta$ and $-\boldsymbol{a}\boldsymbol{x} \leq -\beta$. Hence, any generalized Max FS instance I with m_1 equations and m_2 inequalities can obviously be reduced to a Max FS instance I' with $2m_1 + m_2$ inequalities and no equations, in which one aims at maximizing the number of such pairs that can be simultaneously satisfied. Since any vector $\boldsymbol{x}$ satisfies at least one inequality of each pair, an optimal solution of I contains m^* linear relations if and only if an optimal solution of I' contains $m^* + m_1$ inequalities. Thus, from a computational point of view, solving such generalized instances of Max FS with mixed systems is equivalent to solving Max FS for instances of the form $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$. Nevertheless, the branch-and-cut algorithm presented in Chapter 5 directly solves generalized instances of Max FS.

Not all of the results of this chapter, however, can easily be extended to mixed systems. For instance, a generalization of the simplex decomposition characterization (Theorem 1.18) is unknown. Theorem 1.26, about facets

of P_{FS} arising from IISs, is false in the general setting, see Example 1.28. On the other hand, all complexity results obviously carry over to the general case, e.g., for the MIN IIS problem (Theorem 1.21) and the IIS-hypergraph recognition problem (Theorem 2.19). Note that a generalized version of the alternative polyhedron result (Theorem 1.12) for general mixed systems is given in Theorem 1.13.

1.1.2 Applications

Many interesting applications of MAX FS/MIN IIS COVER exist, for instance, in linear programming, telecommunications, machine learning, artificial neural networks, image processing, speech machine translation, and computational biology. In this section, we briefly sketch some of these applications. Of course, we do not claim that this list is complete. Let us refer to the thesis of Amaldi [4] for a more detailed collection of applications.

A well investigated application of MAX FS/MIN IIS COVER is in linear programming. Here, it arises when the LP formulation phase yields infeasible models and one wishes to diagnose and resolve infeasibility. In most situations this cannot be done by inspection and the need for effective algorithmic tools has become more acute since the models have become larger and larger over the last decades. One approach to tackle this problem is to find a set of constraints that is as small as possible such that deleting these constraints results in a feasible system; this is just the MIN IIS COVER problem. The hope is that the number of inequalities that have to be deleted is small and these inequalities capture the cause of the infeasibility well. This application is discussed, for instance, in Greenberg and Murphy [67], and Parker and Ryan [95]. Chinneck [46] gives an overview. Algorithms for it are also mentioned in Section 1.1.3.

An application of MAX FS in telecommunications is described by Rossi, Sassano, and Smriglio [101]. Here, to plan the digital video broadcasting network of Italy, transmitters have to be placed and their emission frequency and power have to be chosen as to maximize the area coverage, subject to quality constraints. A subproblem of this can be modeled as a linear inequality system. Interference of the signals leads to areas where the digital signal cannot be received, resulting in an infeasible system. Maximizing the total weight of satisfied inequalities then amounts to maximize the area coverage.

In machine learning, the application of MAX FS/MIN IIS COVER arises when the goal is to find a linear classifier (a hyperplane) separating two classes of points in $\mathbb{R}^{n-1}$. One wants to minimize the number of misclassified

points in a test set, as to maximize the chances that a new point can be correctly classified. The problem of finding such a classifier can easily be formulated as an instance of Min IIS Cover in $\mathbb{R}^n$, compare Section 5.3.4 of Chapter 5. See Bennett and Bredensteiner [24], Bennett and Mangasarian [25], Greer [68], Mangasarian [83], and Parker [94] for more information and examples.

An application equivalent to finding such a linear classifier arises in artificial neural networks. The building blocks of an artificial neural network are so-called perceptrons. A perceptron has n inputs $x_i \in \mathbb{R}$ and a weight $w_i \in \mathbb{R}$ attached to each of these inputs. The perceptron is determined by the weight vector $(w_1, \ldots, w_n)$ and an additional number $w_0 \in \mathbb{R}$. If an input vector $\boldsymbol{x} \in \mathbb{R}^n$ is given, the output of the perceptron is 1 if $\sum_{i=1}^{n} w_i\, x_i > w_0$ and -1 otherwise. Hence, a perceptron is just a linear classifier and the goal in training a perceptron is to minimize the misclassified input vectors. We refer to Amaldi [3, 4] for more information.

Max FS also appears as a subproblem in a method to solve line detection problems in image processing. Here, one is given a black and white picture with noise and wants to detect line segments. Any line (not going through $\mathbf{0}$) can be defined by an equation $a_1x_1 + a_2x_2 = 1$, where (x_1, x_2) are the parameters of the line we are looking for and (a_1, a_2) is a point on the line. If the matrix A has a row (a_1, a_2) for each black point in the picture, we get a system $A\boldsymbol{x} = \mathbb{1}$, which is usually infeasible. The set of points lying on a common line gives rise to a feasible subsystem. This leads to the Min PFS problem: Given an infeasible system $A\boldsymbol{x} = \boldsymbol{b}$, partition it into a minimum number of feasible subsystems. To solve such a problem in practice, one replaces each equation $\boldsymbol{a}^k\boldsymbol{x} = b_k$ by the two inequalities $\boldsymbol{a}^k\boldsymbol{x} \leq b_k + \varepsilon$ and $\boldsymbol{a}^k\boldsymbol{x} \geq b_k - \varepsilon$ for some $\varepsilon > 0$. A greedy approach to solve Min PFS consists of finding a maximum feasible subsystem, removing it from the system, and iterating. Using relaxation methods as a heuristic for the Max FS problem (see Section 1.1.3) Amaldi and Mattavelli [8] obtain very good results compared to other known approaches; see [8] and the references therein for more information on this application.

An application of Max FS in speech machine translation is discussed by Küssner and Tidhar [79]. In their approach one is given four different speech translation programs and wants to exploit the fact that each of them provides a good translation in different contexts to obtain a good translation as possible. When one of these programs returns a translated sentence, it also outputs a confidence value for the correctness of the translation. Each such value is a number in the interval $[0, 100]$, but values of different programs are not directly comparable to each other. Hence, the goal is to rescale

these confidence values (linearly) such that they become comparable. For a given input sentence one then takes the translation of the program that returned the best rescaled confidence value. The rescaling is performed as follows. Each confidence value $c(p)$ of a translation program p is rescaled to $a(p) \cdot c(p) + b(p)$, where $a(p), b(p) \in \mathbb{R}$ have to be determined. The programs are tested on training data and evaluated by human annotators. For each test sentence, linear inequalities are introduced that imply that the rescaled confidence value of the program with the best performance should be greater than the values of the other programs. This naturally leads to infeasible linear inequality systems and the goal is to find $a(p)$ and $b(p)$ for each program p, such that the number of satisfied inequalities is maximized.

In computational biology an application of MAX FS arises in the context of protein folding. One is given the sequence of amino acid residues that form a given protein and wants to predict its "native" three-dimensional structure (a folding). This is of great interest since the structure crucially determines the chemical properties of a protein. Wagner, Meller, and Elber [114] describe the following approach to protein folding. The structure of the protein is assumed to be determined by an energy potential function $E(s, x)$ for a sequence s and three-dimensional structure (positions of the amino acids) x. One adopts the usual assumption that the protein with sequence s is in its "native" folding structure x^* if and only if $E(s, x^*)$ is minimum over all possible structures x. The energy E is then expressed by appropriate "basis functions" ϕ_i, such that $E(s, x) = \sum_i y_i \, \phi_i(s, x)$, where the y_i are parameters to be determined. For instance, $\phi_i(s, x)$ could measure the number of "contacts" between different amino acid residues that have influence on the energy function. Using a huge database on sequences and a program to generate misfolded structures, one introduces inequalities as to guarantee that $E(s, x)$ is minimized by the native structure. Since often the data contains errors and the model and data are not good enough to identify the "native" structure, this leads to an infeasible linear inequality system with millions of inequalities and with hundreds of variables. Solving a feasible subsystem of maximum cardinality yield the parameters y_i, which are likely to determine the "native" three-dimensional structure of the protein. Wagner, Meller, and Elber use a heuristic to find a large feasible subsystem by repeatedly applying an interior point solver.

1.1.3 Computational Complexity, Exact and Heuristical Solution Methods

MAX FS is strongly $\mathcal{NP}$-hard (compare Chakravarti [41] and Johnson and Preparata [72]). It is strongly $\mathcal{NP}$-hard even when the matrix A has $-1/1$

coefficients only (Amaldi and Kann [6]) and when A is totally unimodular and $\boldsymbol{b}$ is integer (Sankaran [105]). It is solvable in polynomial time, however, if $[A\ \boldsymbol{b}]$ is totally unimodular (see [105]). If the number of variables n is fixed, Max FS can be solved in polynomial time by an $\mathcal{O}(n \cdot m^n/2^{n-1})$ time algorithm of Greer [68]. If the number of constraints m is fixed, one can enumerate all possible subsystems in polynomial time.

Max FS can be approximated within a factor 2, but it does not admit a polynomial-time approximation scheme, unless $\mathcal{P} = \mathcal{NP}$ (see Amaldi and Kann [6]). Although Max FS and Min IIS Cover are equivalent with respect to solving them optimally (and hence Min IIS Cover is $\mathcal{NP}$-hard), the latter is much harder to approximate: Unless $\mathcal{P} = \mathcal{NP}$, Min IIS Cover cannot be approximated within any constant factor. It can, however, be approximated within $n + 1$, where n is the number of variables (Amaldi and Kann [7]). With respect to approximation, different versions of the problem with varying types of the relations (i.e., "$\leq$", "$=$", "$\neq$") behave differently. We refer to Amaldi and Kann [6, 7] for details.

Let DTIME($T(s)$) denote the class of problems whose instances of size s can be solved in deterministic time $\mathcal{O}(T(s))$, where $T : \mathbb{N} \to \mathbb{R}$ is some function, and write $\operatorname{polylog} m$ for any polynomial in $\log m$. The assumption $\mathcal{NP} \not\subseteq \text{DTIME}(s^{\operatorname{polylog} s})$ is stronger than $\mathcal{NP} \neq \mathcal{P}$, but it is also believed to be extremely likely; it amounts to the claim that not all problems in $\mathcal{NP}$ can be solved in quasi-polynomial time. The Min IIS Cover problem cannot be approximated within a factor of $2^{\log^{1-\epsilon} n}$, for any $\epsilon > 0$, unless $\mathcal{NP} \subseteq \text{DTIME}(s^{\operatorname{polylog} s})$, where n is the number of variables. This was proved by Amaldi and Kann [7], applying results of Arora, Babai, Stern, and Sweedyk [10].

An exact algorithm to solve Min IIS Cover, based on a set cover formulation, is proposed in Parker [94] and Parker and Ryan [95]. The algorithm iteratively solves the integer program (1.1) for a partial list of IISs, using an integer programming solver. Then either it is correctly concluded that the optimal cover is found, or at least one uncovered IIS is found and the process is iterated. This approach is explained in more detail in Section 5.1 of Chapter 5. Parker and Ryan tested variants differing in the method to find uncovered IISs on a collection of infeasible LPs available in the Netlib library (see Section 5.3.2 of Chapter 5). These problems are also used as a test set for the heuristics described below.

In the context of linear programming, several heuristics were proposed to solve Min IIS Cover. In order to help the modeler resolve infeasibility of large linear inequality systems, attention was first devoted to the problem of identifying IISs with a small and possibly minimum number of inequalities

(see Greenberg and Murphy [67]), which amounts to solving or approximating MIN IIS. Chinneck [47], Chinneck and Dravnieks [49] proposed and tested several heuristics, which are now available in commercial LP-solvers such as CPLEX and MINOS (see Chinneck [44]). These heuristics are of greedy type.

Clearly, in the presence of many overlapping IISs, this does not provide enough information to repair the original system. Hence, the emphasis shifted to the application of MIN IIS COVER, in hope that a minimum cardinality IIS-cover comprises the essential information about infeasibility of the model. Heuristics and computational results are given in Chinneck [45, 48]. The heuristics are again greedy algorithms.

Many of the mentioned heuristics use so-called *elastic programs*. Here, slack variables are introduced into an infeasible system $A\boldsymbol{x} \leq \boldsymbol{b}$ to obtain the feasible system: $A\boldsymbol{x} - \boldsymbol{s} \leq \boldsymbol{b}$, $\boldsymbol{s} \geq \boldsymbol{0}$. Equations should be relaxed "two-sided". Other cases can be handled similarly. A feasible solution of such a system, e.g., minimizing $\sum_{i=1}^{m} s_i$, can then heuristically be exploited to search for small IIS-covers.

Going one step further, one can formulate MIN IIS COVER as a 0/1-integer program using a "big-M" as follows:

$$\begin{aligned} \text{minimize} \quad & z_1 + z_2 \cdots + z_m \\ \text{s.t.} \quad & A\boldsymbol{x} - \boldsymbol{s} \leq \boldsymbol{b} \\ & \boldsymbol{s} \leq M\boldsymbol{z} \\ & \boldsymbol{s} \geq \boldsymbol{0},\ \boldsymbol{z} \in \{0,1\}^m. \end{aligned}$$

Clearly, the choice of M is delicate: It has to be large enough so that the system is feasible and small enough not to introduce too much numerical instability. See the thesis of Parker [94] for more information. He reports that this approach has problems to even solve medium size instances in the Netlib library (compare the results of Section 5.3.2). Nevertheless, for the application in digital video broadcasting, explained above, the "big-M" formulation works reasonable, outperforming other exact approaches. One reason for this behavior could be that in this application bounds on the variables are not allowed to be removed in order to obtain a feasible solution.

For the application of MIN IIS COVER in machine learning (see Section 1.1.2), several heuristics were proposed which use methods from nonlinear programming (Bennett and Bredensteiner [24]; Bennett and Mangasarian [25]; Mangasarian [83]). Mangasarian [84] introduced an algorithm which by Theorem 1.12 is a heuristic for MIN IIS.

A class of algorithms for solving MAX FS are extensions of the *relaxation method*. Originally, this method was developed by Agmon [2] and Motzkin

and Schoenberg [89] to find a solution of a feasible system $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$. At iteration k, one has a current point $\boldsymbol{z}_k \in \mathbb{R}^n$. One then chooses $i \in [m]$ such that $\boldsymbol{a}^i \boldsymbol{z}_k > b_i$, if this is possible. The new point $\boldsymbol{z}_{k+1}$ is obtained as

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k - \lambda \frac{\boldsymbol{a}^i \boldsymbol{z}_k - b_i}{\|\boldsymbol{a}^i\|^2} \boldsymbol{a}^i,$$

where $\lambda \in (0, 2]$ is a chosen step length. If no violated inequality $\boldsymbol{a}^i \boldsymbol{x} \leq b_i$ is found, the process terminates. After each step, the violation $\boldsymbol{a}^i \boldsymbol{z}_k - b_i$ is decreased, while the violation of other inequalities may have increased.

If $\lambda = 2$ and the system is feasible and full-dimensional, this process yields a feasible solution in a finite number of steps (although exponential in the worst-case), see Chapter 12.3 of Schrijver [107]. If this approach is applied to an infeasible system, it obviously can never terminate or converge. One can, however, decrease the step length λ after every iteration. If this decrease is sufficiently slow, the method converges. The decisions in the algorithm can also be randomized, leading to so-called "probabilistic thermal perceptron algorithms", see Amaldi [4]. Amaldi and Hauser [5] propose more variants, prove convergence under certain circumstances, and report on computational experiments. Such methods rapidly find a solution satisfying many inequalities and are insensitive to numerical instabilities. Relaxation methods, however, have so far only be applied to the case without implicit equations in the system. Nevertheless, many applications of Max FS are formulated with strict inequalities and hence no implicit equations are possible. This, for instance, is the case for the applications in machine learning, protein folding, and digital video broadcasting, presented in Section 1.1.2.

1.2 Foundations of Irreducible Inconsistent Subsystems

The known characterizations of irreducible inconsistent subsystems are based on the following version of the Farkas lemma:

Proposition 1.3. For any system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$, either $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ is feasible or there exists $\boldsymbol{y} \geq \boldsymbol{0}$ such that $\boldsymbol{y}A = \boldsymbol{0}$ and $\boldsymbol{y}\boldsymbol{b} < 0$, but not both.

Throughout this section let $A' \in \mathbb{R}^{k \times n}$ and $\boldsymbol{b}' \in \mathbb{R}^k$; e.g., $\{A'\boldsymbol{x} \leq \boldsymbol{b}'\}$ could be an infeasible subsystem of Σ.

Theorem 1.4 (Motzkin [88]). If the system $\{A'\boldsymbol{x} \leq \boldsymbol{b}'\}$ is an IIS then $\operatorname{rank}(A') = k - 1$.

This immediately has the following consequence.

Corollary 1.5 (Motzkin [88]). Any IIS of Σ has size at most $n+1$.

Combining Proposition 1.3 and Theorem 1.4, we get the following characterization of IISs, which is a strengthened version of a theorem obtained by Fan (see van Loon [113]).

Theorem 1.6 (Fan [55]). The system $\{A'\boldsymbol{x} \leq \boldsymbol{b}'\}$ is an IIS if and only if $\operatorname{rank}(A') = k-1$ and there exists $\boldsymbol{y} \in \mathbb{R}^k$, $\boldsymbol{y} > \boldsymbol{0}$, such that $\boldsymbol{y}A = \boldsymbol{0}$ and $\boldsymbol{y}\boldsymbol{b} < 0$.

Van Loon characterized IISs in terms of simplex tableaus. For notational convenience we let $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ be the candidate for an IIS, where $A \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. For his approach, one introduces slack variables to obtain the system $A\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{b}$, $\boldsymbol{s} \geq \boldsymbol{0}$. For $1 \leq i \leq m$, let A^i be the matrix obtained by deleting row i from A. Similarly, let $\boldsymbol{b}^i$ and $\boldsymbol{s}^i$ be the vectors obtained by deleting the ith component of $\boldsymbol{b}$ and $\boldsymbol{s}$, respectively. Denote by $\boldsymbol{x}_B$ a subset of $m-1$ variables of $\boldsymbol{x}$ and $\boldsymbol{x}_N$ the remaining ones. Let B and N be the submatrices of A^i consisting of the columns corresponding to $\boldsymbol{x}_B$ and $\boldsymbol{x}_N$, respectively.

Theorem 1.7 (Van Loon [113]). The system $A\boldsymbol{x} + \boldsymbol{s} = \boldsymbol{b}$, $\boldsymbol{s} \geq \boldsymbol{0}$ corresponds to an IIS if and only if there exists a slack variable s_i, such that the system can be solved with respect to s_i and $\boldsymbol{x}_B$, i.e., there exists $y_i \in \mathbb{R}$, $\boldsymbol{y} \in \mathbb{R}^{m-1}$, $\boldsymbol{s} \in \mathbb{R}^m$, and $\boldsymbol{x}_B$ with corresponding non-singular B, such that

$$\begin{aligned} &s_i = y_i - \boldsymbol{y}\boldsymbol{s}^i \\ &\boldsymbol{x}_B = B^{-1}\boldsymbol{b}^i - B^{-1}N\boldsymbol{x}_N - B^{-1}\boldsymbol{s}^i \\ &y_i < 0,\ \boldsymbol{y} > \boldsymbol{0}\,. \end{aligned}$$

Van Loon further observes that one can identify an IIS in a larger infeasible system if the simplex tableau of a phase I invocation is available. Moreover, other IISs may be found by pivoting in this tableau. This result provided the foundation for several approaches to use IISs as a way to analyze infeasibility of linear programs (see also Section 1.1.3).

The following result establishes an interesting geometric property of the polyhedra obtained by deleting any inequality from an IIS.

Theorem 1.8 (Motzkin [88]). Let Σ' be an IIS. Then the polyhedron defined by the subsystem of Σ' obtained by removal of an arbitrary inequality is an affine convex cone.

1.2.1 Alternative Polyhedron

There exists a simple – but far reaching – relation between the IISs of Σ and the vertices of the following polyhedron.

Definition 1.9. Define

$$P(\Sigma) := \{\boldsymbol{y} \in \mathbb{R}^m \,:\, \boldsymbol{y}A = \boldsymbol{0},\ \boldsymbol{y}\boldsymbol{b} = -1,\ \boldsymbol{y} \geq \boldsymbol{0}\}$$

to be the *alternative polyhedron* associated to Σ.

Two polyhedra $P \subset \mathbb{R}^p$ and $Q \subset \mathbb{R}^q$ are *affinely equivalent* (which is denoted by $P \cong Q$), if there exists an affine map $\phi : \mathbb{R}^p \to \mathbb{R}^q$ which establishes a one-to-one correspondence between points in P and Q. Any pointed polyhedron Q that is not a cone can be expressed as an alternative polyhedron, which is affinely equivalent to Q.

Lemma 1.10. Let P be a d-dimensional pointed polyhedron with m facets which is not a polyhedral cone. Then there exists $A \in \mathbb{R}^{m\times(m-d-1)}$, $\boldsymbol{b} \in \mathbb{R}^m$, such that the polyhedron

$$P' = \{\boldsymbol{y} \in \mathbb{R}^m \,:\, \boldsymbol{y}A = \boldsymbol{0},\ \boldsymbol{y}\boldsymbol{b} = -1,\ \boldsymbol{y} \geq \boldsymbol{0}\}$$

is affinely equivalent to P, and all inequalities $y_j \geq 0$, $1 \leq j \leq m$, define facets of P'.

Proof. The proof uses standard techniques.

Without loss of generality, we can assume that P is full-dimensional (otherwise find implicit equations by linear programming and use these to eliminate variables). Furthermore, we can assume that P is represented by $\{\boldsymbol{x} \in \mathbb{R}^d \,:\, D\boldsymbol{x} \leq \boldsymbol{d}\}$, where each inequality defines a facet of P (redundant inequalities can again be removed using linear programming). Since P is pointed, D has full rank and hence P can be represented as:

$$\left\{\boldsymbol{x} \in \mathbb{R}^d \;\middle|\; \begin{pmatrix} D_1 \\ D_2 \end{pmatrix} \boldsymbol{x} \leq \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{pmatrix}\right\},$$

where D_1 is a non-singular $d\times d$ matrix, D_2 is an $(m-d)\times d$ matrix, $\boldsymbol{d}_1 \in \mathbb{R}^d$, and $\boldsymbol{d}_2 \in \mathbb{R}^{m-d}$. Apply the affine transformation $\boldsymbol{x} \mapsto D_1^{-1}(\boldsymbol{d}_1 - \boldsymbol{u})$, where $\boldsymbol{u} := \boldsymbol{d}_1 - D_1\boldsymbol{x} \in \mathbb{R}^d$, to obtain:

$$\begin{pmatrix} D_1 \\ D_2 \end{pmatrix} D_1^{-1}(\boldsymbol{d}_1 - \boldsymbol{u}) \leq \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{pmatrix} \;\Leftrightarrow\; \begin{pmatrix} -I \\ -D_2D_1^{-1} \end{pmatrix} \boldsymbol{u} \leq \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{d}_2 - D_2D_1^{-1}\boldsymbol{d}_1 \end{pmatrix}.$$

Setting $\boldsymbol{d}' := \boldsymbol{d}_2 - D_2 D_1^{-1} \boldsymbol{d}_1$ and $D' := -D_2 D_1^{-1} \in \mathbb{R}^{(m-d)\times d}$ gives

$$P \cong \{\boldsymbol{u} \in \mathbb{R}^d \,:\, D'\boldsymbol{u} \leq \boldsymbol{d}',\ \boldsymbol{u} \geq \boldsymbol{0}\}.$$

Clearly, all inequalities define facets. The usual introduction of slack variables $\boldsymbol{s} \in \mathbb{R}^{m-d}$ yields

$$P \cong \{(\boldsymbol{u}, \boldsymbol{s}) \in \mathbb{R}^d \times \mathbb{R}^{m-d} \,:\, D'\boldsymbol{u} + I\boldsymbol{s} = \boldsymbol{d}',\ \boldsymbol{u} \geq \boldsymbol{0},\ \boldsymbol{s} \geq \boldsymbol{0}\}.$$

All inequalities define facets and the matrix $[D'\ I]$ has size $(m-d) \times m$.

Since P is not a cone, $\boldsymbol{d}' \neq \boldsymbol{0}$. Therefore, $\boldsymbol{d}'$ has at least one nonzero component, say the last one. By adding multiples of the last row to the other rows of $[D'\ I \,|\, \boldsymbol{d}']$, we can eliminate all other nonzero components of $\boldsymbol{d}'$. The resulting system with matrix $[A'\ A'']$ and right hand side $(0, \ldots, 0, \alpha)$, with $\alpha \neq 0$, clearly defines an affinely equivalent polyhedron. Let A^T denote the matrix $[A'\ A'']$ without the last row and let $\boldsymbol{b}$ be the last row of $[A'\ A'']$ divided by $-\alpha$ (in order to scale the right hand side to -1). We then have $A \in \mathbb{R}^{m\times(m-d-1)}$, $\boldsymbol{b} \in \mathbb{R}^m$, and

$$P \cong P' := \{\boldsymbol{y} \in \mathbb{R}^m \,:\, \boldsymbol{y}A = \boldsymbol{0},\ \boldsymbol{y}\boldsymbol{b} = -1,\ \boldsymbol{y} \geq \boldsymbol{0}\},$$

where each inequality $y_j \geq 0$ defines a facet for $j = 1, \ldots, m$. Since only affine transformations were applied, P' is affinely equivalent to P. □

Remark 1.11. Starting from a (rational) description of P, all transformations used in the proof of Lemma 1.10 can be applied in polynomial time (in the size of P). Furthermore, if P is described with coefficients in some subfield K of $\mathbb{R}$, e.g., $K = \mathbb{Q}$, the resulting description has coefficients in K as well. This will be used later (see Section 2.4).

We now come to the key relation between IISs and (supports of) vertices of the alternative polyhedron. Recall that the *support* of a vector $\boldsymbol{y} \in \mathbb{R}^m$, denoted by $\operatorname{supp} \boldsymbol{y}$, is the set of indices of its nonzero components, i.e., $\operatorname{supp} \boldsymbol{y} := \{i \in \{1, \ldots, m\} \,:\, y_i \neq 0\}$.

Theorem 1.12 (Gleeson and Ryan [65]). The index sets of IISs of Σ are exactly the supports of vertices of the alternative polyhedron $P(\Sigma)$.

Proof. We provide a new proof for convenience of the reader.

Let $\Sigma' : \{A'\boldsymbol{x} \leq \boldsymbol{b}'\}$ be an IIS of size k of $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$. W. l. o. g. we can assume that Σ' consists of the first k rows of Σ. Since Σ' is infeasible, by the Farkas lemma (Proposition 1.3) there exists $\boldsymbol{y} = (\boldsymbol{y}', \boldsymbol{0}) \geq \boldsymbol{0}$ with $\boldsymbol{y}' \in \mathbb{R}^k$, such that $A^T\boldsymbol{y} = \boldsymbol{0}$ and $\boldsymbol{y}\boldsymbol{b} < 0$. Since Σ' is minimally infeasible, it follows

that $\boldsymbol{y}' > \boldsymbol{0}$. We want to show that there exists a vertex $\boldsymbol{v}$ of $P(\Sigma)$ whose support indexes the rows of Σ', i.e., $\operatorname{supp} \boldsymbol{v} = \{1, \ldots, k\}$.

By Minkowski's theorem (see, e.g., Nemhauser and Wolsey [90, Chapter I.4] or Schrijver [107, Chapter 8]), there exist vertices $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s$ of $P(\Sigma)$ and generators $\boldsymbol{r}_1, \ldots, \boldsymbol{r}_t$ of extreme rays of $P(\Sigma)$ such that

$$\boldsymbol{y} = \lambda_1 \boldsymbol{v}_1 + \cdots + \lambda_s \boldsymbol{v}_s \; + \; \mu_1 \boldsymbol{r}_1 + \cdots + \mu_t \boldsymbol{r}_t$$

for appropriate

$$\lambda_1, \ldots, \lambda_s > 0 \quad \text{with} \quad \lambda_1 + \cdots + \lambda_s = 1 \quad \text{and} \quad \mu_1, \ldots, \mu_t > 0.$$

It is possible that $t = 0$. Since $\boldsymbol{v}_i, \boldsymbol{r}_j \geq \boldsymbol{0}$ and $\lambda_i, \mu_j > 0$, it follows that $\operatorname{supp} \boldsymbol{v}_i \subseteq \operatorname{supp} \boldsymbol{y}$ for all $i = 1, \ldots, s$. In fact, we have $\operatorname{supp} \boldsymbol{v}_i = \operatorname{supp} \boldsymbol{y}$, since otherwise Σ' would not be minimal by the Farkas lemma. Hence, $\boldsymbol{v}_1$ has the properties we looked for.

For the converse direction let $\boldsymbol{y}$ be a vertex of $P(\Sigma)$ and let the subsystem Σ' be indexed by the support of $\boldsymbol{y}$. By the Farkas lemma it follows that Σ' is infeasible. Assume that Σ' is not minimally infeasible and let Σ'' be an IIS strictly contained in Σ'. By the above direction, there exists a vertex $\boldsymbol{v}$ of $P(\Sigma)$ such that the support of $\boldsymbol{v}$ indexes the rows of Σ''. Hence, we have $\operatorname{supp} \boldsymbol{v} \subset \operatorname{supp} \boldsymbol{y}$. But then more inequalities of $P(\Sigma)$ would be satisfied at equality at $\boldsymbol{v}$ than at $\boldsymbol{y}$, and hence $\boldsymbol{y}$ cannot be a vertex. □

Since the support of each vertex is unique for polyhedra in equality form, the correspondence holds between the IISs of Σ and the vertices of $P(\Sigma)$. See also Example 2.25 on page 58 for an illustration of Theorem 1.12. This theorem can be extended (Parker [94], Parker and Ryan [95]) as follows:

Theorem 1.13. Let $\{C\boldsymbol{x} \leq \boldsymbol{c},\ D\boldsymbol{x} = \boldsymbol{d},\ \boldsymbol{\ell} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$ be an infeasible system. The sets of indices of irreducible inconsistent subsystems of this system are exactly the supports of vertices of the polyhedron

$$\begin{aligned} \{(\boldsymbol{y}, \boldsymbol{v}, \boldsymbol{w}, \boldsymbol{z}) \,:\; & \boldsymbol{y}C + \boldsymbol{v}D + \boldsymbol{w} - \boldsymbol{z} = \boldsymbol{0} \\ & \boldsymbol{y}\boldsymbol{c} + \boldsymbol{v}\boldsymbol{d} + \boldsymbol{w}\boldsymbol{u} - \boldsymbol{z}\boldsymbol{\ell} = -1 \\ & \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z} \geq \boldsymbol{0},\ \boldsymbol{v} \text{ free}\}. \end{aligned}$$

Furthermore, Theorem 1.12 can also be stated in terms of rays (see Parker and Ryan [95]) and elementary vectors.

Definition 1.14. An *elementary vector* of a subspace $L \subseteq \mathbb{R}^m$ is a nonzero vector $\boldsymbol{y} \in L$ which has minimal support (w. r. t. inclusion). In other words, if $\boldsymbol{x} \in L$ and $\operatorname{supp} \boldsymbol{x} \subset \operatorname{supp} \boldsymbol{y}$ then $\boldsymbol{x} = \boldsymbol{0}$.

Corollary 1.15 (Greenberg [66]). The set $S \subseteq [m]$ is an IIS of the system Σ if and only if there exists an elementary vector $\boldsymbol{y}$ in the subspace $L := \{\boldsymbol{y} \in \mathbb{R}^m \,:\, \boldsymbol{y}A = \boldsymbol{0}\}$ with $\boldsymbol{y}\boldsymbol{b} < 0$ and $\boldsymbol{y} \geq \boldsymbol{0}$ such that $S = \operatorname{supp} \boldsymbol{y}$.

1.2.2 IIS Simplex Decomposition

In this section, we provide a new geometric characterization of IISs. For notational convenience we consider the case where $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ is itself an IIS. Again, we assume $A \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. Let $S \subseteq [m] := \{1, \ldots, m\}$ and denote by A_S the $|S| \times n$ matrix consisting of those rows of A that are indexed by S. Let $A^i := A_{[m]\setminus\{i\}}$ denote the $(m-1) \times n$ submatrix of A obtained by removing the ith row of A. Similarly, let $\boldsymbol{b}^i := \boldsymbol{b}_{[m]\setminus\{i\}}$ be the $(m-1)$-dimensional vector $\boldsymbol{b}$ with ith component removed.

We first need the following result, which strengthens the condition of necessity in Theorem 1.6.

Lemma 1.16. Let $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ be an IIS with $m \geq 2$ inequalities. Then A^i has linearly independent rows, for all $1 \leq i \leq m$; i.e., $\operatorname{rank} A^i = m - 1$.

Proof. According to Proposition 1.3, there exists $\boldsymbol{y} > \boldsymbol{0}$ such that $\boldsymbol{y}A = \boldsymbol{0}$ and $\boldsymbol{y}\boldsymbol{b} = -1$ (by scaling $\boldsymbol{y}\boldsymbol{b} < 0$). Suppose some proper subset of rows is linearly dependent; i.e., there exists $\boldsymbol{z}$, such that $\boldsymbol{z}A = \boldsymbol{0}$, $\boldsymbol{z}\boldsymbol{b} \geq 0$ (without loss of generality) and some $z_k = 0$.

If there exists j with $z_j > 0$, consider $(\boldsymbol{y} - \epsilon\boldsymbol{z})A = 0$, $(\boldsymbol{y} - \epsilon\boldsymbol{z})\boldsymbol{b} \leq -1$, where $\epsilon = \min\{y_i/z_i \,:\, 1 \leq i \leq m,\ z_i > 0\} > 0$ (and $\boldsymbol{y}$ is as above). Then $\boldsymbol{y} - \epsilon\boldsymbol{z} \geq 0$, at least one additional component of $\boldsymbol{y} - \epsilon\boldsymbol{z}$ is 0, and the Farkas lemma (Proposition 1.3) contradicts the fact that the system is minimal infeasible (since $\boldsymbol{y} - \epsilon\boldsymbol{z}$ satisfies the requirements).

If $\boldsymbol{z} \leq \boldsymbol{0}$, then $-\boldsymbol{z} \geq \boldsymbol{0}$, $-\boldsymbol{z}A = \boldsymbol{0}$, and $-\boldsymbol{z}\boldsymbol{b} \leq 0$. Provided that $-\boldsymbol{z}\boldsymbol{b} < 0$, setting $\boldsymbol{y} = -\boldsymbol{z}$ in the Farkas lemma leads to a contradiction of minimality. If $-\boldsymbol{z}\boldsymbol{b} = 0$, then setting $\epsilon = \min\{\, y_i/(-z_i) \,:\, 1 \leq i \leq m,\ -z_i > 0\} > 0$ in $(\boldsymbol{y} + \epsilon\boldsymbol{z})A = \boldsymbol{0}$, $(\boldsymbol{y} + \epsilon\boldsymbol{z})\boldsymbol{b} = -1$ leads to a contradiction as above. □

Corollary 1.17. Let $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ be an infeasible system (with $m \geq 2$ inequalities). Then Σ is an IIS if and only if $\operatorname{rank} A^i = m-1$ for $i = 1, \ldots, m$.

Proof. If Σ is an IIS then $\operatorname{rank} A^i = m - 1$ by Lemma 1.16.

Now let $\operatorname{rank} A^i = m - 1$ for $i = 1, \ldots, m$. Assume that Σ is not an IIS, i.e., there exists a proper subsystem $\Sigma' : \{A'\boldsymbol{x} \leq \boldsymbol{b}'\}$ of Σ that is infeasible. Then A' has full rank, which is a contradiction to Theorem 1.4. □

We have the following *simplex decomposition* result for IISs:

Theorem 1.18. The system $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ is an IIS if and only if $\{A\boldsymbol{x} = \boldsymbol{b}\}$ does not have a solution and $\{\boldsymbol{x} \in \mathbb{R}^n \,:\, A\boldsymbol{x} \geq \boldsymbol{b}\} = L + Q$, where L is the lineality subspace $\{\boldsymbol{x} \in \mathbb{R}^n \,:\, A\boldsymbol{x} = \boldsymbol{0}\}$, Q is an $(m-1)$-simplex, and each vertex of Q is a solution for a subsystem $\{A^i\boldsymbol{x} = \boldsymbol{b}^i\}$, $1 \leq i \leq m$.[3]

[3] If $m = 1$, the system $\{A^1\boldsymbol{x} = \boldsymbol{b}^1\}$ is empty and hence has a solution (see Remark 1.19).

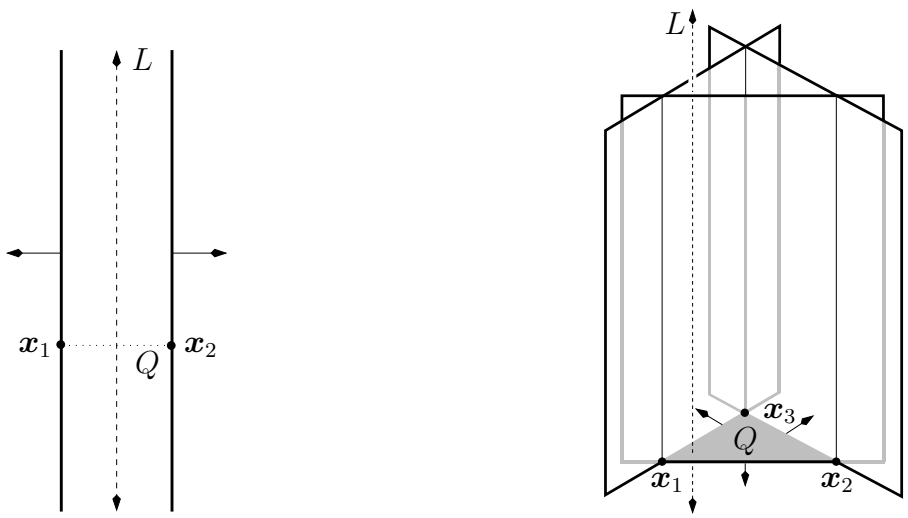


Figure 1.1: Two pictures illustrating Theorem 1.18 in dimensions $n = 2$ and $n = 3$. The IISs corresponding to $A\boldsymbol{x} \leq \boldsymbol{b}$ are indicated by the halfspaces with arrows pointing inward. If the halfspaces are turned around, the resulting polyhedron can be written as the sum of a simplex Q (indicated by the dotted segment and grey area, respectively) and a lineality space L (indicated by the dashed lines).

Proof. ($\Rightarrow$) Clearly, $\{A\boldsymbol{x} = \boldsymbol{b}\}$ does not have a solution.

To see the feasibility of $\{A\boldsymbol{x} \geq \boldsymbol{b}\}$, delete the ith constraint $\boldsymbol{a}^i\boldsymbol{x} \geq b_i$ to obtain the equality system $\{A^i\boldsymbol{x} = \boldsymbol{b}^i\}$. By Lemma 1.16, this system has a solution $\boldsymbol{x}_i$ (which is not necessarily unique). Then we have $\boldsymbol{a}^i\boldsymbol{x}_i > b_i$, otherwise $\boldsymbol{x}_i$ would satisfy $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$. Applying the polyhedral decomposition theorem, it follows that $P := \{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} \geq \boldsymbol{b}\} \neq \varnothing$ can be written as $P = K + Q$, where $K = \{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} \geq \boldsymbol{0}\}$ is its recession cone and $Q \subseteq P$ is a polytope, generated by minimal nonempty faces of P (see, e.g., Schrijver [107, Chapter 8]).

If $\hat{\boldsymbol{x}}$ satisfies $A\hat{\boldsymbol{x}} \geq \boldsymbol{0}$ and $\boldsymbol{a}^i\hat{\boldsymbol{x}} > 0$ for the ith row $\boldsymbol{a}^i$ of A then, for sufficiently large $\lambda > 0$, the vector $\boldsymbol{x}_i - \lambda\hat{\boldsymbol{x}}$ satisfies $A(\boldsymbol{x}_i - \lambda\hat{\boldsymbol{x}}) \leq \boldsymbol{b}$ and the original system $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ would be feasible. Therefore, $\boldsymbol{a}_i\hat{\boldsymbol{x}} = 0$ for all $1 \leq i \leq m$ and every $\hat{\boldsymbol{x}} \in K$. Hence, $K = L := \{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} = \boldsymbol{0}\}$.

Each minimal face of P is given by changing a maximal set of inequalities into equalities (in our context, all but one relation). Thus, we can take $Q = \operatorname{conv}\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$, where $\boldsymbol{x}_i$ is obtained by solving $\{A^i\boldsymbol{x} = \boldsymbol{b}^i\}$, as above. Assume, for the sake of contradiction, that $\boldsymbol{x}_i$ is affinely dependent on the other $m - 1$ points. Then there exist $\lambda_1, \ldots, \lambda_{i-1}, \lambda_{i+1}, \ldots, \lambda_m \in \mathbb{R}$, such that

$$\boldsymbol{x}_i = \sum_{j \neq i} \lambda_j \boldsymbol{x}_j \quad \text{with} \quad \sum_{j \neq i} \lambda_j = 1.$$

From the beginning of the proof, we know $\boldsymbol{a}^i\boldsymbol{x}_i > b_i$, but also

$$\boldsymbol{a}^i\boldsymbol{x}_i = \boldsymbol{a}^i(\sum_{j\neq i}\lambda_j\boldsymbol{x}_j) = \sum_{j\neq i}\lambda_j(\boldsymbol{a}^i\boldsymbol{x}_j) = \sum_{j\neq i}\lambda_j b_i = b_i,$$

which is a contradiction. Hence Q is the convex hull of m affinely independent points, i.e., an $(m-1)$-simplex.

($\Leftarrow$) If the system $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ is infeasible, then the minimality is obvious, because the simplex conditions on Q imply that every proper subsystem has an equality solution.

To show that $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ is infeasible, assume for the sake of contradiction that $\hat{\boldsymbol{x}} \in \{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} \leq \boldsymbol{b}\} \neq \varnothing$ and $\hat{\boldsymbol{x}}$ satisfies a maximal number of these relations with equality. Since $\{A\boldsymbol{x} = \boldsymbol{b}\}$ is assumed to be infeasible, we have $A\hat{\boldsymbol{x}} \neq \boldsymbol{b}$, i.e., there exists $i \in \{1,\ldots,m\}$ with $\boldsymbol{a}^i\hat{\boldsymbol{x}} < b_i$. Let $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m$ be the vertices of Q, such that $\boldsymbol{x}_i$ is a solution of $\{A^i\boldsymbol{x} = \boldsymbol{b}^i\}$. Similarly, since $\{A\boldsymbol{x} = \boldsymbol{b}\}$ is infeasible, we have $\boldsymbol{a}^i\boldsymbol{x}_i > b_i$. Thus, we can set $\lambda = (\boldsymbol{a}^i\boldsymbol{x}_i - b_i)/(\boldsymbol{a}^i\boldsymbol{x}_i - \boldsymbol{a}^i\hat{\boldsymbol{x}})$ with $0 < \lambda < 1$, so that $\boldsymbol{a}^i(\lambda\hat{\boldsymbol{x}} + (1-\lambda)\boldsymbol{x}_i) = b_i$. But then at $\lambda\hat{\boldsymbol{x}} + (1-\lambda)\boldsymbol{x}_i$ more relations of $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ hold with equality than at $\hat{\boldsymbol{x}}$, contradicting the choice of $\hat{\boldsymbol{x}}$. □

Remark 1.19. We have the following special cases of Theorem 1.18.

If $m = 1$, any IIS consists of one inequality $\boldsymbol{0}\boldsymbol{x} \leq \alpha$ with $\alpha < 0$. Hence, we have $L = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{0}\boldsymbol{x} = 0\} = \mathbb{R}^n$. The vertex for the 0-simplex Q is unconstrained. Therefore, we can take an arbitrary $\boldsymbol{v} \in \mathbb{R}^n$. We get that $\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{0}\boldsymbol{x} \geq \alpha\} = \mathbb{R}^n = L + \boldsymbol{v}$, as claimed.

If $m = n+1$, then A has $n+1$ rows. Assuming A to be of full column rank, it follows that $L = \{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} = 0\} = \{0\}$, $Q = \operatorname{conv}\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_{n+1}\}$ is an n-simplex and $\{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} \geq \boldsymbol{b}\} = \{0\} + Q$.

Remark 1.20. According to the above proof, among all possible solutions $\boldsymbol{x}_i$ of the corresponding subsystems $\{A^i\boldsymbol{x} = \boldsymbol{b}^i\}$, for $1 \leq i \leq m$, we can take the representatives of the minimal nonempty faces of $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ that lie in $L^\perp$, the orthogonal complement of L; i.e., $Q \subset L^\perp$ (see Figure 1.1). By Lemma 1.16, we know that $\{\boldsymbol{x} \in \mathbb{R}^n : A^i\boldsymbol{x} = \boldsymbol{b}^i\} = \boldsymbol{x}_i + L$, where L is the lineality space of the system $\{A\boldsymbol{x} \geq \boldsymbol{b}\}$.

1.2.3 Minimum Cardinality IISs

We now turn to the complexity status of MIN IIS, the problem to find a minimum cardinality IIS. We prove that MIN IIS is not only $\mathcal{NP}$-hard to solve optimally, but also hard to approximate. This settles an issue left open by Chinneck and Dravnieks [49], Greenberg and Murphy [67], and Parker

and Ryan [95]. Remember that DTIME($T(s)$) denotes the class of problems whose instances of size s can be solved in deterministic time $\mathcal{O}(T(s))$. It is generally believed that $\mathcal{NP} \not\subseteq$ DTIME($s^{\text{polylog}\, s}$), see Section 1.1.3.

Theorem 1.21. Let Σ be an infeasible linear inequality system. Then, assuming $\mathcal{P} \neq \mathcal{NP}$, no polynomial-time algorithm is guaranteed to yield an IIS of Σ whose cardinality is at most c times larger than the minimum one, for any constant $c \geq 1$. In particular, Min IIS is $\mathcal{NP}$-hard.

Furthermore, assuming $\mathcal{NP} \not\subseteq$ DTIME($s^{\text{polylog}\, s}$), Min IIS cannot be approximated within a factor $2^{\log^{1-\varepsilon} m}$, for any $\varepsilon > 0$, where m is the number of inequalities of Σ.

Proof. We proceed by reduction from the following problem: Given a feasible linear system $D\boldsymbol{z} = \boldsymbol{d}$, with $D \in \mathbb{R}^{m' \times n'}$ and $\boldsymbol{d} \in \mathbb{R}^{m'}$, find a solution $\boldsymbol{z}$ satisfying all equations with as few nonzero components as possible. Amaldi and Kann [7] proved that this problem is hard to approximate within the same type of factors, but with m replaced by the number of variables n'. Note that the above nonconstant factor grows faster than any polylogarithmic function, but slower than any polynomial function.

For each instance of the latter problem which has an optimal solution containing s nonzero components, we construct a particular instance of the Min IIS problem with a minimum cardinality IIS containing $s+1$ inequalities. Given any such instance $(D,\ \boldsymbol{d})$, consider the system

$$[D\ -D\ -\boldsymbol{d}] \begin{pmatrix} \boldsymbol{z}^+ \\ \boldsymbol{z}^- \\ z_0 \end{pmatrix} = \boldsymbol{0}, \quad (\boldsymbol{0}\quad \boldsymbol{0}\ -1) \begin{pmatrix} \boldsymbol{z}^+ \\ \boldsymbol{z}^- \\ z_0 \end{pmatrix} < 0, \quad \begin{pmatrix} \boldsymbol{z}^+ \\ \boldsymbol{z}^- \\ z_0 \end{pmatrix} \geq \boldsymbol{0}. \tag{1.2}$$

Since the strict inequality implies $z_0 > 0$, the system $D\boldsymbol{z} = \boldsymbol{d}$ has a solution with s nonzero components if and only if (1.2) has one with $s+1$ nonzero components. Applying Corollary 1.15, (1.2) has such a solution if and only if the system

$$\begin{pmatrix} D^T \\ -D^T \\ -\boldsymbol{d} \end{pmatrix} \boldsymbol{x} \leq \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ -1 \end{pmatrix} \tag{1.3}$$

has an IIS of cardinality $s+1$. Since (1.3) is the alternative system of (1.2), the Farkas lemma (Proposition 1.3) implies that exactly one of these two is feasible. As (1.2) is feasible, (1.3) must be infeasible. Thus, (1.3) is a particular instance of Min IIS with $m = 2n' + 1$ inequalities in $n = m'$ variables.

This polynomial-time reduction preserves the objective function value modulo an additive unit constant. Hence, we obtain the same type of non-approximability factors for MIN IIS. □

A problem similar to MIN IIS is that of determining minimum witnesses of infeasibility in a network with demands and supplies at the nodes. Aggarwal, Ahuja, Hao, and Orlin [1] proved that this problem is $\mathcal{NP}$-hard.

1.3 FEASIBLE SUBSYSTEM POLYTOPE

In this section, we study the feasible subsystem polytope P_{FS}, i.e., the convex hull of all characteristic vectors of feasible subsystems of a given infeasible linear inequality system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ (where as usual $A \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$). We derive the following results about the facial structure of this polytope: First, we give a new proof of the fact that IISs give rise to facets of P_{FS} and prove that the corresponding separation problem is $\mathcal{NP}$-hard (Section 1.3.2). Since every subsystem of a feasible system is again feasible, the set of feasible subsystems of Σ forms a so-called independence system. Generalized antiwebs are structures often leading to facets of the polytopes corresponding to independence systems. In Section 1.3.3, we investigate generalized antiwebs in our context and characterize when they give rise to facets of P_{FS}. For this, results of Chapter 2 and 3 are incorporated. It turns out that only two very specific types of generalized antiwebs can occur.

1.3.1 Independence System Polytopes

Given a finite groundset S and a collection of subsets $\mathcal{I} \subseteq 2^S$, the pair $(S, \mathcal{I})$ is a (finite) *independence system*, if $I \in \mathcal{I}$ and $J \subset I$ imply $J \in \mathcal{I}$. There exist many other names for this concept, which arises in different areas, e.g., abstract simplicial complex, hereditary structure (hypergraph), and order ideal (in posets). In independence systems, a set belonging to $\mathcal{I}$ is called *independent* and a subset of S that does not belong to $\mathcal{I}$ is called *dependent*. The *rank* of a subset $S' \subseteq S$ is defined by $r(S') := \max\{|I| : I \subseteq S',\ I \in \mathcal{I}\}$. The rank of the independence system $(S, \mathcal{I})$ is $r(S)$. An independence system can be given by:

(i) its independent sets $\mathcal{I}$,
(ii) its maximal independent sets, the *bases*,
(iii) its collection of dependent sets,
(iv) its minimal dependent sets $\mathcal{C}$, the *circuits*,
(v) its rank function $r : 2^S \to \mathbb{Z}$.

Here, maximal and minimal are meant with respect to inclusion. All of the above descriptions are equivalent, i.e., from each of the above representations one can compute any other. This is clearly the case for (i) and (ii), as well as for (iii) and (iv). The remaining equivalences follow easily from three facts: The set $S' \subseteq S$ is independent if and only if $r(S') = |S'|$. A set $S' \subseteq S$ is independent if and only if it contains no circuit. The set S' is dependent if and only if it is not contained in any base.

To any independence system $(S, \mathcal{I})$ we can associate the *independence system polytope*

$$P_{IS}(\mathcal{I}) := \text{conv}\{\boldsymbol{y} \in \{0,1\}^{|S|} \,:\, \boldsymbol{y} \text{ is the incidence vector of a set } I \in \mathcal{I}\},$$

which we usually abbreviate by P_{IS} if it is clear from the context to which independent system it belongs. Sometimes we also write $P_{IS}(\mathcal{C})$ for $P_{IS}(\mathcal{I})$, where $\mathcal{C}$ is the collection of circuits of $\mathcal{I}$. The polytope P_{IS} is closely related to the *set covering polytope*

$$P_{SC}(\mathcal{C}) := \text{conv}\{\boldsymbol{y} \in \{0,1\}^{|S|} \,:\, \sum_{i \in C} y_i \geq 1 \quad \text{for all } C \in \mathcal{C}\}.$$

Again, we usually write P_{SC} if it is clear which circuits $\mathcal{C}$ refers to. A more standard description of P_{SC} is $\{\boldsymbol{y} \in \{0,1\}^{|S|} \,:\, M\boldsymbol{y} \geq \mathbb{1}\}$, where $\mathbb{1}$ denotes the vector of all ones and the $|\mathcal{C}| \times |S|$ matrix M has the incidence vectors of the sets in $\mathcal{C}$ as rows.

A little thought shows that $\boldsymbol{y} \in P_{IS}$ if and only if $\boldsymbol{z} = \mathbb{1} - \boldsymbol{y} \in P_{SC}$. As this is an affine transformation, facets of P_{IS} correspond to facets of P_{SC} and vice versa. Hence, a facet-defining inequality for one polytope can be easily transformed to a facet-defining inequality for the other polytope.

As a special case of independence systems we have stable sets (independent sets) in a graph, with corresponding polytope P_{ST}. Similarly, we have the vertex covers in a graph, with corresponding polytope P_{VC}, as a special case of set covering problems. The above relation between P_{IS} (P_{ST}) and P_{SC} (P_{VC}) then translates to the statement that the complement of a stable set is a vertex cover and conversely.

More about the structure of P_{IS} and P_{SC} and algorithms to solve related optimization problems can be found in the references of the annotated bibliography of Ceria, Nobili, and Sassano [40]. An overview of known results about the facial structure is also given by Borndörfer [35]. Cornuéjols [51] discusses conditions under which the LP relaxation of P_{SC} is integral, i.e., has only integral vertices.

Consider the infeasible system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ and (as usual) identify each inequality of Σ with its index in $[m] := \{1, \ldots, m\}$. If $\mathcal{I}(\Sigma)$ denotes the

set of all feasible subsystems of Σ, $([m], \mathcal{I}(\Sigma))$ is clearly an independence system, and the set of circuits $\mathcal{C}(\Sigma)$ is the set of all IISs. We denote by $P_{FS}(\Sigma) := P_{IS}(\mathcal{I}(\Sigma))$ (or P_{FS} for short) the *feasible subsystem polytope*, which is the convex hull of all the incidence vectors of feasible subsystems of Σ. The set covering polytope $P_{SC}(\mathcal{C}(\Sigma))$ becomes the *IIS-covering polytope*, denoted by $P_{IISC} = P_{IISC}(\Sigma)$, when restricted to this special independence system (see Equation (1.1)).

Clearly, to solve (weighted) versions of MAX FS and MIN IIS COVER, one has to optimize a linear function over P_{FS} and P_{IISC}, respectively. Hence, in order to apply polyhedral solution methods like branch-and-cut one should investigate the structure of these polytopes. Before doing so, we need to recall some more definitions and facts regarding (general) independence system polytopes.

Let $(S, \mathcal{I})$ be an independence system. For any $S' \subseteq S$, the *rank inequality* for S' is

$$\sum_{s \in S'} y_s \leq r(S')\,, \tag{1.4}$$

which is clearly valid for P_{IS}. A subset $S' \subseteq S$ is *closed* if

$$r(S') < r(S' \cup \{t\}) \quad \text{for all } t \in S \setminus S',$$

and *nonseparable* if

$$r(S') < r(T) + r(S' \setminus T) \quad \text{for all } T \subset S',\ T \neq \varnothing.$$

The following result is well known.

Lemma 1.22. Let $(S, \mathcal{I})$ be an independence system such that $\{s\} \in \mathcal{I}$ for all $s \in S$. If the rank inequality (1.4) for $S' \subseteq S$ defines a facet of $P_{IS}(\mathcal{I})$, then S' is closed and nonseparable.

Proof. By assumption each single element of the groundset is independent. Therefore, P_{IS} contains the $|S|+1$ affinely independent vectors $\mathbf{0}, \boldsymbol{e}_1, \ldots, \boldsymbol{e}_{|S|}$ and hence is full-dimensional.

If S' is not closed, there exists $t \in S \setminus S'$ such that $r(S' \cup \{t\}) = r(S')$. Then the inequality

$$\sum_{s \in S' \cup \{t\}} y_s \leq r(S' \cup \{t\}) = r(S')$$

is clearly valid for P_{IS} and dominates (1.4). Since P_{IS} is full-dimensional, it follows that (1.4) cannot define a facet.

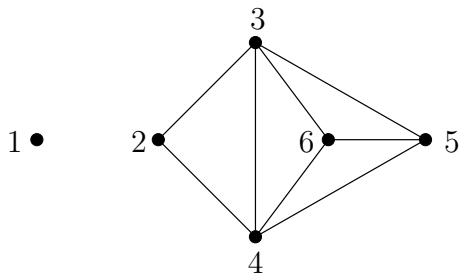


Figure 1.2: Critical graph of Example 1.24.

If S' is separable (i.e., not nonseparable), there exists $T \subset S'$, $T \neq \varnothing$ with $r(S') = r(T) + r(S' \setminus T)$. But then (1.4) is the sum of the inequalities

$$\sum_{s \in T} y_s \leq r(T) \quad \text{and} \quad \sum_{s \in S' \setminus T} y_s \leq r(S' \setminus T),$$

which both dominate it. Hence, (1.4) cannot define a facet, since P_{IS} is full-dimensional. □

In general, the condition that S' has to be closed and nonseparable is only necessary, but sufficient conditions can be stated using the following concept (see Laurent [80]). For $S' \subseteq S$, the *critical graph* $G_{S'}(\mathcal{I}) = (S', E)$ w. r. t. S' is defined as follows: $\{s, t\} \in E$, for $s, t \in S'$, if and only if there exists an independent set I such that $I \subseteq S'$, $|I| = r(S')$, and $s \in I$, $t \notin I$, $I - s + t \in \mathcal{I}$.

Theorem 1.23 (Laurent [80], Sassano [106]). Let $(S, \mathcal{I})$ be an independence system such that $\{s\} \in \mathcal{I}$ for all $s \in S$. If S' is a closed subset of S and the critical graph $G_{S'}(\mathcal{I})$ is connected, then the corresponding rank inequality (1.4) induces a facet of $P_{IS}(\mathcal{I})$.

Example 1.24. Cornuéjols and Sassano [52] state an example due to John Hooker which shows that this condition is not necessary: Consider the independence system with groundset $S := \{1, 2, \ldots, 6\}$ and which has the following circuits:

{1, 3}, {1, 5}, {1, 6}, {2, 3}, {2, 4, 5}, {2, 4, 6}, {3, 4, 5, 6}.

The maximal independent sets are:

{1, 2, 4}, {3, 4, 5}, {3, 4, 6}, {2, 5, 6}, {3, 5, 6}, {4, 5, 6}.

The critical graph G_S is given in Figure 1.2. Hence, G_S is not connected and S is (trivially) closed. But one can check, e.g., using `polymake` [63, 64], that $y_1 + y_2 + y_3 + y_4 + y_5 + y_6 \leq 3$ defines a facet of P_{IS}.

To date, no characterization of the cases when a rank inequality is facet-defining is known.

1.3.2 Facets of the Feasible Subsystem Polytope

We now turn our attention to the feasible subsystem polytope. Parker [94] began an investigation of P_{IISC}. Using well-known facts about independence system polytopes he obtained the following results, which are translated to P_{FS} by the above mentioned transformation.

Lemma 1.25 (Parker [94]).

(a) $P_{FS}(\Sigma)$ is full-dimensional if and only if there are no IISs of cardinality one in Σ. (See proof of Lemma 1.22.)

(b) The inequality $y_i \geq 0$ defines a facet of $P_{FS}(\Sigma)$ for $1 \leq i \leq m$ if and only if $\{i\}$ is not an IIS of cardinality one.

(c) For each i, $1 \leq i \leq m$, the inequality $y_i \leq 1$ defines a facet of $P_{FS}(\Sigma)$ if and only if there is no IIS in Σ of cardinality one or two including i.

In a preprocessing step, IISs of cardinality one, i.e., single inequalities of the form $\mathbf{0}\boldsymbol{x} \leq -\alpha$ with $\alpha > 0$, can easily be detected and removed. Since the solution of MAX FS can then be adjusted accordingly, we agree on the following for the rest of this chapter.

Assumption. The system Σ contains no IIS of cardinality one.

We turn to facets arising from IISs. Let $C \in \mathcal{C}(\Sigma)$ be an arbitrary IIS of Σ, with $A_C\boldsymbol{x} \leq \boldsymbol{b}_C$ as its associated subsystem. The rank inequality (1.4) takes the form:

$$\sum_{i \in C} y_i \leq r(C) = |C| - 1,$$

which is referred to as an *IIS-inequality* in the following. The corresponding covering inequality $\sum_{i \in C} y_i \geq 1$ for P_{IISC} is proved to be facet-defining in [94]. Hence, we have:

Theorem 1.26 (Parker [94]). Every IIS-inequality defines a (rank) facet of $P_{FS}(\Sigma)$.

Proof. It is easy to verify that IIS-inequalities are valid for $P_{FS}(\Sigma)$.

For every IIS C, each subset I of size $|C| - 1$ is independent (feasible). Hence, if $\{t\} = C \setminus I$, then $I - s + t$ is independent for every $s \in I$. It follows that the critical graph G_C is a complete graph and therefore connected. The next proposition shows that every IIS is closed, which finishes the proof by Theorem 1.23. □

Proposition 1.27. Let $A \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, and let $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ be an infeasible system. Then every IIS $C \subseteq [m]$ is closed in the independence system $([m], \mathcal{I}(\Sigma))$ arising from the feasible subsystems of Σ.

This proposition clearly follows from Theorem 1.26 (as proved in [94]) and Lemma 1.22. We give a new geometric proof (based on Theorem 1.18), which provides additional insight into the structure of IISs.

Proof. (a) First consider the case of a maximal IIS C, i.e., $|C| = n + 1 > 1$. For each $i \in C$, consider the unique solution $\boldsymbol{x}_i$ of $A_{C\setminus\{i\}}\boldsymbol{x} = \boldsymbol{b}_{C\setminus\{i\}}$. By the proof of Theorem 1.18, we know that $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n+1}$ are affinely independent. Define the barycenter of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n+1}$:

$$\hat{\boldsymbol{x}} := \frac{1}{n+1}\sum_{i=1}^{n+1} \boldsymbol{x}_i,$$

and define $\boldsymbol{d}_i := (\boldsymbol{x}_i - \hat{\boldsymbol{x}})$ for all i, $1 \leq i \leq n + 1$. Then $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n+1}$ are also affinely independent. Clearly, $\sum_{i=1}^{n+1} \boldsymbol{d}_i = \boldsymbol{0}$ and the $\boldsymbol{d}_i$'s generate $\mathbb{R}^n$ (see Figure 1.3). Since each $\boldsymbol{x}_i$ satisfies exactly n of the $n + 1$ inequalities in $\{A_C\boldsymbol{x} \leq \boldsymbol{b}_C\}$ with equality and for the ith one $\boldsymbol{a}^i\boldsymbol{x}_i > b_i$ (otherwise C would be feasible), we have $\hat{\boldsymbol{x}} \in \{\boldsymbol{x} \in \mathbb{R}^n : A_C\boldsymbol{x} \geq \boldsymbol{b}_C\}$; i.e., $\hat{\boldsymbol{x}}$ satisfies the reversed inequalities of the IIS C. In fact, $\hat{\boldsymbol{x}}$ is an interior point of the polyhedron $\{\boldsymbol{x} \in \mathbb{R}^n : A_C\boldsymbol{x} \geq \boldsymbol{b}_C\}$.

According to Theorem 1.8, deleting any inequality from an IIS yields a feasible subsystem that defines an affine (convex) cone. In our case, we

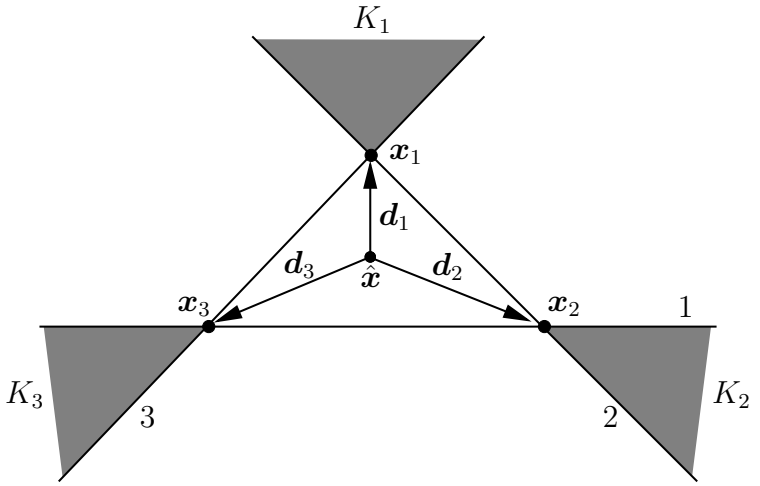


Figure 1.3: Illustration of the proof of Proposition 1.27.

have $n+1$ affine cones $K_i := \boldsymbol{x}_i + K_i'$, where $K_i' = \{\boldsymbol{x} \in \mathbb{R}^n : A_{C\setminus\{i\}}\boldsymbol{x} \leq \boldsymbol{0}\}$ for $1 \leq i \leq n+1$. The ray $R_i := \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{x} = \boldsymbol{x}_i + \lambda\,\boldsymbol{d}_i,\ \lambda \geq 0\}$, generated by $\boldsymbol{d}_i$ and passing through $\boldsymbol{x}_i$, is contained in K_i, because for any $\lambda \geq 0$ we have:

$$A_{C\setminus\{i\}}(\lambda\,\boldsymbol{d}_i) = \lambda\,A_{C\setminus\{i\}}(\boldsymbol{x}_i - \hat{\boldsymbol{x}}) = \lambda\,(\boldsymbol{b}_{C\setminus\{i\}} - A_{C\setminus\{i\}}\hat{\boldsymbol{x}}) \leq \boldsymbol{0},$$

where we used the fact that $A_{C\setminus\{i\}}\hat{\boldsymbol{x}} \geq \boldsymbol{b}_{C\setminus\{i\}}$.

Consider an arbitrary inequality $\boldsymbol{a}\boldsymbol{x} \leq \alpha$ with $\boldsymbol{a} \neq \boldsymbol{0}$. We will verify that the halfspace $H := \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{a}\boldsymbol{x} \leq \alpha\}$ has a nonempty intersection with K_i for some i ($1 \leq i \leq n+1$). In particular, if $\boldsymbol{a}\boldsymbol{x} \leq \alpha$ is the ith inequality of Σ and is not contained in C, i.e., $i \in [m] \setminus C$, then $C \cup \{i\}$ contains a feasible system of size $|C| = (|C| - 1) + 1 = r(C) + 1$. Therefore, it follows that $r(C \cup \{i\}) = |C| > r(C)$, which proves that C is closed.

Since $\sum_{i=1}^{n+1} \boldsymbol{d}_i = \boldsymbol{0}$, it follows that

$$\sum_{i=1}^{n+1} \boldsymbol{a}\boldsymbol{d}_i = \boldsymbol{a}(\sum_{i=1}^{n+1} \boldsymbol{d}_i) = 0.$$

Because $\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n+1}$ generate $\mathbb{R}^n$ and $\boldsymbol{a} \neq \boldsymbol{0}$, there exists i, $1 \leq i \leq n+1$, such that $\boldsymbol{a}\boldsymbol{d}_i \neq 0$. Thus, there also exists at least one i such that $\boldsymbol{a}\boldsymbol{d}_i < 0$. But this implies that $R_i \cap H \neq \varnothing$. In other words, $K_i \cap H \neq \varnothing$, which proves the proposition for maximal IISs.

(b) Consider the case of a non-maximal IIS C, i.e., $|C| < n+1$. It follows from Theorem 1.18 and Remark 1.20 that $P := \{\boldsymbol{x} \in \mathbb{R}^n : A_C\boldsymbol{x} \geq \boldsymbol{b}_C\} = L + Q$ with $Q \subseteq L^\perp$, where $L = \{\boldsymbol{x} \in \mathbb{R}^n : A_C\boldsymbol{x} = \boldsymbol{0}\}$ is a linear space and Q is a polytope generated by $|C|$ affinely independent points. Since P is full-dimensional (the vertex-barycenter of Q is an interior point, unless C is an IIS of cardinality one, a case which we excluded), we have $n = \dim P = \dim L + \dim Q$. Because $\dim Q = |C| - 1 < n$, this implies that $\dim L \geq 1$.

Two cases can arise:

(i) If the above mentioned $\boldsymbol{a}$ is contained in $\operatorname{lin}\{\boldsymbol{a}^i : i \in C\} = L^\perp$, the linear hull of the rows of A_C, then we can project H onto $L^\perp$. Since $\dim L^\perp = \dim Q$, the IIS C projected to $L^\perp$ is maximal and the above result applies (see Figure 1.3).

(ii) If we have $\boldsymbol{a} \notin \operatorname{lin}\{\boldsymbol{a}^i : i \in C\} = L^\perp$, the projection of the hyperplane $H^= := \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{a}\boldsymbol{x} = \alpha\}$ onto $L^\perp$ yields all of $L^\perp$. Therefore, $H = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{a}\boldsymbol{x} \leq \alpha\}$ has a nonempty intersection with all the cones corresponding to maximal consistent subsystems of $\{A_C\boldsymbol{x} \leq \boldsymbol{b}_C\}$.

This finishes the proof. □

Remark. In general independence systems not all circuits have to be closed. For example, consider the independent system defined by stable sets in a simple graph. Here, the circuits correspond to the edges of the graph, which are not necessarily closed: consider any triangle in the graph.

Example 1.28. In the more general case where we also allow equations in the system Σ, the generalization of Proposition 1.27 is false. The definition of P_{FS} and IISs easily carry over to this case. If equations are allowed, then IISs are not necessarily closed. Consider the following system in $\mathbb{R}^2$:

$$\begin{array}{lrcl} (1) & x_1 & \leq & 0 \\ (2) & x_2 & \leq & 0 \\ (3) & x_1 + x_2 & = & 1 \\ (4) & x_1 + x_2 & \geq & 2. \end{array}$$

The first three inequalities $\{1, 2, 3\}$ form an IIS. The inequalities $\{3, 4\}$ and $\{1, 2, 4\}$ also form IISs. Hence, there exists no subset of three inequalities which is feasible, i.e., the IIS $\{1, 2, 3\}$ is not closed. Using `polymake` [63, 64] one can easily check with this example that the generalization of Theorem 1.26 is false, too.

We now turn to the following problem:

IIS-inequality separation problem: Given an infeasible system Σ and an arbitrary vector $\boldsymbol{y} \in \mathbb{R}^m$, show that $\boldsymbol{y}$ satisfies all IIS-inequalities or find at least one violated by $\boldsymbol{y}$.

In view of this problem, we can assume that $\boldsymbol{y} \in [0, 1]^m$. Moreover, we may assume without loss of generality that the nonzero components of $\boldsymbol{y}$ correspond to an infeasible subsystem of Σ.

Proposition 1.29. The separation problem for IIS-inequalities is $\mathcal{NP}$-hard.

Proof. We proceed by polynomial-time reduction from the decision version of the Min IIS problem, which is $\mathcal{NP}$-complete according to Theorem 1.21. This decision version is: Given an infeasible system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ with m inequalities, n variables, and a positive integer K with $1 \leq K \leq n+1$, does Σ have an IIS of cardinality at most K?

Let $(A, \boldsymbol{b}, K)$ be an arbitrary instance of the above decision problem. Consider the particular instance of the separation problem given by the same infeasible system together with the vector $\boldsymbol{y}$ such that $y_i = 1 - 1/(K+1)$ for all i, $1 \leq i \leq m$.

Suppose that Σ has an IIS C of cardinality at most K. Then the corresponding IIS-inequality $\sum_{i\in C} y_i \leq |C|-1$ is violated by the vector $\boldsymbol{y}$, because

$$\sum_{i\in C} y_i = \sum_{i\in C}\Big(1-\frac{1}{K+1}\Big) = |C| - \frac{|C|}{K+1} > |C|-1,$$

where the strict inequality is implied by $|C| \leq K$. Thus, the vector $\boldsymbol{y}$ can be separated from P_{FS}.

Conversely, if there exists an IIS-inequality violated by $\boldsymbol{y}$, then

$$\sum_{i\in C} y_i = |C| - \frac{|C|}{(K+1)} > |C|-1$$

implies that the cardinality of C is at most K. Therefore, Σ has an IIS of cardinality at most K if and only if some IIS-inequality is violated by the given vector $\boldsymbol{y}$. □

1.3.3 Rank Facets Arising from Generalized Antiwebs

Laurent [80] introduced generalized antiwebs, which generalize cliques, odd holes, and antiholes in graphs to independence systems. Necessary and sufficient conditions were also established for the corresponding rank inequalities to define facets of P_{IS}.

Let m, t, q be integers such that $2 \leq q \leq t \leq m$, let $S = \{s_0, \ldots, s_{m-1}\}$ be a finite set, and define for each $i \in M := \{0, \ldots, m-1\}$ the subset $S_i = \{s_i, \ldots, s_{i+t-1}\}$ (where the indices are taken modulo m) formed by t consecutive elements of S. An (m,t,q)-*generalized antiweb* on S is the independence system having the following family of subsets of S as circuits:

$$\mathcal{AW}(m,t,q) = \{C \subseteq S \,:\, C \subseteq S_i \ \text{ for some } \ i \in M,\ |C| = q\}.$$

See Figure 1.4 on page 36 for an example.

Let $P_{IS}(\mathcal{AW}(m,t,q))$ be the polytope of the (m,t,q)-generalized antiweb and define $\mathcal{AW}(m,t) := \mathcal{AW}(m,t,t)$. Note that the case $t = q = 1$ would correspond to m trivially infeasible inequalities, e.g., $\mathbf{0}\,\boldsymbol{x} \leq -1$. The circuit-element incidence matrix of $\mathcal{AW}(m,t)$ is just an (m,t)-circulant (see Section 3.5.2).

As observed in [80], $\mathcal{AW}(m,t,q)$ corresponds to *generalized cliques* when $m = t$, to *generalized odd holes* when $q = t$ and t does not divide m, and to *generalized antiholes* when $m = q\,t + 1$.

In this section we determine under which circumstances generalized antiwebs give rise to rank facets of the form $\sum_{i\in S'} y_i \leq r(S')$ of $P_{FS}(\Sigma)$. Define

the hypergraph $\mathcal{H}(\mathcal{AW}(m,t,q)) := (S, \mathcal{AW}(m,t,q))$, which has S as node set and the sets in $\mathcal{AW}(m,t,q)$ as edges; see Chapter 2 for a definition of hypergraphs. A hypergraph $(S, \mathcal{E})$ is an *IIS-hypergraph*, if the nodes in S correspond to the inequalities of an infeasible linear inequality system and the edges in $\mathcal{E}$ correspond to the (sets of inequalities of) IISs of this system (cf. Definition 2.1).

The first question we consider is: For which values of m, t, and q is the hypergraph $\mathcal{H}(\mathcal{AW}(m,t,q))$ an IIS-hypergraph?

Lemma 1.30. If $\mathcal{H}(\mathcal{AW}(m,t,q))$ is an IIS-hypergraph then $t = q$.

Proof. Suppose that $q < t$. Consider an arbitrary circuit $C \in \mathcal{AW}(m,t,q)$ that is contained in S_1 and an arbitrary element $s \in S_1 \setminus C$. By definition of $\mathcal{AW}(m,t,q)$, any cardinality q subset of S_1 is a circuit. In particular, this is true for all subsets containing s and $q-1$ elements of C. But then C cannot be closed because $r(C \cup \{s\}) = r(C)$ and we have a contradiction to the fact that all IISs are closed (Proposition 1.27). □

Together with Lemma 2.16 of Chapter 2 and Proposition 3.28 of Chapter 3, we obtain the following result. In fact, proving this proposition was a motivation for the investigations of parts Chapter 2 and 3.

Proposition 1.31. $\mathcal{H}(\mathcal{AW}(m,t,q))$ is an IIS-hypergraph if and only if $t = q$ and

(i) $t = m - 2$ or
(ii) $t = m$.

Proof. Lemma 1.30 implies that necessarily $t = q$. Now assume $\mathcal{H}(\mathcal{AW}(m,t))$ is an IIS-hypergraph. If $t = m$, we have a single IIS of size m. Therefore assume $t < m$.

In order to apply Lemma 2.16, we need the following concepts, which are defined more formally in Section 2.4 of Chapter 2. Let $H = (S, \mathcal{E})$ be a hypergraph. Define the *complement hypergraph* $\overline{H} := (S, (S \setminus E : E \in \mathcal{E}))$ of H. Let E^* be a distinct node for every edge E of H and for any node $s \in S$ of H let $\mathcal{E}_s^* := \{E^* : s \in E,\ E \in \mathcal{E}\}$. We can then define the *dual hypergraph* $H^* := (\{E^* : E \in \mathcal{E}\}, (\mathcal{E}_s^* : s \in S))$ of H, which has the nodes corresponding to the edges of H as nodes and the sets $\mathcal{E}_s^*$ as edges. If M is the edge-node incidence matrix of H, then the corresponding incidence matrix of $\overline{H}$ is the matrix obtained by exchanging zeros and ones. The incidence matrix of H^* is just the transposed incidence matrix of M. See Figure 1.4 for an example. The hypergraph H is *isomorphic* to a hypergraph H', if the rows and columns of the incidence matrix of H' can be permuted so that the result equals the incidence matrix of H; compare the definition in Chapter 2.

If we set $H := \mathcal{H}(\mathcal{AW}(m,t))$, then $\overline{H}$ is isomorphic to $\mathcal{H}(\mathcal{AW}(m,k))$ with $k := m - t > 0$. Furthermore, we have that $\mathcal{H}(\mathcal{AW}(m,k))^*$ is isomorphic to $\mathcal{H}(\mathcal{AW}(m,k))$ again. Therefore, $\overline{H^*} = \overline{\mathcal{H}(\mathcal{AW}(m,t))^*}$ is isomorphic to $\mathcal{H}(\mathcal{AW}(m,k))$.

Since $t < m$, $\overline{H^*}$ is a clutter hypergraph (no edge contains any other edge). By Lemma 2.16, $\overline{H^*}$ is a vertex-facet incidence hypergraph of a polyhedron P, i.e., there exists a polyhedron P, such that the nodes of $\overline{H^*}$ correspond to the vertices of P and the edges correspond exactly to the vertex sets of facets of P. It follows that $\mathcal{H}(\mathcal{AW}(m,k))$ is a vertex-facet incidence hypergraph of P. Since $2 \leq t < m$, we have $0 < k < m - 1$. Furthermore $k > 1$, because $\mathcal{H}(\mathcal{AW}(m,1))$ can only be a vertex-facet hypergraph if $m = k = 1$, and this case is excluded by $2 \leq t < m$. Then by Proposition 3.28, P is a polygon, i.e., $k = 2$ ($t = m-2$). Note that the simplex case, where we would have $k = m - 1$, cannot arise.

Clearly, for all possible values of the above parameters, examples of infeasible inequality systems exist, which proves sufficiency. □

This proposition implies that only two types of generalized antiwebs can arise as induced hypergraphs of IIS-hypergraphs. In particular, the only generalized cliques that can occur are those with $m = t$, namely those corresponding to single IISs. For generalized odd holes the only cases that can arise are those with $t = m-2$. Finally, all generalized antiholes are ruled out since the equation $m = q\,t + 1$ holds if and only $m = (m-2)^2 + 1$ (if $t = q = m - 2$), or $m = m^2 + 1$ (if $t = q = m$). Both equations are never satisfied.

To determine in which cases facets arise from generalized antiwebs, we need the following results:

Lemma 1.32 (Laurent [80]). The inequality $\sum_{s\in S} y_s \leq \lfloor m(q-1)/t \rfloor$ arising from a generalized antiweb is valid and defines a facet of the independence system polytope $P_{IS}(\mathcal{AW}(m,t,q))$ if and only if $m = t$ or t does not divide $m(q-1)$.

Note that the right hand side of the above inequality is the rank of the independence system defined by $\mathcal{AW}(m,t,q)$ (see [80]) and hence it is a rank inequality.

Let $\mathcal{C}$ be the circuits of an independence system $(S, \mathcal{I})$. For any $S' \subseteq S$, let $\mathcal{C}_{S'} = \{C \in \mathcal{C} : C \subseteq S'\}$ denote the family of circuits contained in S'.

Lemma 1.33 (Laurent [80]). The rank inequality $\sum_{s\in S'} y_s \leq r(S')$ induces a facet of $P_{IS}(\mathcal{C})$ if and only if S' is closed and it induces a facet of $P_{IS}(\mathcal{C}_{S'})$.

Altogether, we obtain the following characterization of the rank facets of P_{FS} that can arise from generalized antiwebs.

Theorem 1.34. Let Σ be an infeasible inequality system with m inequalities and let $\mathcal{C}$ be the IISs of Σ. Let $S \subseteq [m]$ and assume $\mathcal{C}_S = \mathcal{AW}(|S|, t)$ for some $2 \leq t \leq |S|$. The rank inequality

$$\sum_{e \in S} y_e \leq \left\lfloor \frac{|S|(q-1)}{t} \right\rfloor \tag{1.5}$$

defines a facet of $P_{FS}(\Sigma)$ if and only if $q = t$ and one of the following holds

(i) $t = |S|$ (IIS-inequality)
(ii) S is closed, $t = |S| - 2$, and $t \neq 2$ (i.e., $|S| \neq 4$).

Proof. By Proposition 1.31, there are two cases where $\mathcal{AW}(|S|, t)$ can arise as an induced hypergraph of an IIS-hypergraph (i.e., as a deletion minor, see Section 2.6). By Lemma 1.30, in both cases necessarily $q = t$.

$\boldsymbol{t = |S|}$ **:** In this case $\mathcal{AW}(|S|, t)$ consists of a single circuit (IIS). Since, by Proposition 1.27, S is closed, this gives (together with Lemma 1.33) another proof that rank facets arising from IISs define facets.

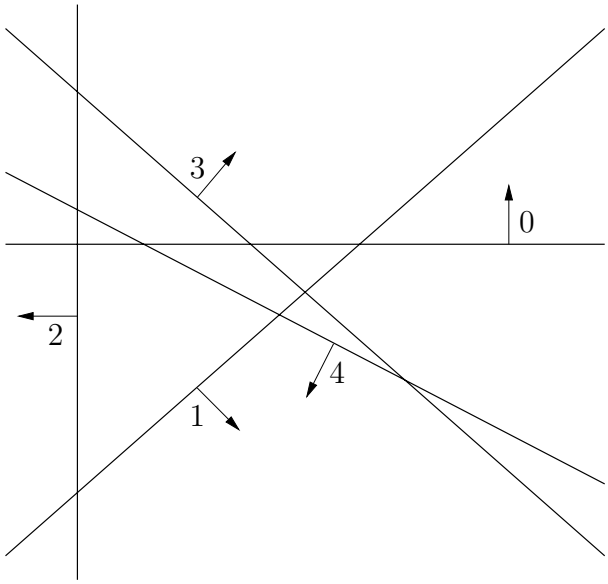
$\boldsymbol{t = |S| - 2}$ **:** By Lemma 1.32, inequality (1.5) defines a facet for the polytope $P_{IS}(\mathcal{C}_S) = P_{IS}(\mathcal{AW}(|S|, t))$ if and only if t does not divide

$$|S|(t-1) = (t+2)(t-1) = t^2 + t - 2.$$

Clearly, this can only be the case if $t = 1$ (which is not feasible) or $t = 2$. Therefore by Lemma 1.33, inequality (1.5) defines a facet of P_{FS} if and only if S is closed and $t \neq 2$. □

Example 1.35. Figure 1.4 gives an example of an infeasible system with five inequalities in dimension 2 (see also [96]). Its IISs form an $\mathcal{AW}(5, 3)$. The inequalities are indexed by $0, 1, 2, 3, 4$. In the corresponding polytope $P_{FS}(\Sigma)$ the variables are numbered likewise. Its full description is given by the following facets:

- Trivial bounds: $0 \leq y_i \leq 1$, for $0 \leq i \leq 4$.
- The IIS-inequalities: $\sum_{i \in S'} y_i \leq 2$ for $S' = \{0, 1, 2\}$, $\{1, 2, 3\}$, $\{2, 3, 4\}$, $\{3, 4, 0\}$, $\{4, 0, 1\}$.
- The rank inequality $y_0 + y_1 + y_2 + y_3 + y_4 \leq 3$, arising from the unique generalized antiweb.



	s_0	s_1	s_2	s_3	s_4
S_0	1	1	1		
S_1		1	1	1	
S_2			1	1	1
S_3	1			1	1
S_4	1	1			1

	S_0^*	S_1^*	S_2^*	S_3^*	S_4^*
$\mathcal{E}_{s_0}^*$	1			1	1
$\mathcal{E}_{s_1}^*$	1	1			1
$\mathcal{E}_{s_2}^*$	1	1	1		
$\mathcal{E}_{s_3}^*$		1	1	1	
$\mathcal{E}_{s_4}^*$			1	1	1

Figure 1.4: Top: An infeasible linear inequality system, whose IISs $\{0,1,2\}$, $\{1,2,3\}$, $\{2,3,4\}$, $\{3,4,0\}$, and $\{4,0,1\}$ form a generalized antiweb $\mathcal{AW}(5,3)$. The arrows point into the corresponding halfspaces. **Bottom left:** Edge-node incidence matrix of $\mathcal{H}(\mathcal{AW}(5,3))$. **Bottom right:** Edge-node incidence matrix of the dual hypergraph $\mathcal{H}(\mathcal{AW}(5,3))^*$, according to the notation of the proof of Proposition 1.33. This matrix is the transpose of the matrix to the left. The incidence matrix of the complement hypergraph is a vertex-facet incidence matrix of a polygon.

Chapter 2

IIS-Hypergraphs

In this chapter we investigate IIS-hypergraphs, i.e., hypergraphs whose nodes correspond to the inequalities of some infeasible inequality system Σ and whose edges correspond to the IISs of Σ (see Definition 1.1). Hence we look beyond the properties of a single IIS at the structure of *all* IISs of Σ.

First, we review known results about IIS-hypergraphs and discuss how to generate the IIS-hypergraph corresponding to Σ (Sections 2.1 and 2.2). Closely related are IIS-transversal hypergraphs, which are discussed in Section 2.3. In Section 2.4, applying Theorem 1.12, we examine the relation of IIS-hypergraphs to vertex-facet incidences of polyhedra. These results are fundamental for Section 1.3.3, where we study facets of the polytope $P_{FS}(\Sigma)$, the convex hull of all incidence vectors of feasible subsystems of Σ. In the last two sections of this chapter (Sections 2.5 and 2.6), the recognition of IIS-hypergraphs is discussed. Parts of this chapter appear in [9].

Before defining IIS-hypergraphs (Definition 2.1), we need some notation. Let $H = (S, \mathcal{E})$ be a finite hypergraph, i.e., S is a finite set and $\mathcal{E}$ is a family of subsets of S.[1] The elements of S are the *nodes* and the sets in $\mathcal{E}$ are the *edges* of H. All hypergraphs in this thesis will be finite and multiple edges are allowed. We do not require that $\bigcup(E \in \mathcal{E}) = S$. If there are no multiple edges in H we sometimes write the edge family as a set. H is called a *clutter* hypergraph, if no set of $\mathcal{E}$ contains any other set of $\mathcal{E}$, i.e., $\mathcal{E}$ is a *clutter*. Usually, clutter hypergraphs are called *simple* hypergraphs. A hypergraph is *nonempty*, if it contains a nonempty edge. A hypergraph $H = (S, \mathcal{E})$ is *isomorphic* to a hypergraph $H' = (S', \mathcal{E}')$ if there exists a bijection $\pi : S \to S'$ and a bijection $\tau : \mathcal{E} \to \mathcal{E}'$ such that

$$\tau(E) = \{\pi(e) \,:\, e \in E\} \quad \text{for all } E \in \mathcal{E}.$$

This relation is denoted by $H \cong H'$.

We refer to Berge [26] for hypergraph terminology and theory.

[1] We write the elements of a family in parentheses, e.g., $\mathcal{E} = (E_1, \ldots, E_k)$. Further, we use $E \in \mathcal{E}$ to mean that there exists i such that $E = E_i$, where we distinguish between different elements of $\mathcal{E}$, e.g., $(E \in \mathcal{E}) = (E_1, \ldots, E_k)$.

In this chapter, let K denote any subfield of $\mathbb{R}$. The most interesting cases occur if K is one of $\mathbb{Q}$, $\mathbb{A}$, or $\mathbb{R}$. Recall that $\mathbb{A}$ denotes the real algebraic numbers, that is, all real numbers that are roots of nonzero polynomials with integer coefficients. Unlike $\mathbb{R}$, every number in $\mathbb{A}$ can be finitely represented (cf. Lovász [82]). Moreover, by the quantifier elimination result of Tarski, each polyhedron can be realized with coefficients in $\mathbb{A}$ only. By Theorem 1.12, we can restrict attention to inequality systems with coefficients in $\mathbb{A}$. We will implicitly do so in the following, if we speak about complexity or computational results.

As in Chapter 1, let $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ be an *infeasible* inequality system, with $A \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. Remember that we usually identify the ith inequality with i and that $\mathcal{C}(\Sigma)$ denotes the set of all IISs of Σ. Also keep in mind that $|\mathcal{C}(\Sigma)|$ can be exponential in m and n (see Sections 1.1.1 and 2.2).

Definition 2.1. A hypergraph $H = (S, \mathcal{E})$, with $m := |S|$ nodes, is an *IIS-hypergraph of rank r* (over K) if there exists an infeasible inequality system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$, with $A \in K^{m\times n}$ (for some n), $\operatorname{rank}(A) = r$, and $\boldsymbol{b} \in K^m$, such that H is isomorphic to the hypergraph $\mathcal{H}(\Sigma) := (\{1, 2, \ldots, m\}, \mathcal{C}(\Sigma))$. The hypergraph H is an *IIS-hypergraph* (over K) if H is an IIS-hypergraph of rank r for some $r \geq 0$.

In this definition, infeasibility is meant with respect to $\mathbb{R}$. K is only important for the coefficients.

It turns out that some hypergraphs are only IIS-hypergraphs over some ordered subfield of $\mathbb{R}$, while this is not the case for other subfields. For instance, there exists a well known example of an 8-polytope with 12 vertices due to Perles, which is not "realizable" with rational coordinates, but by definition with real coordinates; see Section 5.5 of Grünbaum [70] and Example 6.21 in Ziegler [30]. Richter-Gebert found a 4-polytope with 33 vertices with the same property (see Section 9.2 of [99]). By Lemma 2.17, each such example produces a hypergraph that is an IIS-hypergraph over $\mathbb{R}$, but not over $\mathbb{Q}$.

At this point we should make some comments about the rank of an IIS-hypergraph.

Example 2.2. The hypergraph $H = (\{1, 2, 3, 4\}, \{\{1, 2\}, \{3, 4\}\})$ is an IIS-hypergraph of rank 2 (and hence an IIS-hypergraph), but not of rank 1 (see Figure 2.1).

Remark 2.3. In general, an IIS-hypergraph H does not uniquely determine its rank. To see this, consider the following construction starting as in Example 3.4.

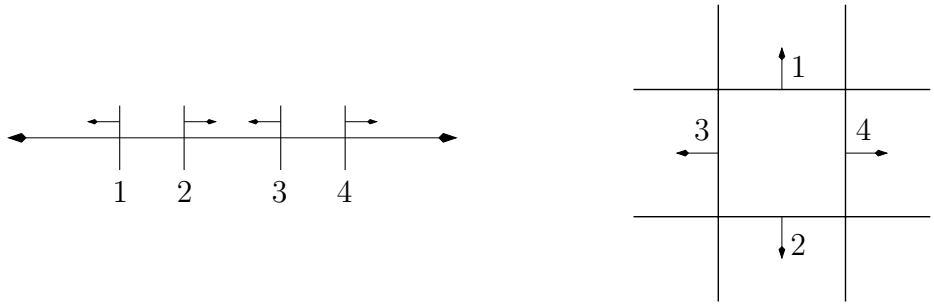


Figure 2.1: Illustration of Example 2.2. **Left:** Trying to realize H with rank $r = 1$, which always leads to at least one additional edge (IIS); in this case $\{1, 4\}$. **Right:** Realization of H with $r = 2$. Each arrow points into its corresponding halfspace.

Take Q to be a 1-simplex, C_1 to be a 3-dimensional cone with 4 facets, and C_2 to be a 4-dimensional cone with 4 facets. Then $P_1 := Q \times C_1$ is of dimension 4, and $P_2 := Q \times C_2$ is of dimension 5. Both have two vertices, $m = 6$ facets, and isomorphic vertex-facet incidences. Since P_1 and P_2 are not cones, by Lemma 1.10 they can be transformed to polyhedra P_1' and P_2', respectively, which are in the form of an alternative polyhedron as in Definition 1.9. Let $\Sigma_1 : \{A_1\boldsymbol{x} \leq \boldsymbol{b}_1\}$ and $\Sigma_2 : \{A_2\boldsymbol{x} \leq \boldsymbol{b}_1\}$ be the corresponding infeasible systems of P_1' and P_2', respectively (see Theorem 1.12). Then A_1 contains one column with $\operatorname{rank}(A_1) = 1$, and A_2 is the zero matrix. This agrees with the general condition that $m - \dim(P(\Sigma)) = \operatorname{rank}([A\ \boldsymbol{b}])$, where $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ is an infeasible system and A has m rows. These computations can easily be performed with `polymake` [63, 64].

By Lemma 2.17 Σ_1 and Σ_2 have isomorphic IIS-hypergraphs, but the ranks of the constraint matrices are different. We furthermore note that P_1' is not simple (degenerate), while the description of P_2' is nondegenerate and P_2' is therefore simple.

We need the following definition:

Definition 2.4. Let $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ be an infeasible system. The IIS-hypergraph $\mathcal{H}(\Sigma)$ is *nondegenerate* if the alternative polyhedron $P(\Sigma)$ as in Definition 1.9 is nondegenerate.

The examples in Remark 2.3 show that the nondegeneracy property depends on the concrete infeasible system Σ, i.e., this property is not preserved under isomorphism between IIS-hypergraphs.

According to hypergraph terminology, Min IIS Cover, defined in Section 1.1.1, amounts to finding a minimum cardinality (hypergraph) *transversal* of $\mathcal{H}(\Sigma)$, i.e., a subset of nodes having nonempty intersection with every edge of the IIS-hypergraph $\mathcal{H}(\Sigma)$ (see also Section 2.3).

2.1 Basic Properties of IIS-Hypergraphs

Ryan [103, 104] began the investigation of the structure of IIS-hypergraphs (over $\mathbb{R}$). IIS-hypergraphs (with no trivial IISs of cardinality one) turn out to be *bicolorable*, i.e., their nodes can be partitioned into two subsets so that neither subset contains an edge (IIS), see Greenberg [66] and Ryan [103]. Hence, each of the two subsets corresponds to a feasible subsystem. This also shows that Max FS has a 2-approximation (compare Section 1.1.3). Bicolorability, however, does not imply that a hypergraph is an IIS-hypergraph (see [103] for an example); in general many different such bipartitions exist. Because they are bicolorable, IIS-hypergraphs form a hypergraph class generalizing bipartite graphs, but do not share many properties with other such known classes. See, for instance, the figure in [104] that summarizes how IIS-hypergraphs fit into Berge's hierarchy of hypergraphs generalizing bipartite graphs.

The structure of 2-uniform IIS-hypergraphs, i.e., where each edge has size 2, can be characterized as follows. They are bipartite [103]. Moreover, following [103], call a bipartite graph $G = (A \dot\cup B, E)$ *linearly orderable* if there exist orderings $(a_1, a_2, \ldots, a_s)$ and $(b_1, b_2, \ldots, b_t)$ of the nodes in A and B, respectively, such that $\{a_i, b_j\} \in E$ implies that $\{a_i, b_{j+1}\} \in E$ and $\{a_{i-1}, b_j\} \in E$, for all $2 \leq i \leq s$, $1 \leq j \leq t-1$.

Theorem 2.5 (Ryan [103]). A hypergraph is a 2-uniform IIS-hypergraph if and only if it is a linearly orderable bipartite graph.

For 2-uniform hypergraphs (i.e., graphs) the minimum transversal problem is the vertex cover problem. For bipartite graphs, this problem can be solved in polynomial time (by König's Theorem it suffices to compute a maximum matching). Hence, a minimum transversal of a 2-uniform IIS-hypergraph can be found in polynomial time. Moreover, the greedy algorithm (successively remove nodes of maximum degree among the currently uncovered edges) always yields an optimal solution in this case (see Ryan [103]).

2.2 Generating IIS-hypergraphs

The problem to generate the IIS-hypergraph $\mathcal{H}(\Sigma)$, given the infeasible system Σ, is by Theorem 1.12 and Lemma 1.10 equivalent to the following problem (which is Problem 1 in [77]).

Vertex enumeration problem: Given a polyhedron P described by an linear inequality system, list all vertices of P (without repetition).

Let $d = \dim(P)$ and m be the number of inequalities in the description of P. We assume that P is pointed, i.e., it has at least one vertex. It is

well known that the number of vertices can be exponential ($\Omega(m^{\lfloor d/2 \rfloor})$) in the number of inequalities (e.g., dual cyclic polytopes). Therefore the number of IISs can be exponential in the number of inequalities of Σ, as well (see also Chakravarti [41]). Hence, we are interested in a *polynomial total time* algorithm to solve this problem, i.e., an algorithm whose running time can be bounded by a polynomial in the sizes of the input and the output (in contrast to a *polynomial-time* algorithm whose running time would be bounded by a polynomial just in the input size). Note that the notion of "polynomial total time" only makes sense with respect to problems which explicitly require the output to be nonredundant. One of the most important open questions in computational geometry is whether there exists a polynomial total time algorithm for the vertex enumeration problem. See Seidel [109] for an overview and Avis, Bremner, and Seidel [13] for a discussion of the complexity of most known types of algorithms for the vertex enumeration problem.

For fixed d, Chazelle [43] found an $\mathcal{O}\big(m^{\lfloor d/2 \rfloor}\big)$ polynomial-time algorithm for the vertex enumeration problem, which is optimal by the upper bound theorem of McMullen [85]. There exist algorithms which are faster than Chazelle's algorithm for small numbers of vertices n, e.g., an algorithm of Chan [42] with running time $\mathcal{O}\big(m \log n + (mn)^{1-1/(\lfloor d/2 \rfloor + 1)} \operatorname{polylog} m\big)$.

For general d, the reverse search method of Avis and Fukuda [14] solves the vertex enumeration problem for *simple* polyhedra in polynomial total time, using working space (without space for output) bounded polynomially in the input size. An algorithm of Bremner, Fukuda, and Marzetta [37] solves the problem in polynomial total time for *simplicial* polytopes. Note that these algorithms need a vertex of P to start from.

The following generalization of our situation was considered by Gurvich and Khachiyan [71]. Let $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$ be a system of polyhedra in $\mathbb{R}^n$ and let $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$ $(1 \le i_1 < i_2 < \cdots < i_k \le m)$ be a subsystem of $\mathcal{P}$. This subsystem is called a *feasible subsystem with respect to* $\mathcal{P}$, if $P_{i_1} \cap P_{i_2} \cap \cdots \cap P_{i_k} \neq \varnothing$, otherwise the system is *infeasible with respect to* $\mathcal{P}$. As before, we are interested in maximal feasible and minimal infeasible subsystems.

As a special case, the vertex enumeration problem appears twice. Once by the relation via the alternative polyhedron, discussed above, since enumerating all IISs of an infeasible linear inequality system Σ is just a special case of enumerating all minimal infeasible subsystems w. r. t. $\mathcal{P}$ (if $\mathcal{P}$ consists of the halfspaces given by Σ). On the other hand, if $\mathcal{P}$ is the collection of all facets of a pointed polyhedron, then every maximal feasible subsystem w. r. t. $\mathcal{P}$ corresponds to a vertex. Gurvich and Khachiyan [71] proved that it is co$\mathcal{NP}$-complete to decide whether a collection of maximal feasible subsystems w. r. t. $\mathcal{P}$ or a collection of minimal infeasible subsystems w. r. t. $\mathcal{P}$

is complete. Hence, unless $\mathcal{P} = \mathcal{NP}$, there exists no polynomial total time algorithm to enumerate the maximal feasible or the minimal infeasible subsystems with respect to $\mathcal{P}$.

2.3 Generating IIS-Transversal Hypergraphs

Definition 2.6. Let $H = (S, \mathcal{E})$ be a hypergraph. A subset T of S is a hypergraph *transversal* of H, if $T \cap E \neq \varnothing$ for all $E \in \mathcal{E}$. Define $\mathrm{tr}(H)$ to be the *transversal hypergraph*, i.e., the hypergraph that has S as its nodes and the *minimal* transversals (w. r. t. inclusion) of H as its edges.[2]

One can assume w. l. o. g. that H is a clutter hypergraph, since it suffices to take the maximal edges w. r. t. inclusion of H. Furthermore, $\mathrm{tr}\, H$ is always a clutter hypergraph. Under these conditions, $\mathrm{tr}(\mathrm{tr}(H)) = H$.

Definition 2.7. Let Σ be an infeasible inequality system and $\mathcal{H}(\Sigma)$ the corresponding IIS-hypergraph (see Definition 2.1). The hypergraph $\mathrm{tr}(\mathcal{H}(\Sigma))$ is called the *IIS-transversal hypergraph* of Σ.

The elements of $\mathrm{tr}(\mathcal{H}(\Sigma))$ are called *minimal IIS-transversals* or *minimal IIS-covers*; compare also the definition in Section 1.1.1.

Example 2.8. The sizes of the hypergraphs $\mathcal{H}(\Sigma)$ and $\mathrm{tr}(\mathcal{H}(\Sigma))$ can be very different: For instance, take the infeasible system given by the inequalities $x_i \leq 0$ and $x_i \geq 1$ for $i = 1, 2 \ldots, k$, where k is any fixed positive integer. This system has k IISs and 2^k minimal IIS-transversals. Its alternative polyhedron (see Definition 1.9) is a $(k-1)$-dimensional simplex.[3]

Conversely, consider the infeasible system determined by

$$x_1 + \cdots + x_{k-1} \geq 1, \quad x_1 + \cdots + x_{k-1} \geq 2$$
$$x_i \leq 0, \quad x_i \leq -1 \quad \text{for } i = 1, 2, \ldots, k-1.$$

This system has 2^k IISs, which are all of size k: They consist of one of $x_i \leq 0$ or $x_i \leq -1$ for each $i = 1, 2, \ldots, k-1$ and one of the first two inequalities. It has k IIS-transversals, all of size 2, namely either the first two inequalities or both $x_i \leq 0$ and $x_i \leq -1$, for each $i = 1, 2, \ldots, k-1$. In fact, the alternative polyhedron of this example is a k-dimensional cube (see Definition 1.9 and Example 2.25).[4]

[2] The collection of edges of $\mathrm{tr}(H)$ is sometimes called the *blocker* of H.

[3] It can be described as: $\{\, (\boldsymbol{x}, \boldsymbol{x}') \in \mathbb{R}^{2k} \,:\, \mathbb{1}\boldsymbol{x} = 1,\ \boldsymbol{x} = \boldsymbol{x}',\ \boldsymbol{x} \geq \boldsymbol{0},\ \boldsymbol{x}' \geq \boldsymbol{0} \,\}$. For each $i = 1, \ldots, k$, one of $x_i \geq 0$ and $x'_i \geq 0$ is redundant.

[4] The alternative polyhedron of the system $\{\, \mathbb{1}\boldsymbol{x} \geq 1,\ \mathbb{1}\boldsymbol{x} \geq 1,\ \boldsymbol{x} \leq \boldsymbol{0},\ \boldsymbol{x} \leq \boldsymbol{0} \,\}$ is the k-dimensional 0/1-cube.

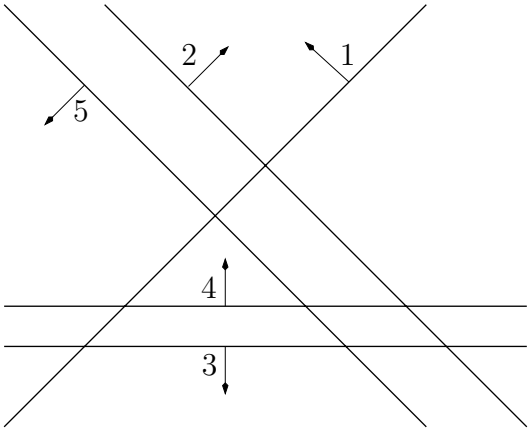


Figure 2.2: Realization of the hypergraph H discussed in Example 2.9. The arrows point into the corresponding halfspace.

Hence, as it was the case for the generation of all IISs, we are interested in an polynomial total time algorithm to generate all IIS-transversals. It is unknown if such an algorithm exists.

A question that comes to mind in this context is whether IIS-transversal hypergraphs are IIS-hypergraphs (and conversely). If this was the case and the corresponding infeasible system would be at hand or could be efficiently computed, we could use the approach of the previous section and apply vertex enumeration algorithms to generate the IIS-transversal hypergraph. The next example, however, shows that the answer to this question is negative.

Example 2.9. Consider the hypergraph H with nodes $\{1, 2, 3, 4, 5\}$ and the following edges: $\{2, 5\}$, $\{3, 4\}$, $\{1, 2, 3\}$. H is an IIS-hypergraph, which is demonstrated by the realization of H shown in Figure 2.2.

The minimal transversal hypergraph $\mathrm{tr}(H)$ of H has $\{2, 3\}$, $\{2, 4\}$, $\{3, 5\}$, and $\{1, 4, 5\}$ as its edges. It cannot be an IIS-hypergraph, since the hyperplanes corresponding to 2, 3, 4, and 5 would have to be parallel (implied by the first three edges), but the last edge $\{1, 4, 5\}$ forces 4 and 5 not to be parallel.

Conversely, H cannot be an IIS-transversal hypergraph: Otherwise the equation $H = \mathrm{tr}(\mathrm{tr}(H))$ implies that $\mathrm{tr}(H)$ has to be an IIS-hypergraph, but this cannot be the case as we just observed.

Let $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ be an infeasible inequality system, where $A \in \mathbb{R}^{m \times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. The IIS-transversal hypergraph corresponding to Σ can be generated as follows, although not in polynomial total time. We consider the

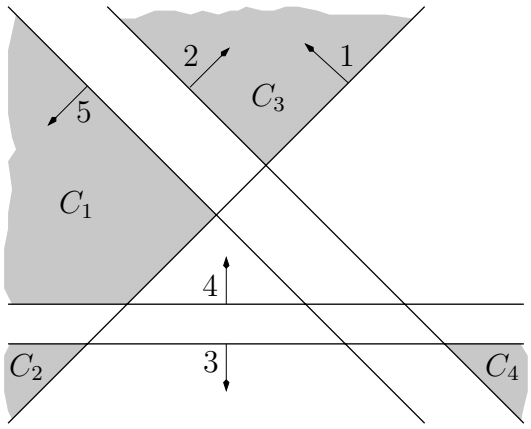


Figure 2.3: Oriented hyperplane arrangement taken from Figure 2.2, with shaded cells $C_i = C(\,([5] \setminus T_i^-, T_i^-)\,)$ for $i = 1, 2, 3, 4$, where $T_1^- = \{2, 3\}$, $T_2^- = \{2, 4\}$, $T_3^- = \{3, 5\}$, and $T_4^- = \{1, 4, 5\}$. These cells correspond to minimal IIS-transversals.

affine oriented hyperplane arrangement given by the inequalities of Σ. For each point $\boldsymbol{z} \in \mathbb{R}^n$ define

$$\begin{aligned} S^+(\boldsymbol{z}) &:= \{i \in [m] \,:\, \boldsymbol{a}^i \boldsymbol{z} < b_i\}, \\ S^=(\boldsymbol{z}) &:= \{i \in [m] \,:\, \boldsymbol{a}^i \boldsymbol{z} = b_i\}, \\ S^-(\boldsymbol{z}) &:= \{i \in [m] \,:\, \boldsymbol{a}^i \boldsymbol{z} > b_i\}, \end{aligned}$$

where $\boldsymbol{a}^i$ is the ith row of A.

The set $S^-(\boldsymbol{z})$ is just the subset of inequalities of Σ that are violated by the point $\boldsymbol{z}$. Hence, if we remove the inequalities in $S^-(\boldsymbol{z})$ from Σ, we obtain a feasible system. Therefore, each $\boldsymbol{z} \in \mathbb{R}^n$ corresponds to an IIS-transversal $S^-(\boldsymbol{z})$. Let $T := (T^+, T^-)$ be a partition of the set of inequalities $[m]$ and define $C(T) := \{\boldsymbol{z} \in \mathbb{R}^n \,:\, S^+(\boldsymbol{z}) = T^+,\ S^-(\boldsymbol{z}) = T^-\}$. We call $C(T)$ a *cell* if it is nonempty. Clearly, $C(T)$ is the interior of an (possibly unbounded) polyhedron. Then an IIS-transversal $S \subseteq [m]$ corresponds to the cell $C(\,([m] \setminus S,\, S)\,)$.

Let $B(T) \subseteq [m]$ be the inequalities that define the boundary of $C(T)$. If we take $\boldsymbol{z} \in C(T)$ and if $B(T) = T^+$, starting from $\boldsymbol{z}$ we cannot leave the closure of $C(T)$ without violating more inequalities, i.e., T^- is a *minimal* IIS-transversal. Hence, if Σ contains no implicit equations, minimal IIS-transversals correspond to cells $C(T)$ such that $B(T) = T^+$; see Figure 2.3.

If we are given Σ, containing no implicit equations, we can generate the IIS-transversal hypergraph by first generating all cells with the reverse search algorithm by Avis and Fukuda [15] and then find the ones that correspond

to minimal IIS-transversals. In fact, the tree algorithm of Greer [68] computes all cells that correspond to minimal IIS-transversals; he calls such cells hills, as they correspond to local optima w. r. t. the objective function counting the inequalities satisfied by a point $\boldsymbol{z} \in \mathbb{R}^n$. This algorithm runs in $\mathcal{O}(n \cdot m^n/2^{n-1})$ time and also works if Σ contains implicit equations. For fixed dimension n, both are are polynomial-time algorithms.

Given the infeasible system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ and a (partial) list of IISs and minimal IIS-transversals of Σ, an algorithm of Gurvich and Khachiyan [71] produces a new IIS or IIS-transversal or concludes that none exists in subexponential time: $k^{o(\log k)}$ if k is the sum of the number of known IISs and IIS-transversals.

With respect to counting minimal IIS-transversals only, we have the following result:

Proposition 2.10. Given an infeasible system $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$, counting the number of minimal IIS-transversals is $\#\mathcal{P}$-complete.

Proof. Let $D = (V, A)$ be a directed graph. A feedback arc set of D is a subset A' of the arcs such that $(V, A \setminus A')$ contains no directed cycle. Schwikowski and Speckenmeyer [108] prove that computing the number of minimal feedback arc sets of a directed graph is $\#\mathcal{P}$-complete.

Sankaran [105] introduced the system $\Sigma : \{x_u - x_v \leq -1 : (u, v) \in A\}$, which is feasible if and only if D contains no directed cycle. He proved that if Σ is infeasible, then IIS-transversals for Σ correspond to feedback arc sets of D. It directly follows that counting the number of minimal IIS-transversals is $\#\mathcal{P}$-complete. □

If the input is the IIS-hypergraph $\mathcal{H}(\Sigma)$ (and no geometrical information), the problem to compute the IIS-transversal hypergraph of Σ can be reduced to the following.

Hypergraph transversal problem: Given a hypergraph H, generate its transversal hypergraph $\mathrm{tr}(H)$.

This problem has lots of applications, most of them in computer science, see Eiter and Gottlob [54] for many examples. One interesting application is to generate the minimal non-faces of a simplicial complex, which are needed to compute the Stanley-Reisner ring, cf. Stanley [110] (see below for the special case where the simplicial complex is shellable). The hypergraph transversal problem is closely related to the

Hypergraph transversal completeness problem:
Given a hypergraph H and $\mathcal{T} \subseteq \mathrm{tr}(H)$, either decide that $\mathcal{T} = \mathrm{tr}(H)$ or compute $T \in \mathrm{tr}(H) \setminus \mathcal{T}$.

If one can solve the hypergraph transversal completeness problem in polynomial time then one can solve the hypergraph transversal problem in polynomial total time and conversely (the back direction follows from a standard simulation argument). Many interesting problems, for which no polynomial-time algorithm could be found (despite numerous efforts), can be reduced to the hypergraph transversal completeness problem. Therefore, it is assumed that this problem is "hard" (see Bioch and Ibaraki [28]). It is, however, unlikely to be $\mathcal{NP}$-hard, since there exists a subexponential $k^{o(\log k)}$ algorithm of Fredman and Khachiyan [57], where k is the size of the input, i.e., the sum of the sizes of H and of T. Many special cases of the hypergraph transversal completeness problem are solvable in polynomial time. For instance, this is the case if the sizes of the edges are bounded (by a constant) or if the degree of each node is bounded (i.e., the number of edges that contain a node is bounded), see Eiter and Gottlob [54].

Ryan [104] showed that one can generate the IIS-transversal hypergraph in polynomial time, if the IIS-hypergraph $\mathcal{H}(\Sigma)$ is nondegenerate, i.e., if the alternative polyhedron $P(\Sigma)$ is nondegenerate. For this, she first shows that in this case there exist only polynomially many minimal transversals in the size of $\mathcal{H}(\Sigma)$ and then gives an algorithm which generates all minimal transversals. The proof, however, is a bit complicated and (without explanation) implicitly uses a dual shelling of $P(\Sigma)$. We will give a new proof of this result in Section 2.3.2, since it is simpler and gives insight into the structure of nondegenerate IIS-hypergraphs, but we first have to treat non-faces of shellable simplicial complexes.

2.3.1 Non-Faces of Simplicial Complexes

A (finite abstract) *simplicial complex* is a finite hypergraph (V, Δ) with no multiple edges that has the property that $G \subset F \in \Delta$ implies that $G \in \Delta$. The elements of V are called *vertices*, the elements of Δ are the *faces*, and a set $S \subseteq V$ with $S \notin \Delta$ is a *non-face* of (V, Δ). If the vertex set V is clear from the context, we call the simplicial complex itself Δ. The *dimension* of a face $F \in \Delta$ is $\dim F := |F| - 1$ and the dimension of the complex Δ is $\dim \Delta := \max\{\dim F : F \in \Delta\}$. The inclusion-wise maximal elements of Δ are called *facets*. The simplicial complex is *pure* if all facets have the same dimension. We refer to Björner [29] for more information; note that in contrast to this reference we allow $\varnothing \in \Delta$ and that facets are defined differently.

A simplicial complex Δ with s facets is shellable, if there exists an ordering of its facets $F_1, F_2, \ldots, F_s$, such that the following property holds. For every

index i and k with $1 \leq i < k \leq s$, there exists j with $1 \leq j < k$ and $x \in F_k$ such that $F_i \cap F_k \subseteq F_j \cap F_k = F_k - \{x\}$. Such an ordering is called a shelling order. See Björner and Wachs [31] and Ziegler [115] for more information. This definition does not require Δ to be pure, but we will only use it for pure simplicial complexes. We need the following Lemma, which was proved by Volker Kaibel.

Lemma 2.11. Let (V, Δ) be a shellable simplicial complex with s facets. Then (V, Δ) has at most $s \cdot |V|$ minimal non-faces.

Proof. Let $F_1, F_2, \ldots, F_s$ be a shelling order of the facets of (V, Δ). For $k = 1, \ldots, s$, define $\Delta_k := \{S \subseteq V : \exists\, i \leq k \text{ such that } S \subseteq F_i\}$. Then (V, Δ_k) is a simplicial complex.

We show by induction on k that the number of minimal non-faces of (V, Δ_k) is at most $k \cdot |V|$. For $k = 1$, the number of non-faces of (V, Δ_1) is $|V \setminus F_1| \leq |V|$. Hence, for the following assume $k > 1$.

Define the so-called *restriction set* $R_k := \{v \in F_k : F_k - \{v\} \in \Delta_{k-1}\}$. For every set $G \in \Delta_k \setminus \Delta_{k-1}$, we have $R_k \subseteq G \subseteq F_k$ (see Proposition 2.5 of Björner and Wachs [31] or Section 8.3 in Ziegler [115]). We show that by adding all subsets of F_k to Δ_{k-1} (and hence obtaining (V, Δ_k)), at most $|V|$ new minimal non-faces are introduced (while others are lost). Assume that S is a new minimal non-face in (V, Δ_k), which was not minimal in (V, Δ_{k-1}). Clearly, S is a non-face in (V, Δ_i) for every $i \leq k$.

We claim that $R_k \subset S$. Since S is not minimal in (V, Δ_{k-1}), there exists a set $S' \subset S$ which is a minimal non-face in (V, Δ_{k-1}). But since S is minimal, S' is a face in Δ_k. Because $S' \in \Delta_k \setminus \Delta_{k-1}$, it follows that $R_k \subseteq S'$. Therefore we have $R_k \subseteq S' \subset S$, as claimed.

We have $S \not\subseteq F_k$ (otherwise S would be a face in Δ_k), $R_k \subset S$, and $R_k \subseteq F_k$. Consider any $v \in S \setminus R_k$. Since S is a minimal non-face of (V, Δ_k), the set $S - \{v\}$ is a face of Δ_k. But $S - \{v\} \supseteq R_k$ and since Δ_{k-1} does not contain R_k, $S - \{v\}$ is not a face in Δ_{k-1}. Therefore, $S - \{v\} \subseteq F_k$ and hence $S \setminus F_k = \{v\}$, which implies that v is unique, i.e., $S = R_k \cup \{v\}$. This proves that $R_k \cup \{v\}$ are the only candidates for new non-faces when proceeding from (V, Δ_{k-1}) to (V, Δ_k); and there are at most $|V|$ of them. □

In contrast to this result, the number of facets need not be polynomially bounded in the number of non-faces and vertices of a shellable simplicial complex. For instance, take the boundary complex (V, Δ) of the d-crosspolytope as a simplicial complex, which is shellable since boundary complexes of simplicial polytopes are shellable (see Section 8.2 of Ziegler [115]). It has $2d$ vertices and d non-faces (the "diagonals"), but 2^d facets.

If a shelling is available, the proof of Lemma 2.11 immediately suggests a polynomial-time algorithm for computing the minimal non-faces. This yields a polynomial-time algorithm for the hypergraph transversal problem if a shelling of the hypergraph is available, i.e., if we know a shelling of the simplicial complex whose facets are the edges of the hypergraph (if it is shellable at all).

Boros, Crama, Ekin, Hammer, Ibaraki, and Kogan [36] prove a result similar to Lemma 2.11 and provide a polynomial-time algorithm for computing the minimal non-faces. They first define a shellable disjunctive normal form and then work in terms of (monotone) boolean functions.

Consider the problem to decide whether a simplicial complex Δ given by the list of its facets is shellable. The complexity status of this problem is open, even if the input is the set of all faces of Δ (see Problem 34 of [77]). If we are given an ordering of the facets of Δ, it is easy to test whether this is a shelling order in polynomial time. Hence, the former problem is in $\mathcal{NP}$.

2.3.2 The Transversal Hypergraph of Nondegenerate IIS-Hypergraphs

We return to the case of computing the IIS-transversal hypergraph for an infeasible inequality system whose alternative polyhedron is nondegenerate. Let $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ be an infeasible system, where as usual $A \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. Let $P = \{\boldsymbol{y} \in \mathbb{R}^m : \boldsymbol{y}A = \boldsymbol{0},\ \boldsymbol{y}\boldsymbol{b} = -1,\ \boldsymbol{y} \geq \boldsymbol{0}\}$ be the alternative polyhedron of Σ (Definition 1.9). Assume that P is nondegenerate. For a vertex v of P, let $F(v)$ be the indices of facets of P that contain v. Since P is nondegenerate, each inequality $y_j \geq 0$ $(1 \leq j \leq m)$ defines a facet and P is simple, i.e., $|F(v)| = \dim P$ for all vertices v of P. Furthermore, every k-subset of $F(v)$ defines a unique face of dimension $\dim(P) - k$. Let $\Delta(P)$ be the simplicial complex with $\{1, 2, \ldots, m\}$ as vertex set and $F(v)$ for every vertex v of P as facets. Then $\Delta(P)$ (with an adjoined top element $\hat{1}$) is anti-isomorphic to the face lattice of P. Hence, every set in $\Delta(P)$ corresponds to a (unique) face of P. If P is a polytope, $\Delta(P)$ is isomorphic to the boundary complex of the polar of P. We need a simple Lemma of Ryan.

Lemma 2.12 (Ryan [103]). Let Σ be an infeasible system with m inequalities and P be its alternative polyhedron. A minimal IIS-transversal $T \subseteq [m]$ corresponds to a minimal set of faces of P defined by $y_j \geq 0$, $j \in T$ that have an empty intersection and conversely.

In our situation, we want to compute the minimal sets of *facets* of P that have empty intersection. Each such set corresponds to a minimal subset $S \subseteq [m]$ with the property that $S \notin \Delta(P)$, i.e., S is a non-face of $([m], \Delta(P))$.

Therefore, to apply Lemma 2.11 and enumerate all minimal IIS-transversals, we need a shelling of $\Delta(P)$.

The boundary of a simplicial polytope is a simplicial complex, which is shellable by a famous result of Bruggesser and Mani [38] (see also Section 8.2. of Ziegler [115]). Here we are in the dual situation, but have to consider unbounded simple polyhedra, too. Nonetheless, the same ideas can be used to show that $\Delta(P)$ is shellable, as follows. The linear objective function given by $\boldsymbol{c} = (\varepsilon, \varepsilon^2, \ldots, \varepsilon^n)$, for some $\varepsilon > 0$, defines a linear ordering on the vertices (IISs) of P, since $\boldsymbol{c}$ is in general position w. r. t. the vertices of P, if ε is small enough. If the vertices of P are $v_1, v_2, \ldots, v_s$, sorted by increasing objective value w. r. t. $\boldsymbol{c}$, then $F(v_1), F(v_2), \ldots, F(v_s)$ is a shelling order of $\Delta(P)$ (see Ball and Provan [21]).

The only known way to efficiently generate a shelling is by using geometry as just explained. In fact, this defines an abstract objective function (Problem 17 of [77]). Hence, we either assume that we are given $\mathcal{H}(\Sigma)$ together with P, from which we can efficiently compute a shelling of $\Delta(P)$ as above, or we are given $\mathcal{H}(\Sigma)$ together with a shelling of $\Delta(P)$. Applying Lemma 2.11 and Theorem 1.12 yields:

Theorem 2.13 (Ryan [104]). Let Σ be an infeasible system with m inequalities and P be its alternative polyhedron. If P is nondegenerate, then the number of minimal transversals is at most $m \times \ell$, if ℓ is the number of IISs (vertices of P). Furthermore, if we are given a shelling of $\Delta(P)$, $\mathrm{tr}(\mathcal{H}(\Sigma))$ can be computed in polynomial time in the size of Σ and ℓ.

Since ℓ can be exponential in the size of Σ, even for simple polyhedra (see Section 2.2), this result does not show that Min IIS Cover is solvable in polynomial time for systems whose alternative polyhedron is nondegenerate. In fact, it seems to be open whether Min IIS Cover is hard for these cases. All $\mathcal{NP}$-hardness proofs for Min IIS Cover that we know of construct systems that are not of this type.

2.4 Relation to Vertex-Facet Incidence Hypergraphs

Theorem 1.12 provides a connection between the structure of the IISs of any given infeasible system and the vertex-facet incidences of its alternative polyhedron. In this section, we investigate this relation on a combinatorial level. This is used in the next section to address the problem of recognizing whether a given hypergraph is an IIS-hypergraph. The techniques used here are also fundamental for Section 2.6 and Section 1.3.3. We need the following concepts for hypergraphs.

Let $H = (S, \mathcal{E})$ be a hypergraph. For $E \in \mathcal{E}$ define $\overline{E} := S \setminus E$ to obtain the *complement hypergraph* $\overline{H} := (S, \overline{\mathcal{E}})$, where $\overline{\mathcal{E}} := (\overline{E} \,:\, E \in \mathcal{E})$.

Definition 2.14 (see Berge [26]). Let $H = (S, \mathcal{E})$ be a hypergraph. The *dual hypergraph* H^* of H contains a distinct node E^* corresponding to each edge $E \in \mathcal{E}$ and has the family $\mathcal{E}^* := (\mathcal{E}_s^* \,:\, s \in S)$ as edges, where for all $s \in S$, $\mathcal{E}_s^* := \{E^* \,:\, s \in E,\ E \in \mathcal{E}\}$.

This operation corresponds to the transposition of the edge-node incidence matrix of H. See Figure 1.4 on page 36 for an example. One can easily verify that $H^{**} \cong H$ and $(\overline{H})^* \cong \overline{(H^*)}$.

Definition 2.15. Let P be a pointed polyhedron with vertex set V. Let $F_1, \ldots, F_m$ be the facets of P and let $\operatorname{vert}(F) := \{v \in V \,:\, v \in F\}$ be the vertex set of a face F of P. A hypergraph $H = (S, \mathcal{E})$ is a *vertex-facet incidence hypergraph* of P, if H is isomorphic to $\mathcal{H}(P) := (V, (\operatorname{vert}(F_1), \ldots, \operatorname{vert}(F_m)))$. A hypergraph H is a *vertex-facet incidence hypergraph (of dimension d)*, if it is a vertex-facet incidence hypergraph of a polyhedron P (of dimension d).

Note that in IIS-hypergraphs there may exist nodes that are not contained in any edge, which is not possible for vertex-facet incidence hypergraphs (except for 0-dimensional polytopes).

Recall that K is a subfield of $\mathbb{R}$. We have the following relation:

Lemma 2.16. Let $\Sigma : \{A\boldsymbol{x} \leq \boldsymbol{b}\}$ be an infeasible system, where $A \in K^{m \times n}$ and $\boldsymbol{b} \in K^m$. If $\overline{\mathcal{H}(\Sigma)^*}$ is a clutter hypergraph, then it is a vertex-facet incidence hypergraph of the alternative polyhedron corresponding to Σ with a description over K.

Proof. According to Theorem 1.12, the IISs (elements of $\mathcal{C}(\Sigma)$) are the supports of the vertices of the alternative polyhedron

$$P = \{\boldsymbol{y} \in \mathbb{R}^m \,:\, \boldsymbol{y}A = \boldsymbol{0},\ \boldsymbol{y}\boldsymbol{b} = -1,\ \boldsymbol{y} \geq \boldsymbol{0}\}.$$

Let $C \in \mathcal{C}(\Sigma)$ be an IIS, and let $\boldsymbol{v}$ be the vertex of P associated with C. The complement of the support of $\boldsymbol{v}$ is $\overline{C}$; and it determines which inequalities $y_j \geq 0$, $1 \leq j \leq m$, are satisfied by $\boldsymbol{v}$ with equality. Hence, $\overline{C}$ is the set of all faces defined by these inequalities containing $\boldsymbol{v}$. It follows that each set in $(\overline{\mathcal{C}(\Sigma)})^* = \overline{\mathcal{C}(\Sigma)^*}$ corresponds to the vertex set of a face defined by $y_j \geq 0$ for some $1 \leq j \leq m$. Furthermore, for each facet of P there exists j $(1 \leq j \leq m)$ such that $y_j \geq 0$ defines this facet. Since $\overline{\mathcal{C}(\Sigma)^*}$ is a clutter, no vertex set of a face of this type contains another. Altogether this implies that each $y_j \geq 0$ defines a facet of P. Thus, $\overline{\mathcal{H}(\Sigma)^*}$ is a vertex-facet incidence hypergraph of P. □

The reverse direction is as follows:

Lemma 2.17. Let $H = (V, \mathcal{F})$ be a nonempty vertex-facet incidence hypergraph of a pointed polyhedron P (with a description over K) that is not a cone. Then $\overline{H^*}$ is an IIS-hypergraph (over K).

Proof. Since P is not a cone, by Lemma 1.10, there exists a polyhedron $P' = \{\boldsymbol{y} \in \mathbb{R}^m : \boldsymbol{y}A = \boldsymbol{0},\ \boldsymbol{y}\boldsymbol{b} = -1,\ \boldsymbol{y} \geq \boldsymbol{0}\}$ which is affinely equivalent to P and each face of P' defined by $y_j \geq 0$ is a facet ($1 \leq j \leq m$). By Remark 1.11, we have $A \in K^{m \times n}$ and $\boldsymbol{b} \in K^m$ for appropriate n and m. We can identify V with the set of vertices of P' and each set of $\mathcal{F}$ is the vertex set of a facet of P'. An edge F of H^* corresponds to the set of facets which contain a specific vertex v of P', i.e., $F = \mathcal{F}_v^* = \{G^* : v \in G,\ G \in \mathcal{F}\}$. If we identify $[m]$ with the set of facets, $\overline{F}$ is the support of v. Thus, by Theorem 1.12, $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ is an infeasible system whose IISs are in one-to-one correspondence with the edges of $\overline{H^*}$. □

Note the slight asymmetry between the assumptions of Lemma 2.16 and Lemma 2.17, which is due to the fact that vertex-facet incidences cannot capture all information about the face lattice of unbounded polyhedra (see Section 3.2). Restricting attention to hypergraphs H for which $\overline{H^*}$ is a nonempty clutter hypergraph yields the following result.

Corollary 2.18. Let $H = (S, \mathcal{E})$ be a finite hypergraph such that $\overline{H^*}$ is a nonempty clutter hypergraph. Then H is an IIS-hypergraph if and only if $\overline{H^*}$ is a vertex-facet incidence hypergraph of a polyhedron.

Proof. An IIS-hypergraph H is isomorphic to $\mathcal{H}(\Sigma)$ for some infeasible system Σ. By Lemma 2.16, $\overline{H^*} \cong \overline{\mathcal{H}(\Sigma)^*}$ is a vertex-facet incidence hypergraph.

If $\overline{H^*}$ is a vertex-facet incidence hypergraph of a polyhedron P and it is a clutter hypergraph, then P cannot be a cone, unless $P = \{\boldsymbol{0}\}$ (which is excluded since in this case $\overline{H^*}$ would have no edges). Thus by Lemma 2.17, H is an IIS-hypergraph. □

2.5 IIS-Hypergraph Recognition

In this subsection we address the problem of recognizing IIS-hypergraphs, which was first raised by Ryan [104].

IIS-hypergraph recognition problem over K: Given a hypergraph H, is H an IIS-hypergraph over K?

We prove $\mathcal{NP}$-hardness of this problem by polynomial-time reduction from the Steinitz problem for polytopes defined below.

We follow the definitions and terms of posets and lattices in Ziegler [115]. The *face lattice* of a polytope P is its set of faces, ordered by inclusion, with the meet defined by intersection. It is well-known that the face lattice of P has a rank function $r(\cdot)$, satisfying $r(F) = \dim F + 1$ for every face F, and is both atomic and coatomic (see [115]). Two polytopes with isomorphic face lattices are *combinatorially equivalent*.

Steinitz problem over K: Given a lattice L, does there exist a polytope $P \subset \mathbb{R}^d$ (for some d) with vertices in K^d such that the face lattice of P is isomorphic to L?

If the answer is affirmative, L is *realizable* as a polytope. In this case d can be assumed to be the rank of L minus 1. See Bokowski and Sturmfels [34] for related material.

We need a special lattice construction arising from hypergraphs. Consider an arbitrary hypergraph $H = (S, \mathcal{E})$. Define the poset $\mathcal{L}(H)$ as the set of all nonempty intersections of sets in $\mathcal{E}$, ordered by inclusion. Furthermore, adjoin a minimal element $\hat{0}$ and a maximal element $\hat{1}$ to $\mathcal{L}(H)$. Since $\mathcal{L}(H)$ is bounded and has a meet (defined by intersection), it is a lattice. Note that the size of $\mathcal{L}(H)$, i.e., its number of elements, can be exponential in the size of H. If H is a vertex-facet incidence hypergraph of a polyhedron P, we exactly have $\mathcal{L}(H) = \hat{\mathcal{V}}(P)$ (defined in Section 3.3). Moreover, if P is a polytope, $\mathcal{L}(H)$ is isomorphic to the face lattice of P. This follows from the fact that all faces of P are determined by their vertex sets and also by the set of facets they are contained in.

Conversely, consider an arbitrary ranked, atomic, and coatomic lattice L. Then let V be the set of atoms of L and for each coatom F, define the set $E_F := \{v \in V : v \text{ is below } F \text{ in } L\}$. Further define the hypergraph $\mathcal{H}(L) := (V, \{E_F : F \text{ coatom of } L\})$. Note that, since L is atomic, $\mathcal{H}(L)$ is a clutter hypergraph by construction. If L is the face lattice of a polytope, then $\mathcal{H}(L)$ is a vertex-facet incidence hypergraph.

Finally we can prove the main reduction used for the proof of $\mathcal{NP}$-hardness of the IIS-hypergraph recognition problem.

Theorem 2.19. For any subfield K of $\mathbb{A}$, there is a polynomial-time Turing reduction from the Steinitz problem (over K) to the IIS-hypergraph recognition problem (over K).

Proof. We show that for any instance of the Steinitz problem, given by a lattice L, we can construct in polynomial time a special instance of the IIS-hypergraph recognition problem, given by a clutter hypergraph H, such that the answer to L is affirmative if and only if the answer to H is affirmative.

If L is realizable as a polytope P, it is isomorphic to the face lattice of P and hence necessarily ranked, atomic, and coatomic. Furthermore, by Theorem 3.11 and Equation (3.2) we have that $\widetilde{\chi}(\mathcal{V}(P)) = \mu(\hat{\mathcal{V}}(P)) = \mu(L)$ is nonzero[5], where $\mu(L)$ is the Möbius function of L (see (3.1) of Section 3.3). These conditions are used as follows.

First perform Test 1: Check whether L is ranked, atomic, and coatomic. This test can be performed in polynomial time in $|L|$. If L passes the test, take $H = \overline{\mathcal{H}(L)^*}$, which can be constructed in polynomial time in $|L|$. If L fails the test, let H be any hypergraph which is not an IIS-hypergraph, e.g., take $H = (\{1,2,3\}, (\{1,2\}, \{2,3\}, \{1,3\}))$.

Then perform Test 2: Compute $\mu(L)$ in polynomial time in $|L|$ (see Corollary 3.12) and check whether it is nonzero. If $\mu(L) = 0$, then replace H by any hypergraph which is not an IIS-hypergraph. The resulting H after both tests is the input to the IIS-hypergraph recognition problem.

To prove correctness, assume that the answer to the IIS-hypergraph recognition problem for H is affirmative, i.e., H is an IIS-hypergraph. In this case, L passed both tests. As noted above, the atomicity of L implies that $\mathcal{H}(L) = \overline{H^*}$ is a clutter hypergraph. By Lemma 2.16, $\overline{H^*}$ is a vertex-facet incidence hypergraph of a polyhedron P. By construction, $\mathcal{L}(\overline{H^*}) = \mathcal{L}(\mathcal{H}(L))$ is isomorphic to L (since L is atomic and coatomic).

First assume that P is a polytope. Then $\mathcal{L}(\overline{H^*})$ is isomorphic to the face lattice of P and hence so is L, i.e., the answer to the Steinitz problem for L is affirmative.

Second, if P is an unbounded polyhedron, then $\overline{H^*}$ is a vertex-facet incidence hypergraph of an unbounded polyhedron and hence, by Theorem 3.11, it follows that $\mu(\mathcal{L}(\overline{H^*})) = \widetilde{\chi}(\hat{\mathcal{V}}(P)) = 0$. Since $\mathcal{L}(\overline{H^*})$ is isomorphic to L, we have $\mu(L) = 0$. But in this case we replaced the input by an instance which is not an IIS-hypergraph; a contradiction. This proves that if H is an IIS-hypergraph then L is the face lattice of a polytope.

Conversely, assume that the answer to the Steinitz problem for L is affirmative. Then there exists a polytope P such that L is isomorphic to the face lattice of P and hence, by construction, $\overline{H^*}$ is a vertex-facet incidence hypergraph of P. The polytope P is not a cone unless $P = \{\mathbf{0}\}$, a case which can be easily identified and discarded (see also Corollary 2.18). By applying Lemma 2.17 to $\overline{H^*}$, it follows that H is an IIS-hypergraph. □

Given a set of polynomials $f_1, \ldots, f_r, g_1, \ldots, g_s, h_1, \ldots, h_t \in \mathbb{Z}[x_1, \ldots, x_l]$,

[5]The question of how to (efficiently) distinguish between polytopes and polyhedra from their vertex-facet incidences triggered the investigations of Section 3.3. In particular, at first, it was not even clear whether a polytope and an unbounded polyhedron could have the same vertex-facet incidences.

the problem to decide whether the polynomial system $f_1 = 0, \ldots, f_r = 0$, $g_1 \geq 0, \ldots, g_s \geq 0$, $h_1 > 0, \ldots, h_t > 0$ has a solution in $K^l = \mathbb{A}^l$ is called the *Existential theory of the reals* (ETR). ETR is polynomial-time equivalent to the Steinitz problem for 4-polytopes over $\mathbb{A}$ (Richter-Gebert [99]). (All polytopes realizable over $\mathbb{R}$, are realizable over $\mathbb{A}$.) Moreover, ETR is polynomial-time equivalent to the Steinitz problem for d-Polytopes with $d+4$ vertices over $\mathbb{A}$ (Mnëv [87]). Since ETR is easily verified to be $\mathcal{NP}$-hard[6], the same is true for the general Steinitz problem over $\mathbb{A}$, and we obtain:

Corollary 2.20. The IIS-hypergraph recognition problem over $\mathbb{A}$ is $\mathcal{NP}$-hard.

Richter-Gebert's result shows that the Steinitz problem is $\mathcal{NP}$-hard for fixed dimension ($d = 4$), while Mnëv's result shows that the recognition problem for IIS-hypergraphs of fixed rank is $\mathcal{NP}$-hard (by polarity the Steinitz Problem is $\mathcal{NP}$-hard for $m = d + 4$ facets, i.e., the rank is $r = m - d = 4$).

According to Theorem 2.7 of Bokowski and Sturmfels [34], for $K = \mathbb{Q}$ or $K = \mathbb{A}$, deciding whether an arbitrary polynomial $f \in \mathbb{Z}[x_1, \ldots, x_l]$ has zeros in K^l, where l is a positive integer, is equivalent to solving the Steinitz problem for K. Remember that there exist polytopes, which cannot be realized with rational coefficients (see the beginning of this chapter). For $K = \mathbb{Q}$, it is not even clear whether the Steinitz problem (and therefore the IIS-hypergraph recognition problem) is decidable, since finding roots in $K = \mathbb{Q}$ of a single polynomial $f \in \mathbb{Z}[x_1, \ldots, x_l]$ is the unsolved rational version of Hilbert's 10th problem. Reduction of a system of polynomial (in)equalities to an equation involving only one polynomial can be achieved by standard methods, see, e.g., Section 2.3 of Bokowski and Sturmfels [34]. By the quantifier elimination result of Tarski, the problem is decidable for $K = \mathbb{A}$. For $K = \mathbb{A}$, it is unknown whether the Steinitz problem is in $\mathcal{NP}$. See Blum, Cucker, Shub, and Smale [33], Björner, Las Vergnas, Sturmfels, White, and Ziegler [30], Mishra [86], and references therein for this and related issues.

Let us discuss two more aspects of the above results. First note the asymmetry in the size of the input between the Steinitz problem and the IIS-hypergraph recognition problem. For the latter, a hypergraph storing the combinatorics of the IISs as input seems natural. For the Steinitz problem, however, we could take the vertex-facet incidences (instead of a lattice) as the input. This version of the Steinitz problem is clearly $\mathcal{NP}$-hard. A reduction

[6]ETR subsumes the following $\mathcal{NP}$-complete problem (see Schrijver [107, Chapter 18]): Given $\boldsymbol{a} \in \mathbb{Z}^\ell$ and $\beta \in \mathbb{Z}$, does $\boldsymbol{ax} = \beta$ have a 0/1-solution $\boldsymbol{x}$? One equation in the polynomial system of the corresponding instance for ETR is $\boldsymbol{ax} - \beta = 0$. Furthermore, for each $i = 1, \ldots, \ell$, we include $x_i^2 - x_i = 0$ as an equation. These equations force each x_i to be 0 or 1.

as in Theorem 2.19 for this case could not be obtained in this thesis. The reason being that the computational complexity of deciding boundedness of a polyhedron given vertex-facet incidences is unknown. Moreover, it is unknown whether computing the Euler characteristic of a simplicial complex (given the vertex-facet incidences) is $\mathcal{NP}$-hard. See Section 3.4 for further comments.

On the other hand, to provide a reduction leading to a complexity result as in Corollary 2.20, it suffices to consider polyhedra P with $\dim P = 4$ or with $\dim(P)+4$ vertices. In these cases the sizes of the face lattices are polynomially bounded in the size of a vertex-facet incidence matrix. By Corollary 3.12 we can decide boundedness of such polyhedra P in polynomial time in the size of its vertex-facet incidences. This provides an $\mathcal{NP}$-hardness proof of the IIS-hypergraph recognition problem by reduction from the Steinitz problem with vertex-facet incidences as input.

To establish the reverse direction of Theorem 2.19, one would need to provide an appropriate input (a lattice) to the Steinitz problem. This task appears to be difficult, because we need to consider the case of unbounded polyhedra. In fact, as discussed in Section 3.2 (see Figure 3.6 and Figure 3.7), it is in general impossible to reconstruct the face lattice of an unbounded polyhedron P from a vertex-facet incidence hypergraph H, even if H is a clutter hypergraph and $\dim(P) = 4$.

2.6 Finite Excluded Minor Characterization

Continuing the study of IIS-hypergraphs, it is a natural question whether there exists a finite excluded minor characterization of IIS-hypergraphs. We will give a possible notion of a minor below. The class of IIS-hypergraphs, however, is not closed under this minor definition. Nevertheless, we will define partial IIS-hypergraphs, which are closed under this definition. Then, we will see that there cannot exist a finite excluded minor characterization of this class, applying similar results for polytopes.

Let $H = (S, \mathcal{E})$ be a hypergraph and $s \in S$ be a node of H. We consider two operations on H. First, we obtain the hypergraph $H \setminus s$ by *deletion* of s, which has $S - \{s\}$ as nodes and the family $(E \in \mathcal{E} : s \notin E)$ as edges. Second, H/s is the hypergraph $(S - \{s\}, (E - \{s\} : E \in \mathcal{E}, s \in E))$, which is obtained by *contraction* of s.[7] These operations (as defined here) are natural in the

[7]Note that there exist other definitions of the contraction H/s, e.g., the hypergraph with $S - \{s\}$ as nodes and the minimal (w. r. t. inclusion) elements of $\{E - \{s\} : E \in \mathcal{E}\}$ as the edges. See Cornuéjols [51], where this definition arises naturally in the context of the set covering problem. This definition is unsuitable in our context.

sense that they "almost" preserve the property of being an IIS-hypergraph, as we will see in Proposition 2.23 below. To prove this, we need two technical lemmas and the following constructions. Let $E \in \mathcal{E}$ be an edge. Then $H|_E$, the hypergraph H *restricted* to E, is the hypergraph with node set E and $(E \cap E' \,:\, E' \in \mathcal{E},\ E' \neq E)$ as edges. Similarly, we let $H|^E$ be the hypergraph with nodes $S \setminus E$ and the following edges $((S \setminus E) \cap E' \,:\, E' \in \mathcal{E},\ E' \neq E)$. Both constructions can produce empty edges and are not necessarily clutter hypergraphs, even if H is a clutter hypergraph. This cannot happen for deletion and contraction, unless $\{s\}$ is an edge of H and s is contracted.

Lemma 2.21. Let $H = (S, \mathcal{E})$ be a hypergraph and $H^* = (\mathcal{E}, \mathcal{E}^*)$ be its dual hypergraph (see Definition 2.14). Then for $s \in S$ and the corresponding edge $\mathcal{E}^*_s = \{E^* \,:\, s \in E,\ E \in \mathcal{E}\}$ of the dual, we have the following:

(a) $(H/s)^* \cong H^*|_{\mathcal{E}^*_s}$

(b) $(H\setminus s)^* = H^*|^{\mathcal{E}^*_s}$

Proof. The proof is technical but straightforward. For notational convenience, we write $S - s$ for $S - \{s\}$ and use $\mathcal{E}_s := (E \in \mathcal{E} \,:\, s \in E)$ for $s \in S$.

$$
\begin{aligned}
\text{(a)}\quad (H/s)^* &= (S - s,\, (E - s \,:\, E \in \mathcal{E}_s)\,)^* \\
&= (\{(E-s)^* : E \in \mathcal{E}_s\},\, (\{(E-s)^* \,:\, E \in \mathcal{E}_s \cap \mathcal{E}_t\} : t \in S - s)) \\
&\cong (\{E^* \,:\, E \in \mathcal{E}_s\},\, (\{E^* \,:\, E \in \mathcal{E}_s \cap \mathcal{E}_t\} \,:\, t \in S - s)\,) \\
&= (\mathcal{E}^*_s,\, (\mathcal{E}^*_s \cap \mathcal{E}^*_t \,:\, t \in S - s)\,) = H^*|_{\mathcal{E}^*_s}.
\end{aligned}
$$

$$
\begin{aligned}
\text{(b)}\qquad (H\setminus s)^* &= (S - s,\, (E \,:\, E \notin \mathcal{E}_s)\,)^* \\
&= (\{E^* \,:\, E \notin \mathcal{E}_s\},\, (\{E^* \,:\, E \notin \mathcal{E}_s,\, E \in \mathcal{E}_t\},\, t \in S - s)\,) \\
&= (\mathcal{E}^* \setminus \mathcal{E}^*_s,\, (\mathcal{E}^*_t \setminus \mathcal{E}^*_s \,:\, t \in S - s)\,) \\
&= (\mathcal{E}^* \setminus \mathcal{E}^*_s,\, ((\mathcal{E}^* \setminus \mathcal{E}^*_s) \cap \mathcal{E}^*_t \,:\, t \in S - s)\,) = H^*|^{\mathcal{E}^*_s}.
\end{aligned}
$$

□

Note that in the contracted or deleted dual hypergraph (many) empty edges may appear, e.g., if $\mathcal{E}_s \cap \mathcal{E}_t = \varnothing$.

Lemma 2.22. Let $H = (S, \mathcal{E})$ be a hypergraph and E be an edge. Then we have $\overline{H|_E} = \overline{H}|^{\overline{E}}$.

Proof. Here, we have to be careful in what context the complement operator is applied.

$$\begin{aligned}
\overline{H|_E} &= (E,\, \overline{(E \cap E' \,:\, E' \in \mathcal{E},\; E' \neq E)}\,) \\
&= (E,\, (E \setminus (E \cap E') \,:\, E' \in \mathcal{E},\; E' \neq E)\,) \\
&= (E,\, (E \setminus E' \,:\, E' \in \mathcal{E},\; E' \neq E)\,) \\
&= (E,\, ((S \setminus E') \cap E \,:\, E' \in \mathcal{E},\; E' \neq E)\,) \\
&= (E,\, (\overline{E'} \cap E \,:\, E' \in \mathcal{E},\; E' \neq E)\,) \\
&= (S \setminus \overline{E},\, (\overline{E'} \cap (S \setminus \overline{E}) \,:\, \overline{E'} \in \overline{\mathcal{E}},\; \overline{E'} \neq \overline{E})\,) = \overline{H}|^{\overline{E}}.
\end{aligned}$$

The first application of the complement operator is with respect to the vertex set E and all others are with respect to S. □

Proposition 2.23. Let $H = (S,\, \mathcal{E})$ be an IIS-hypergraph and s a node of H.

(i) H/s is not necessarily an IIS-hypergraph, but can be completed to an IIS-hypergraph by adding edges.

(ii) If $H \backslash s$ is nonempty then it is an IIS-hypergraph.

Proof. Let Σ be an infeasible linear inequality system corresponding to H.

(i) If there exists no edge (IIS) containing s, then H/s has no edges and hence is not an IIS-hypergraph. Therefore, H/s is not necessarily an IIS-hypergraph.

By Lemma 2.21 and Lemma 2.22, we have $\overline{(H/s)^*} = \overline{H^*|_{\mathcal{E}_s^*}} = \overline{H^*}|^{\overline{\mathcal{E}_s^*}}$. Let $P = \{\boldsymbol{y} \in \mathbb{R}^m \,:\, \boldsymbol{y}A = \boldsymbol{0},\; \boldsymbol{y}\boldsymbol{b} = -1,\; \boldsymbol{y} \geq \boldsymbol{0}\}$ be the alternative polyhedron of Σ (see Definition 1.9), where m is the number of nodes of H. As in the proof of Lemma 2.16, it follows that $\overline{H^*}$ is a vertex-*face* incidence hypergraph of P, i.e., if we identify its nodes with the vertices of P, every edge is the vertex set of a face defined by an inequality $y_j \geq 0$ for some $1 \leq j \leq m$. The family $\mathcal{E}_s = (E \in \mathcal{E} \,:\, s \in E)$ includes all IISs that contain inequality s. In $\overline{H^*}$, $\overline{\mathcal{E}_s^*}$ is an edge which is the set of vertices of P that lie on the face F corresponding to s. Hence, $\mathcal{E}_s^*$ is the set of vertices not on F. Therefore, the hypergraph $\overline{H^*}|^{\overline{\mathcal{E}_s^*}}$ is the restriction to the set of vertices (IISs) that do not lie on F, i.e., the deletion of F (and all vertices on it). Removing the inequality corresponding to F from the description of P deletes the vertices on F and may produce new vertices (IISs). Therefore, $\overline{H^*}|^{\overline{\mathcal{E}_s^*}}$ can be completed to a vertex-face incidence hypergraph $\overline{K^*}$ by adding nodes (vertices). Again as in the proof of Lemma 2.16, it follows that K is an IIS-hypergraph, which extends H/s by added edges.

(ii) If we delete s from H, this corresponds to deleting inequality s and all edges (IISs) that contain s from Σ. If there remains at least one edge, then the result is again an IIS-hypergraph, otherwise the system is feasible.

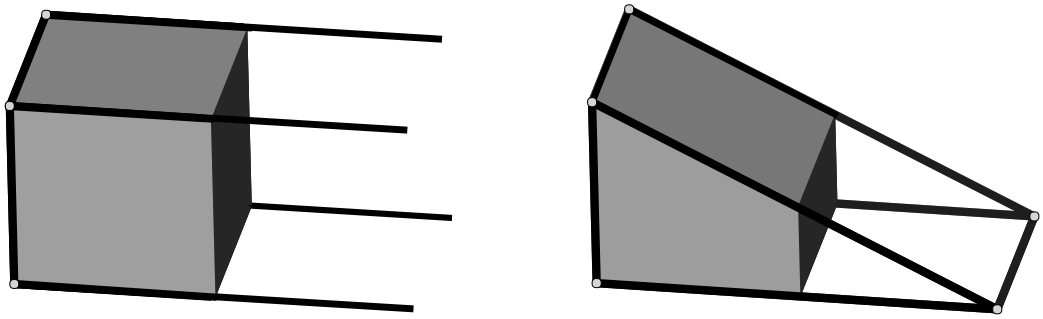


Figure 2.4: Illustration of the contraction operation (Proposition 2.23, Remark 2.24). A regular 3-cube (left) and a perturbed 3-cube (right) are shown as solid parts. Both are combinatorially equivalent, but the results after deleting the right facet, shown in dark grey, are combinatorially not isomorphic. The 1-skeleton (graph) of the result after deletion is indicated by the black edges and the light grey vertices. On the left we get an unbounded polyhedron and on the right a polytope with two new vertices; compare also Figure 2.6.

This operation also is the restriction $\overline{H^*}|_{\overline{\mathcal{E}_s^*}}$ to the face corresponding to s in the alternative polyhedron (see Definition 1.9). Hence, as in part (i), the result also follows from Lemma 2.21 (a) and the fact that $\overline{H|^E} = \overline{H}|_{\overline{E}}$, which can be shown similar to the proof of Lemma 2.22. □

Remark 2.24. The completion in Proposition 2.23 (ii) is not unique, but depends on the geometry of the polyhedron P. Take, for example, the regular 3-cube and a perturbed 3-cube as in Figure 2.4. Deleting the facet to the right results in two combinatorially different polytopes.

Example 2.25. To give a more "concrete" feeling for the situation, we provide explicit coordinates for the perturbed 3-cube in Figure 2.4 and a contraction operation on the corresponding IIS-hypergraph.

The perturbed 3-cube P can be written as

$$\{\boldsymbol{z} \in \mathbb{R}^3 \,:\, z_1 \geq 0,\ z_2 \geq 0,\ z_3 \geq 0,\ z_1 \leq 1,\ z_2 \leq 1,\ \tfrac{1}{2}z_1 + z_3 \leq 1\}\,. \qquad (2.1)$$

By introducing slack variables, then subtracting the last equation from the first two, and multiplying the last row by -1, we get the following description of P:

$$\{\boldsymbol{y} \in \mathbb{R}^6 \,:\, \begin{pmatrix} \frac{1}{2} & 0 & -1 & 1 & 0 & -1 \\ -\frac{1}{2} & 1 & -1 & 0 & 1 & -1 \\ -\frac{1}{2} & 0 & -1 & 0 & 0 & -1 \end{pmatrix} \boldsymbol{y} = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}, \quad \boldsymbol{y} \geq \boldsymbol{0}\}\,.$$

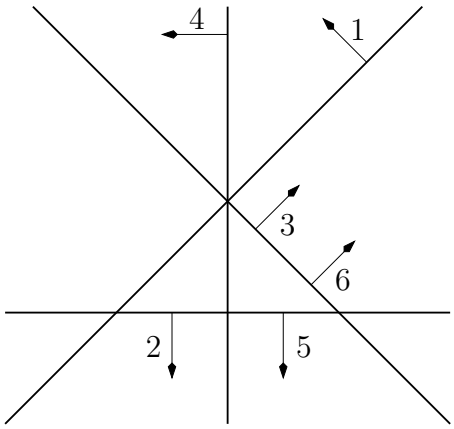


Figure 2.5: Infeasible system corresponding to the perturbed 3-cube on the right of Figure 2.4, see Example 2.25. Arrows point into the corresponding halfspaces.

We have transformed P to $P' := \{\boldsymbol{y} \in \mathbb{R}^m : \boldsymbol{y}A = \boldsymbol{0},\ \boldsymbol{y}\boldsymbol{b} = -1,\ \boldsymbol{y} \geq \boldsymbol{0}\}$, which is in the alternative polyhedron form of Definition 1.9, where

$$A := \begin{pmatrix} \frac{1}{2} & 0 & -1 & 1 & 0 & -1 \\ -\frac{1}{2} & 1 & -1 & 0 & 1 & -1 \end{pmatrix}^T \quad \text{and} \quad \boldsymbol{b} := (-\tfrac{1}{2},\ 0,\ -1,\ 0,\ 0,\ -1)^T.$$

Hence, by Theorem 1.12, the corresponding infeasible system $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ is the following (see Figure 2.5):

$$\begin{array}{lrcl} (1) & \frac{1}{2}x_1 - \frac{1}{2}x_2 & \leq & -\frac{1}{2} \\ (2) & x_2 & \leq & 0 \\ (3) & -x_1 - x_2 & \leq & -1 \\ (4) & x_1 & \leq & 0 \\ (5) & x_2 & \leq & 0 \\ (6) & -x_1 - x_2 & \leq & -1. \end{array}$$

The IISs are: $\{1,2,3\}$, $\{1,2,6\}$, $\{1,3,5\}$, $\{1,5,6\}$, $\{2,4,6\}$, $\{2,3,4\}$, $\{4,5,6\}$, and $\{3,4,5\}$. The corresponding IIS-hypergraph H has these sets as edges and $\{1,2,\ldots,6\}$ as nodes. As claimed by Theorem 1.12, the IISs are the supports of the vertices of P'. If the facets of P are numbered in the order in which the corresponding defining inequalities appear in (2.1), we get a one-to-one correspondence to the facets of P'.

Let us now contract element 4 in H. This corresponds to deleting the face defined by $z_1 \leq 1$ of P. Hence, we have to delete the first row and fourth column of A^T and the fourth component of $\boldsymbol{b}$, to get:

$$\{\boldsymbol{y} \in \mathbb{R}^5 : (-\tfrac{1}{2},\ 1,\ -1,\ 1,\ -1)\,\boldsymbol{y} = 0,\ (-\tfrac{1}{2},\ 0,\ -1,\ 0,\ -1)\,\boldsymbol{y} = -1,\ \boldsymbol{y} \geq \boldsymbol{0}\}.$$

If we keep the original numbering of the inequalities, the corresponding infeasible system is:

$$\begin{array}{ll} (1) & -\frac{1}{2}x \leq -\frac{1}{2} \\ (2) & x \leq 0 \\ (3) & -x \leq -1 \\ (5) & x \leq 0 \\ (6) & -x \leq -1. \end{array}$$

The IISs are: $\{1,2\}$, $\{1,5\}$, $\{2,3\}$, $\{3,5\}$, $\{2,6\}$, $\{5,6\}$, as is easily seen. The contracted hypergraph $H/4$ has $\{1,2,3,5,6\}$ as nodes and $\{2,6\}$, $\{2,3\}$, $\{5,6\}$, $\{3,5\}$ as edges. Hence, if we add the edges $\{1,2\}$ and $\{1,5\}$, which correspond to the two new vertices on the right of Figure 2.4, we again obtain an IIS-hypergraph. In fact, $H/4$ is itself an IIS-hypergraph: The corresponding alternative polyhedron is the unbounded one on the left of Figure 2.4.

Remark 2.26. An exponential number of edges might be necessary to complete H/s to an IIS-hypergraph. For instance, take the polar of a so-called dwarfed polytope, constructed by Avis, Bremner, and Seidel [13], as the alternative polyhedron.

Proposition 2.23 shows that if we contract a node of an IIS-hypergraph, the result is not necessarily an IIS-hypergraph, but can be completed to one. We therefore define a *partial IIS-hypergraph* to be a hypergraph that may be completed to an IIS-hypergraph by adding edges (IISs). A hypergraph that is obtained by sequential contractions and deletions of nodes of a hypergraph H is called a *minor* of H. It follows from Proposition 2.23 that every minor of a partial IIS-hypergraph is again a partial IIS-hypergraph (except if it is empty). Since these operations just correspond to taking submatrices of the incidence matrices, the resulting hypergraph does not depend on the order of the operations.

Our question from the beginning of this section can now be phrased as follows: Do there exist finitely many hypergraphs $M_1, M_2, \ldots, M_k$, such that a hypergraph H is a partial IIS-hypergraph if and only if H does not have a minor isomorphic to some M_i? The answer is negative (see Theorem 2.29). This result is obtained by translating a result for vertex-facet incidences of polytopes obtained by Richter-Gebert. We need the following proposition to state it.

Proposition 2.27. Let $H = (V, \mathcal{F})$ be a vertex-facet incidence hypergraph of a *polytope* P and let $v \in V$ be a vertex. Then

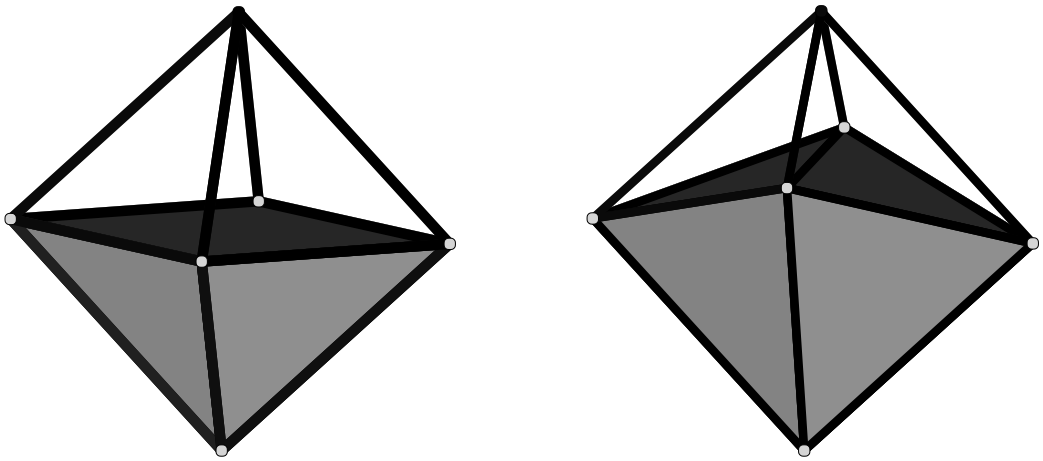


Figure 2.6: Illustration of the deletion operation in Proposition 2.27. A regular octahedron (left) and a perturbed octahedron (right). They are combinatorially equivalent, but the results after deleting the top vertex are combinatorially not isomorphic. The results are the solid parts, where the new facets are shown in dark grey. See also Figure 2.4.

(i) H/v is a vertex-facet incidence hypergraph of a polytope.

(ii) $H \setminus v$ is not necessarily a vertex-facet incidence hypergraph of a polytope, but can be completed to one by adding edges (facets).

Proof. Statement (i) is true since the vertex-facet incidence hypergraph of the vertex figure at v is isomorphic to the hypergraph H/v.

Deleting a vertex "cuts a hole" into the boundary of P (all incident facets are deleted). Taking the convex hull over the rest of the vertices produces a polytope P', which coincides with P on the part not incident to v and has new facets "filling the hole". This proves (ii). □

Note that Proposition 2.27 is not quite the dual version of Proposition 2.23. Proposition 2.27 only works for polytopes, where we always get new edges (facets), while in Proposition 2.23 we allowed unbounded polyhedra, for which it can happen that no new vertices appear, see Figures 2.4 and 2.6.

As for IIS-hypergraphs, we define a *partial vertex-facet incidence hypergraph* (of a polytope) as a hypergraph that can be completed to a vertex-facet incidence hypergraph (of a polytope) by adding edges. Theorem 9.4.3 of Richter-Gebert [99] implies that there does not exist a finite excluded minor characterization of partial vertex-facet incidence hypergraphs of polytopes; we translate it to hypergraph language.

Theorem 2.28 (Richter-Gebert [99]). For each even $n \in \mathbb{N}$ there exists a hypergraph $H_n = (V_n, \mathcal{F}_n)$, with more than n nodes, such that:

(a) Each H_n defines a so-called *non-polytopal combinatorial* 3*-sphere*. That is, H_n is not a partial vertex-facet incidence hypergraph of a polytope, and the order complex of the "intersection" lattice $\mathcal{L}(H_n)$ (see Section 2.5 and 3.3) is a 3-dimensional PL-sphere.

(b) For every $v \in V_n$, the contracted hypergraph H_n/v is a vertex-facet incidence hypergraph of a 3-polytope.

(c) For every $v \in V_n$, the hypergraph $H_n \backslash v$, obtained by deleting v, can be completed (by adding edges) to a vertex-facet incidence hypergraph of a 4-polytope, i.e., is a partial vertex-facet incidence hypergraph.

Part (b) follows since the contraction of a vertex v corresponds to computing the vertex figure at v. This produces a combinatorial 2-sphere and hence is polytopal, i.e., can be realized as a 3-polytope by Steinitz's theorem.

We arrive at the wanted result:

Theorem 2.29. There exists no finite excluded minor characterization of partial IIS-hypergraphs.

Proof. Suppose there exist hypergraphs $M_1, \ldots, M_k$, such that any hypergraph H is a partial IIS-hypergraph if and only if there exists no M_i that is isomorphic to a minor of H. Hence, $M_1, \ldots, M_k$ are not partial IIS-hypergraphs.

Consider the hypergraph $\overline{H_n^*}$ corresponding to H_n of Theorem 2.28. The hypergraph $\overline{H_n^*}$ is not an IIS-hypergraph, for assume this would be the case. Then by Lemma 2.16, H_n would be a vertex-facet incidence hypergraph of a polyhedron, since H_n is a clutter hypergraph. By Theorem 2.28 (a), H_n is not a vertex-facet incidence hypergraph of a polytope. Furthermore, by Theorem 3.11, it cannot belong to an unbounded polyhedron, since the order complex of $\mathcal{L}(H_n)$ is a sphere (and hence $\widetilde{\chi}(\mathcal{L}(H_n)) \neq 0$). It also follows from this argument that H_n cannot be a partial vertex-facet incidence hypergraph of a polyhedron and hence $\overline{H_n^*}$ cannot be a partial IIS-hypergraph by Lemma 2.17.

On the other side, by Theorem 2.28 (b) and (c), every (nonempty) minor of $\overline{H_n^*}$ is a partial IIS-hypergraph.

By assumption, there exists i such that M_i is a minor of $\overline{H_n^*}$. By taking n large enough, we can assume that $\overline{H_n^*}$ is not isomorphic to one of $M_1, \ldots, M_k$. But every (nonempty) minor of $\overline{H_n^*}$ is a partial IIS-hypergraph and hence M_i is a partial IIS-hypergraph, which is a contradiction. □

Remark 2.30. Consider the problem to decide whether a partial vertex-facet incidence hypergraph H of a polytope is complete, i.e., a vertex-facet incidence hypergraph of a polytope. Joswig and Ziegler [75] give an example

which shows that the dimension is essential for the answer to this problem, i.e., the example is complete if the dimension is 3, while it is not if the dimension is 4. Let d be the dimension in which we want to test completeness. First compute $\mathcal{L}(H)$ (see Section 2.5). If $\mathcal{L}(H)$ is not ranked, atomic, or coatomic, H is not complete. Now check whether $d = r(\mathcal{L}(H)) - 1$; if not, H is not complete. Next compute the reduced $(d-1)$-homology group $\widetilde{H}_{d-1}(\Gamma(H); K)$, where K is any field. $\Gamma(H)$ is the so-called cross-cut complex of H, the simplicial complex which has the edges of H as facets and the nodes of H as vertices. If $\widetilde{H}_{d-1}(\Gamma(H); K) \neq 0$ then H is complete (see Joswig and Ziegler [75]). This condition tests if there is a "hole" in the boundary of the complex. It is an open question whether there is an *efficient* way to test completeness, i.e., whether there exists a polynomial-time algorithm in the size of H or in the size of $\mathcal{L}(H)$. On the other hand, $\widetilde{H}_{d-1}(\Gamma(H); K)$ can be computed in polynomial time w. r. t. the size of $\Gamma(H)$, which is in general much larger than the size of $\mathcal{L}(H)$ – see also Problem 33 of [77].

Let H be a partial IIS-hypergraph. It is not clear how to test if H is an IIS-hypergraph, i.e., H is complete. The above mentioned way cannot directly be translated to IIS-hypergraphs, since IIS-hypergraphs can correspond to unbounded polyhedra and therefore testing for "holes" in the boundary does not help.

Remark 2.31. It is unknown whether one can test in polynomial time if a hypergraph of fixed size is a minor of a given hypergraph. Since it is also unknown if completeness of a partial IIS-hypergraph can be tested in polynomial time, we cannot conclude that Theorem 2.29 follows from the $\mathcal{NP}$-hardness of recognizing IIS-hypergraphs (Corollary 2.20).

Note that our minor definition is not directly related to the usual definition of a minor of a graph. It is therefore unclear whether the following result would help to solve the above minor detection problem for hypergraphs: Robertson and Seymour [100] describe a polynomial-time algorithm that tests whether a given fixed graph is a minor of some other graph.

Chapter 3

Vertex-Facet Incidences of Unbounded Polyhedra

In the previous two chapters we often came across the relation between an infeasible linear inequality system Σ and its corresponding alternative polyhedron (Theorem 1.12). In Chapter 2, we were interested in the "combinatorial interplay" of the irreducible inconsistent subsystems (IISs) of Σ. More precisely, we studied IIS-hypergraphs, where a hypergraph is an IIS-hypergraph if its nodes correspond to the inequalities of an infeasible linear inequality system and its edges correspond to the IISs of this system. Hence, IIS-hypergraphs capture the "intersection properties" of IISs. In Chapter 2, the above relation directly led to the development of tools to translate between IIS-hypergraphs and vertex-facet incidence hypergraphs of polyhedra. In this chapter we will take a closer look at the incidences of possibly unbounded polyhedra.

For a polytope P, every (proper) face of P is the convex hull of the vertices it contains, and it is also the intersection of the facets that contain it. Thus, the combinatorial structure of P, i.e., its face lattice, is entirely determined by its vertex-facet incidences. What are the differences when considering not necessarily bounded polyhedra?

The combinatorics of unbounded polyhedra has received only little attention up to now. Some exceptions are the articles of Klee [78], Billera and Lee [27], Barnette, Kleinschmidt, and Lee [22], and Lee [81], which extend the lower bound and upper bound theorem for polytopes to general (simple) polyhedra. The main tool used in these papers are *polytope pairs*, i.e., a polytope with a distinguished face. Applying a projective transformation that sends this face to infinity, we have the situation of a geometrically given unbounded polyhedra with a distinguished face "at infinity".

But what can really be said/detected/reconstructed if only the the vertex-facet incidences are given and no data about the situation "at infinity"?

After introducing notation and some basic facts about unbounded polyhedra in Section 3.1, we discuss these questions in Section 3.2. Several conditions for the possibility to reconstruct the face lattice of an unbounded polyhedron are given and their limits are shown by examples. It turns out

that for many cases such a reconstruction is not possible. The second main result of this chapter (in Section 3.3) will be that one can, however, detect from a vertex-facet incidence matrix whether the polyhedron under consideration is bounded or not. In Section 3.4 we discuss algorithmic issues of this detection method. Thirdly (in Section 3.5), we discuss the "unbounded version" of a very basic lemma about polytopes. Indeed, Exercise 0.1 of [115] asks one to prove that any d-polytope that is both simplicial (every facet has d vertices) and simple (every vertex is on d facets) must either be a simplex or a polygon ($d = 2$). But how about unbounded polyhedra? We prove that a polyhedron of dimension at least 2 that is both simple and simplicial (with the definitions as given here) cannot be unbounded. As a byproduct, we obtain a characterization of those polyhedra that have circulant vertex-facet incidence matrices. It is this characterization which triggered the investigations in this chapter, as it was needed when considering generalized antiwebs, whose incidences are just circulant matrices (see Section 1.3.3).

This chapter is joint work with Michael Joswig, Volker Kaibel, and Günter M. Ziegler. Most parts appear in [74].

3.1 Basic Facts

In this section we fix some notation for this chapter. Generally, P denotes a d-polyhedron with m facets and n vertices. We will always assume that P is pointed (i.e., it has at least one vertex) and that $d \geq 1$. In particular, these conditions imply $n \geq 1$ and $m \geq d \geq 1$.

In this chapter, it will be more convenient to encode vertex-facet incidences in a matrix rather than in a hypergraph (compare Definition 2.15). A 0/1-matrix $A = (a_{fv}) \in \{0,1\}^{m \times n}$ is a *vertex-facet incidence matrix* of P, if the vertices and facets of P can be labeled by $\{1, \ldots, n\}$ and $\{1, \ldots, m\}$, respectively, such that $a_{fv} = 1$ if and only if the vertex with label v is contained in the facet with label f. We will usually identify vertices and facets with their label. We have that A is a vertex-facet incidence matrix if and only if it is the edge-node incidence matrix of a vertex-facet incidence hypergraph (see Definition 2.15).

Any (unbounded) polyhedron P can be projectively transformed to a *polytope*. We denote by $\overline{P}$ any such projectively equivalent polytope. If P is unbounded, then we denote the image of the face at infinity, the so-called *far face*, by F_∞. The face F_∞ is the unique maximal element among the faces of $\overline{P}$ that are not images of faces of P under the projective transformation mapping P to $\overline{P}$. If P is bounded, then we define $F_\infty = \varnothing$. Figure 3.1 illustrates a three-dimensional example.

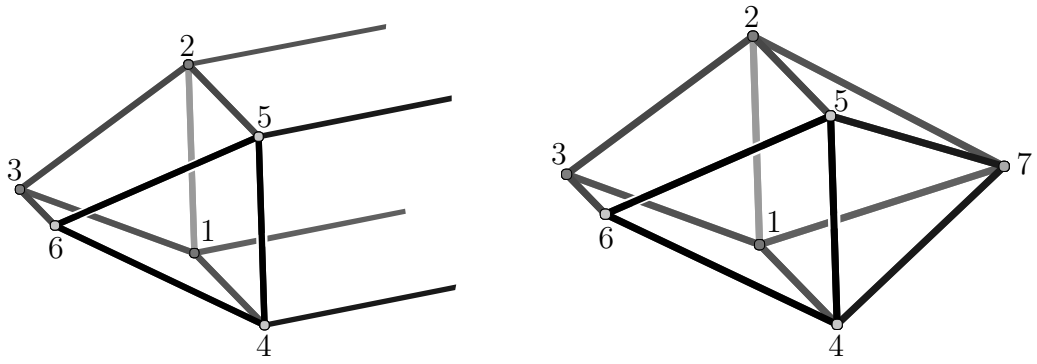


Figure 3.1: Left: Unbounded 3-polyhedron P; all unbounded edges are parallel. **Right:** Projectively transformed into $\overline{P}$. Vertex 7 is the far face F_∞.

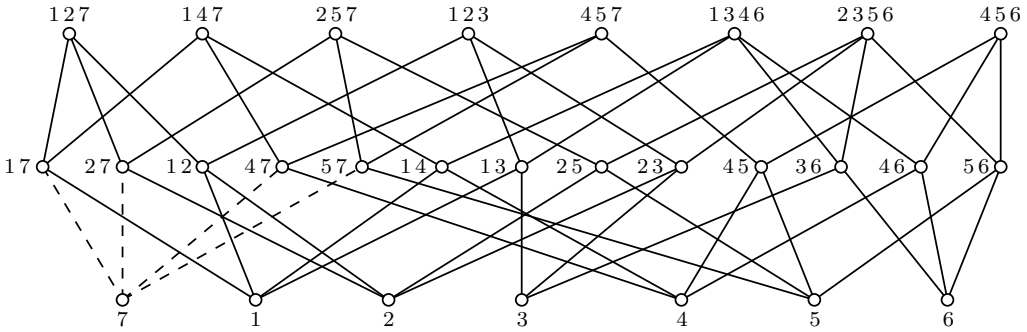


Figure 3.2: Face poset $\mathcal{F}(\overline{P})$ for the example given in Figure 3.1, where the solid part is $\mathcal{F}(P)$. The dashed lines indicate incidences to the far face, which is the vertex labeled 7.

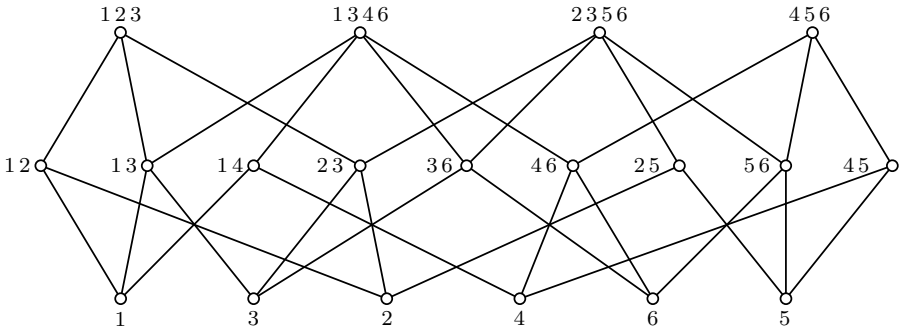


Figure 3.3: The poset $\mathcal{V}(P)$ for the example given in Figure 3.1.

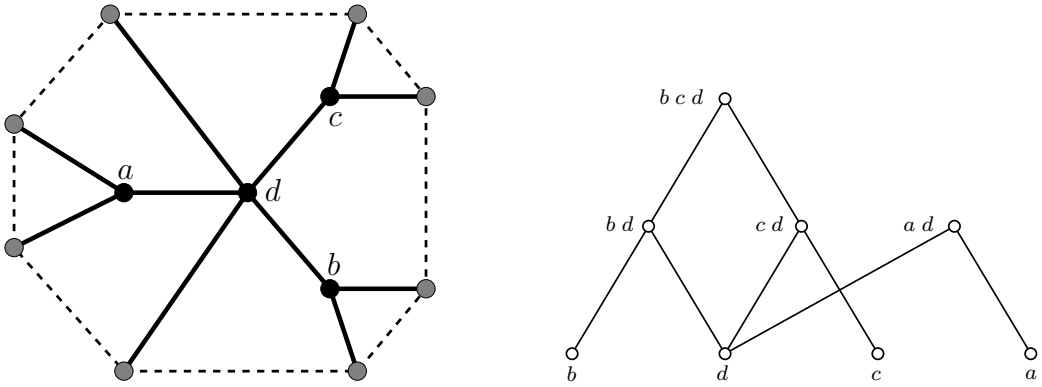


Figure 3.4: Example of an unbounded 3-polyhedron P (left), whose vertex poset $\mathcal{V}(P)$ (right) is not graded. The dashed line and grey vertices indicate the far face.

Let $\mathcal{F}(P)$, the *face poset* of P, be the set of non-trivial faces of P (excluding $\varnothing$ and P itself), ordered by inclusion. The face poset $\mathcal{F}(P)$ arises from the face poset $\mathcal{F}(\overline{P})$ by removing the far face F_∞ (and all its faces). While $\mathcal{F}(\overline{P})$ is independent of the actual choice of $\overline{P}$, in general it depends on the geometry of P, not only on its combinatorial structure; e.g., in the example of Figure 3.1, we get a higher dimensional far face if the unbounded edges are not all parallel. Figure 3.2 shows the two posets $\mathcal{F}(P)$ and $\mathcal{F}(\overline{P})$ for the example in Figure 3.1.

The poset $\mathcal{V}(P) = \{\text{vert}(F) : F \text{ non-trivial face of } P\}$ (where $\text{vert}(F)$ is the set of vertices of F) will play an important role. It can be computed from any vertex-facet incidence matrix $A \in \{0,1\}^{m \times n}$ of P, since it is the set of all nonempty intersections of the subsets of $\{1, \ldots, n\}$ defined by the rows of A. If we adjoin an artificial top element $\hat{1}$ and bottom element $\hat{0}$ to $\mathcal{V}(P)$, we get the lattice $\mathcal{L}(H)$ from Section 2.5, where H is the hypergraph defined by A, i.e., A is the edge-node incidence matrix of H. Figure 3.3 shows $\mathcal{V}(P)$ for the example given in Figure 3.1. In general $\mathcal{V}(P)$ is not graded, as can be seen by the example in Figure 3.4, which was constructed by Volker Kaibel.

In Figure 3.4 and the following examples, a three-dimensional polyhedron is visualized by its graph, where the far face is indicated by a dashed circuit, surrounding the other parts. Alternatively, one can "see" the polyhedron as "capped", i.e., cut by an appropriate hyperplane (indicated by the dashed lines) and then projected to two dimensions. Hence, the pictures show projections of 3-polytopes. By Steinitz's theorem an abstract graph is the graph of a 3-polytope if and only if it is simple, three-connected, and planar (see Richter-Gebert [99] and Ziegler [115]). Therefore, it is easy to check that the examples provided in the following indeed describe 3-polyhedra.

Let the *graph* Γ_P of P be the graph on the vertices of P defined by the bounded one-dimensional faces of P, i.e., Γ_P is the subgraph of the graph of $\overline{P}$ that is induced by those vertices of $\overline{P}$ that are not contained in F_∞. In the following, we call the bounded one-dimensional faces of P *edges*, while unbounded one-dimensional faces are called *extremal rays*. Two vertices of P are connected by an edge of P if and only if there is a face of P which contains exactly these two vertices. Since we can compute $\mathcal{V}(P)$ from the vertex-facet incidences of P, we can compute Γ_P from these incidences, as well. We will use the following fact.

Lemma 3.1. For every polyhedron P, the graph Γ_P is connected. Moreover, all faces of P induce connected subgraphs of Γ_P.

Proof. First assume that P is unbounded. Since P is pointed, it can be written as $\{\boldsymbol{x} \in \mathbb{R}^n : A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{x} \geq \boldsymbol{0}\}$. In this form, it can be cut by a halfspace $H := \{\boldsymbol{x} : \boldsymbol{h}\boldsymbol{x} \leq h_0\}$ to obtain a polytope and such that $\boldsymbol{h}\boldsymbol{v} < h_0$ for all vertices $\boldsymbol{v} \in \operatorname{vert} P$. We can assume w. l. o. g. that h is in general position. Let $F := \{\boldsymbol{x} : \boldsymbol{h}\boldsymbol{x} = h_0\} \cap P$ be the new facet. The vector $\boldsymbol{h}$ defines a linear functional such that each vertex of P has a lower value than every vertex of F. Let $\boldsymbol{u}$ be the (unique) minimum of this functional. By the correctness of the simplex algorithm, there exists a path from any vertex $\boldsymbol{v}$ of P to $\boldsymbol{u}$ that does not meet any of the vertices on F. Hence, Γ_P is connected. By a similar argument, it follows that each face of P induces a connected subgraph.

If P is bounded, choose an arbitrary $\boldsymbol{h}$ in general position and let $\boldsymbol{u}$ be the minimum of the corresponding functional. Again by the correctness of the simplex algorithm there exists a path from any vertex $\boldsymbol{v}$ to $\boldsymbol{u}$ and hence Γ_P is connected. □

In fact, Balinski's theorem shows that the graph of a d-polytope is d-connected (see Theorem 3.14 in Ziegler [115]). For an unbounded polyhedron P, however, the example of a Cartesian product of a polyhedral cone with a segment shows that it is possible for Γ_P to be only 1-connected.

Let P be a pointed d-polyhedron ($d \geq 1$). Then, P is called *simple* if every vertex of P is contained in precisely d facets (or, equivalently, if precisely d edges and extremal rays are incident to each vertex). Furthermore, P is called *simplicial* if every facet of P has exactly d vertices. These notions generalize the well-known notions *simple* and *simplicial* for polytopes to polyhedra. While this generalization is standard for simple polyhedra, it is not common for simplicial polyhedra. Thus, it seems to be worth mentioning that simplicial unbounded polyhedra form a non-trivial class of polyhedra, as is seen by the following examples.

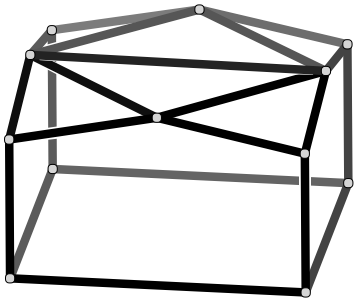


Figure 3.5: If the bottom face of this example is moved to infinity, we obtain a simplicial unbounded 3-polyhedron.

Examples 3.2. By a modification of the construction of a prism, every simplicial d-polytope (with $d \geq 2$) can be made the far face of a simplicial unbounded $(d+1)$-polyhedron (compare Figure 3.5):

For $d \geq 2$, take an arbitrary simplicial d-polytope Q and consider the prism P over Q. Denote by Q' the counterpart of Q in P. Let $F_1, F_2, \ldots, F_m$ be the facets of Q and let F_i' be the opposite face of F_i in Q' for $i = 1, 2, \ldots, m$. The faces F_i and F_i' lie in a unique hyperplane H_i, which defines a facet of P. Let $\hat{\boldsymbol{v}}_i$ and $\hat{\boldsymbol{v}}_i'$ be the vertex barycenters of F_i and F_i', respectively. Let $\alpha_i > 0$, for $i = 1, \ldots, m$, be values to be determined later. Obtain the point $\boldsymbol{v}_i$ by pulling $\hat{\boldsymbol{v}}_i$ a distance α_i away from $\hat{\boldsymbol{v}}_i'$ on the line through $\hat{\boldsymbol{v}}_i$ and $\hat{\boldsymbol{v}}_i'$. Then $\boldsymbol{v}_i$ still lies in H_i. Let $\hat{P}_i$ be the convex hull of P and the points $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_i$. The vertices of $\hat{P}_i$ are $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_i$ and the vertices of P. Let V_i be the union of $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_i\}$ with the set of vertices of Q. We can choose each α_i such that $\boldsymbol{v}_i$ lies in general position with respect to V_{i-1}, i.e., $\boldsymbol{v}_i$ does not lie in a common hyperplane with $d+1$ pairwise different points of V_{i-1} (this only excludes a finite number of positions for $\boldsymbol{v}_i$). Furthermore, each hyperplane H_i defines a facet $\hat{F}_i$ of $\hat{P}_i$, which has $2d+1$ vertices: d vertices of F_i and F_i', respectively, and the vertex $\boldsymbol{v}_i$. Therefore, all facets of $\hat{P}_i$ except $\hat{F}_1, \hat{F}_2, \ldots, \hat{F}_i$ and maybe Q' are simplices. Apply a projective transformation to $\hat{P}_m$ moving Q' to infinity to yield the unbounded polyhedron $\tilde{P}$. This transforms the facets $\hat{F}_1, \hat{F}_2, \ldots, \hat{F}_m$ to unbounded facets of $\tilde{P}$ and removes d vertices from each. The combinatorics of all other facets of $\hat{P}_m$ (except Q') is not changed. Therefore, $\tilde{P}$ is simplicial.

3.2 Reconstructing Polyhedra from Vertex-Facet Incidences

In this section, we investigate several conditions under which it is possible to reconstruct the face poset $\mathcal{F}(P)$ from the vertex-facet incidences of an (unbounded) d-dimensional polyhedron P. It turns out that these conditions

are quite diverse and do not yield a complete characterization.

The first question is whether one can compute the dimension of P from its incidences. Given any vertex-facet incidence matrix of a pointed d-polyhedron P, it is easy to decide whether $d \in \{1, 2\}$. Furthermore, if $d \in \{1, 2\}$, one can immediately read off $\mathcal{F}(P)$ from the vertex-facet incidences. Thus, for the rest of this section we restrict our attention to d-polyhedra with $d \geq 3$.

The example of polyhedral cones shows that reconstructing $\mathcal{F}(P)$ from the vertex-facet incidences of a d-polyhedron P with $d \geq 4$ is impossible in general, even if additionally the dimension d is specified. Furthermore, the same example demonstrates that it is, in general, impossible to detect the dimension of a d-polyhedron from its vertex-facet incidences for $d \geq 3$. For $d = 3$, however, these dimensional ambiguities occur for cones only.

Proposition 3.3. Given a vertex-facet incidence matrix of a d-polyhedron P with $d \geq 3$, it is possible to decide whether $d = 3$ or $d \geq 4$, unless P is a cone with more than three facets.

Proof. If P is a cone with three facets (i.e., we have $n = 1$ and $m = 3$), then clearly $d = 3$ holds. If P is not a cone, then it must have at least two vertices. Thus, by Lemma 3.1, P has at least one edge (which can be determined from the vertex-facet incidences of P). This edge is contained in precisely two facets of P if $d = 3$; otherwise, it is contained in more than two facets. □

In dimensions larger than three, cones are not the only polyhedra for which one cannot determine the dimension from the vertex-facet incidences.

Example 3.4. For instance, let Q be some q-polytope and let C be a cone with $m \geq 3$ facets. Then $P = Q \times C$ will be a polyhedron of dimension $q + \dim C$, whose vertex-facet incidences only depend on Q and m. For every d with $q + 3 \leq d \leq q + m$, there exists a d-polyhedron that has the same vertex-facet incidences as P. In particular, dimensional ambiguities already occur for 4-polyhedra not being cones.

These Cartesian products, however, are also "cone-like" in the sense that they do not have any bounded facet.

Proposition 3.5. Given a vertex-facet incidence matrix of a d-polyhedron P that has a bounded facet, one can determine d. Furthermore, one can decide from the vertex-facet incidences of P whether it has a bounded facet or not.

Proof. If P has a bounded facet, then the maximum length of a chain in $\mathcal{V}(P)$ is $d-1$, thus one can compute d from $\mathcal{V}(P)$ in this case. Corollary 3.13 proves the second statement of the proposition. □

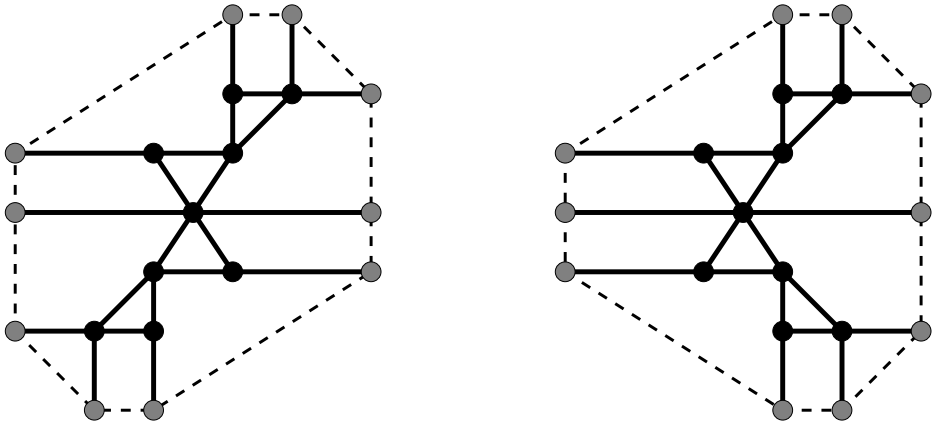


Figure 3.6: Example of two combinatorially different 3-polyhedra with isomorphic vertex-facet incidence matrices. The figures indicate the graphs (1-skeleta).

Having these results in mind, we now turn to the question under what conditions it is possible to reconstruct the face lattice of a polyhedron given its vertex-facet incidences. As noted above, one can see from the example of polyhedral cones that in general the combinatorial structure of an unbounded polyhedron is not determined by its incidences. A d-dimensional cone may have the combinatorial structure of any $(d-1)$-dimensional polytope (via homogenization); but from its vertex-facet incidences one can only read off its number of facets. The point is that, for unbounded polyhedra, the combinatorial information is based not only on the vertex-facet incidences, but also on the incidences of extremal rays and facets. For cones, nearly the entire information is contained in the latter incidences. The lattice-theoretic reason for such ambiguities is that the face lattice of an unbounded polyhedron is only coatomic, but not atomic.

Nevertheless, Propositions 3.3 and 3.5 might suggest to ask if the entire combinatorial structure of a d-polyhedron can be reconstructed from its vertex-facet incidences if $d = 3$ or if P has a bounded facet. However, the example given in Figure 3.6 shows that both conditions do not suffice. The crucial feature of the example is that one can reflect the "lower" parts in the drawings without affecting the vertex-facet incidences, while this changes the face poset: In contrast to the left polyhedron, the right one has two adjacent unbounded facets that contain three vertices each. For three-dimensional polyhedra this is more or less the only kind of ambiguity that can arise.

Proposition 3.6. Let P be a 3-polyhedron for which Γ_P is 2-connected. Then the face poset $\mathcal{F}(P)$ can be computed from the vertex-facet incidences of P.

Proof. One can compute Γ_P from the vertex-facet incidences of P and therefore the graph of each facet of P. If all the graphs of facets are cycles, then P is bounded and the statement is correct. Otherwise, consider the paths that are the graphs of the unbounded facets of P. Due to the 2-connectedness of Γ_P, there is (up to reorientation) a unique way to arrange these paths as a cycle going around the "finite" part of P. From this cycle, it is easy to determine the incidences of extremal rays and facets of P, which then allow to reconstruct the entire combinatorial structure of P. □

In larger dimensions, however, it is not true that higher connectedness of Γ_P for a polyhedron P is a sufficient condition for the possibility to reconstruct its combinatorial structure from its vertex-facet incidences. Figure 3.7 shows Schlegel diagrams of two unbounded 4-polyhedra which have been capped to obtain a 4-polytope (see also [73]). The new facet obtained by this construction is the outer Egyptian pyramid, i.e., the facet the Schlegel diagram is based on. The two original polyhedra have the same vertex-facet incidences and a 3-connected graph Γ_P, but their face posets are different (e.g., the right polyhedron has one extremal ray more than the left one).

In order to construct the examples, we start with the Cartesian product Q of a three-dimensional pyramid over a square and a ray. Thus, Q is a four-dimensional polyhedron with one bounded facet (a three-pyramid over a square), one facet which is an infinite prism over a square, and four facets that are infinite prisms over triangles. Let $\boldsymbol{v}$ be the vertex which comes from the top of the pyramid. From $\boldsymbol{v}$ one single extremal ray emanates. To create the example on the right of Figure 3.7, we slightly tilt one of the facets that meet in $\boldsymbol{v}$; this does not change the vertex-facet incidences of the polyhedron, but creates a second extremal ray that emanates from $\boldsymbol{v}$. To make the vertex sets of facets a clutter, i.e., no such set contains another, and to ensure that the convex hull of the vertices is full-dimensional, we additionally cut off two opposite vertices of the base of the pyramid.

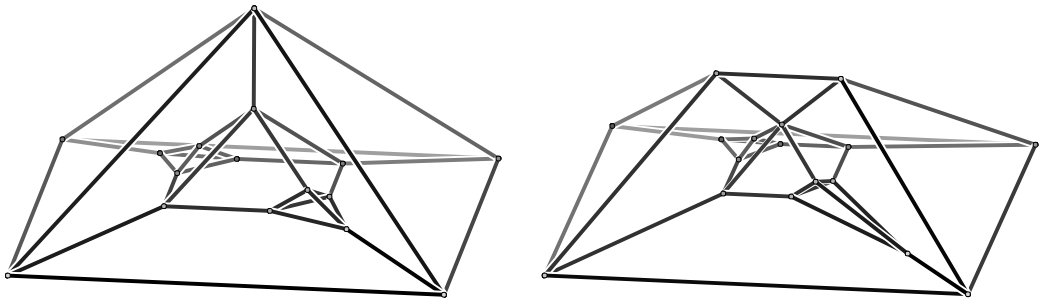


Figure 3.7: Schlegel diagrams illustrating two 4-polyhedra P_1 and P_2 which have the same vertex-facet incidences, but different face posets.

The examples illustrated in Figures 3.6 and 3.7 show that "cone-like" polyhedra are not the only polyhedra whose face poset cannot be reconstructed from their vertex-facet incidences (not even in dimensions three and four). Hence, "cone-like" polyhedra are not (extreme) examples of rather exotic unbounded polyhedra for which one obviously does not have any chance to reconstruct the combinatorial structure from their vertex-facet incidences, while this might be possible for all "reasonable" polyhedra. To make this a bit more precise we note that a cone is a quite degenerate polyhedron with respect to several criteria: (i) its set of vertices does not have the same dimension as the whole polyhedron, (ii) it does not have any bounded facet, and (iii) all its facets have the same set of vertices. The polyhedra in both of the above examples, however, are quite different with respect to these criteria. In fact, each of them has a full-dimensional vertex set (i.e., the convex hull of the vertices is full-dimensional), bounded facets, and the property that no two facets have the same vertex set. Furthermore, in the four-dimensional example, the vertex sets of the facets even form an clutter.

Nevertheless, any ambiguities in reconstructing the face poset of an unbounded polyhedron from its vertex-facet incidences arise from some degeneracy of P.

Theorem 3.7. Let P be a simple polyhedron. Then $\mathcal{F}(P)$ can be computed from the vertex-facet incidences of P.

Proof. Let v be a vertex of a simple d-polyhedron P and let $F_1, \ldots, F_d$ be the facets of P that contain v. The edges and extremal rays containing v are precisely the sets

$$\bigcap_{i \in \{1,\ldots,d\} \setminus \{i_0\}} F_i \qquad \text{for } i_0 = 1, \ldots, d.$$

Since we can compute the (finite) edges of P from a vertex-facet incidence matrix, we can also (combinatorially) deduce the extremal rays of P and the information which ray is contained in which facets. From that we can compute the entire face poset of P. □

The example of cones shows that without dimension information one can (in general) not decide from the vertex-facet incidences of a polyhedron whether it is simple.

The example in Figure 3.8 shows that an analogous result to Theorem 3.7 for simplicial polyhedra does not exist; the two simplicial polyhedra in the figure are combinatorially different, but have isomorphic vertex-facet incidences. This relies on the fact that that one can flip the "outer parts" around

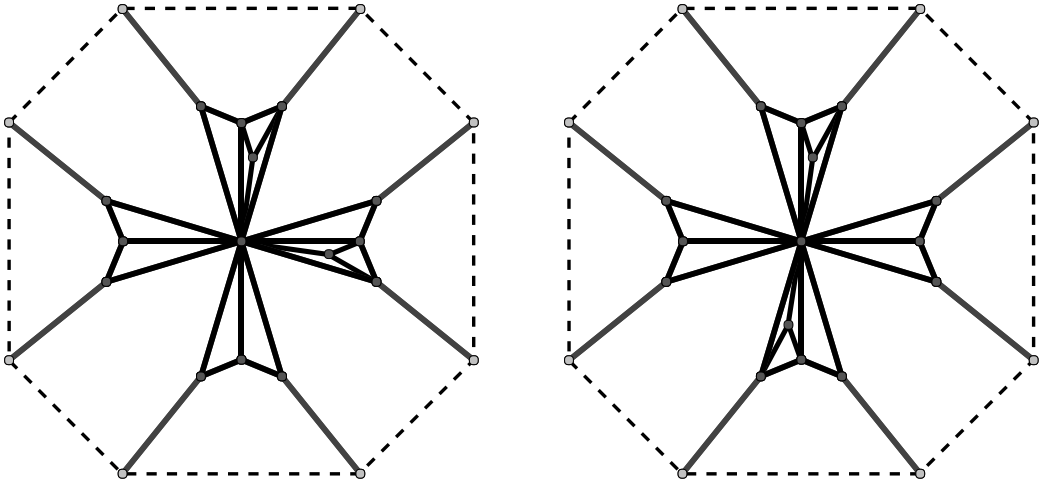


Figure 3.8: Two unbounded combinatorially different *simplicial* 3-polyhedra that have isomorphic vertex-facet incidence matrices.

the center vertex without changing the vertex-facet incidences (up to isomorphism). In general, the notion of a simplicial polyhedron is weaker than simpliciality for polytopes. For instance, there is no duality notion for (the face lattices of) unbounded polyhedra, so that simple and simplicial are not symmetrical by duality.

All algorithms described in this section can be implemented such that their running time is bounded by a polynomial in $|\mathcal{V}(P)|$.

To summarize the results in this section: We presented large classes of (unbounded) polyhedra whose combinatorial structure can be reconstructed from their vertex-facet incidences, as well as several examples of polyhedra for which this is not possible. Unfortunately, these results do not yield a *characterization* of the class of those polyhedra that allow such reconstructions.

3.3 Detecting Boundedness

In this section, we show that, given the vertex-facet incidences of a pointed polyhedron P, one can decide whether it is bounded or not. It turns out that this only depends on the Euler characteristic of (the order complex of) $\mathcal{V}(P)$, which can be computed via the Möbius function of $\mathcal{V}(P)$. We refer to Björner [29] for terminology and facts of combinatorial topology used in this section.

Let Π be a finite poset. Recall that the *order complex* $\Delta(\Pi)$ of Π is the finite simplicial complex of all chains in Π. In the following, when applying

terminology from topology in the context of finite posets such as Π, we refer to $\|\Delta(\Pi)\|$ (i.e., any geometric realization of $\Delta(\Pi)$, endowed with its standard topology); e.g., if $\|\Delta(\Pi)\|$ is a sphere, we say that Π is a sphere.

It is well-known that the order complex $\Delta(\mathcal{F}(P))$ of the face poset of a bounded d-polytope P is isomorphic (as a simplicial complex) to the barycentric subdivision of the boundary ∂P of P. In particular, the topological type of $\mathcal{F}(P)$ is as follows.

Lemma 3.8. If P is a d-polytope, then $\mathcal{F}(P)$ is homeomorphic to the $(d-1)$-sphere.

Remember that an unbounded (pointed) polyhedron P can be projectively transformed to a polytope $\overline{P}$. We then can consider $\mathcal{F}(P)$ as the sub-poset of $\mathcal{F}(\overline{P})$ consisting of all faces of $\overline{P}$ that are not contained in F_∞. Thus, we will identify $\Delta(\mathcal{F}(P))$ with the sub-complex of $\Delta(\mathcal{F}(\overline{P}))$ that is induced by all chains $\{F\}$, where F is a face of $\overline{P}$ with $F \not\subseteq F_\infty$.

Lemma 3.9. If P is an unbounded (pointed) polyhedron, then $\mathcal{F}(P)$ is contractible.

Proof. By Lemma 3.8, $\|\Delta(\mathcal{F}(\overline{P}))\|$ is homeomorphic to a sphere. The faces of $\overline{P}$ are the vertices of $\Delta(\mathcal{F}(\overline{P}))$, since they are the one-element chains in $\mathcal{F}(\overline{P})$. The induced subcomplexes $A = \Delta(\mathcal{F}(P))$ and $B = \Delta(\mathcal{F}(F_\infty))$ cover all vertices of $\Delta(\mathcal{F}(\overline{P}))$. By using barycentric coordinates, it is seen that $\|\Delta(\mathcal{F}(\overline{P}))\| \setminus \|B\|$ retracts onto $\|A\|$. Thus, $\|A\|$ has the same homotopy type as $\|\Delta(\mathcal{F}(\overline{P}))\| \setminus \|B\|$, where the latter is a simplicial sphere minus an induced ball. Hence, $\mathcal{F}(P)$ is contractible. □

These two lemmas allow to distinguish between the face posets of bounded and unbounded polyhedra. Of course, there are simpler ways to decide whether a face poset belongs to a bounded or to an unbounded polyhedron, e.g., checking if every rank one element (edge) is a join of two rank zero elements (vertices). In general, however, we cannot reconstruct the face poset of a polyhedron P from its vertex-facet incidences (see Section 3.2). Instead, we are interested in criteria allowing to distinguish between bounded and unbounded polyhedra that can be computed from $\mathcal{V}(P)$. It turns out that Lemmas 3.8 and 3.9 can be exploited for this.

Lemma 3.10. Let P be a pointed polyhedron. Then the face poset $\mathcal{F}(P)$ is homotopy equivalent to the poset $\mathcal{V}(P)$.

Proof. Consider the poset maps $\phi : \mathcal{F}(P) \to \mathcal{V}(P)$, mapping a face F of P to $\operatorname{vert} F$, and $\psi : \mathcal{V}(P) \to \mathcal{F}(P)$, mapping the vertex set S of a face to

the minimal face (w. r. t. inclusion) containing S. Both ϕ and ψ are order preserving. Moreover, we have $\phi(\psi(S)) = S$ and $\psi(\phi(F)) \subseteq F$. Hence, $f(F) := \psi(\phi(F))$ is an order preserving map from $\mathcal{F}(P)$ into itself, such that $f(F) \subseteq F$. Using the order homotopy theorem, $\mathcal{F}(P)$ is homotopy equivalent to the image $f(\mathcal{F}(P))$ (see Björner [29, Corollary 10.12]). In fact, we have $f(f(F)) = f(F)$ and $f(\mathcal{F}(P))$ is a strong deformation retract of $\mathcal{F}(P)$. This proves the lemma, since ψ is a poset isomorphism from $\mathcal{V}(P)$ onto $\psi(\mathcal{V}(P)) = \psi(\phi(\mathcal{F}(P))) = f(\mathcal{F}(P))$. □

The reduced Euler characteristic of (the order complex of) a poset Π is denoted by $\widetilde{\chi}(\Pi)$, i.e.,

$$\widetilde{\chi}(\Pi) = \sum_{i=-1}^{D} (-1)^i \, f_i(\Delta(\Pi)),$$

where $f_i(\Delta(\Pi))$ is the number of i-faces of $\Delta(\Pi)$, and D is the dimension of $\Delta(\Pi)$.

By collecting these pieces, we get the following result:

Theorem 3.11. Let P be a pointed polyhedron. Then P is bounded if and only if $\widetilde{\chi}(\mathcal{V}(P)) \neq 0$.

Proof. The reduced Euler characteristic of a $(d-1)$-sphere equals $(-1)^{d-1}$, while the reduced Euler characteristic of a contractible space vanishes. Thus the claim follows from Lemma 3.8, Lemma 3.9, and Lemma 3.10. □

In particular, this result shows that a polytope and an unbounded polyhedron cannot have isomorphic vertex-facet incidence matrices.

As an example, consider the case where the unbounded polyhedron P has a face F which contains all vertices of P. Then $\Delta(\mathcal{V}(P))$ is a cone over F, i.e., the facets of $\Delta(\mathcal{V}(P))$ (maximal chains in $\mathcal{V}(P)$) all contain the vertex F; in particular, $\Delta(\mathcal{V}(P))$ is contractible and thus $\widetilde{\chi}(\mathcal{V}(P)) = 0$.

The reduced Euler characteristic of the poset $\mathcal{V}(P)$ can be computed efficiently as follows. By adjoining an artificial top element $\hat{1}$ and an artificial bottom element $\hat{0}$, the poset $\mathcal{V}(P)$ becomes a lattice $\hat{\mathcal{V}}(P)$ (see also Section 2.5). Note that we also adjoin $\hat{1}$ in the case where $\mathcal{V}(P)$ already has a top element corresponding to a face containing all vertices of P.

For every element $S \in \hat{\mathcal{V}}(P)$, define the *Möbius function*, see Rota [102] and Stanley [111], by

$$\mu(S) = \begin{cases} 1 & \text{if } S = \hat{0}\,, \\ -\sum_{S' \subset S} \mu(S') & \text{otherwise.} \end{cases} \tag{3.1}$$

The *Möbius number* $\mu(\mathcal{V}(P)) := \mu(\hat{\mathcal{V}}(P)) = \mu(\hat{1})$ of $\mathcal{V}(P)$ can be computed in time bounded polynomially in $|\mathcal{V}(P)|$. It is well-known (see Rota [102] and Stanley [111, 3.8.6]) that

$$\mu(\mathcal{V}(P)) = \widetilde{\chi}(\mathcal{V}(P)). \tag{3.2}$$

This proves the following complexity result:

Corollary 3.12. There is an algorithm that, given a vertex-facet incidence matrix of a polyhedron P, decides if P is bounded. Its running time is bounded by a polynomial in $|\mathcal{V}(P)|$.

See Theorem 3.14 for a version with concrete running time estimates.

Actually, Theorem 3.11 allows to infer even more from the vertex-facet incidences of a polyhedron P. Once we have computed $\mathcal{V}(P)$, we can also determine $\hat{\mathcal{V}}(F)$ for every facet F of P (since we know $\operatorname{vert} F$ for every facet F of P). This is the interval between $\hat{0}$ and $\operatorname{vert} F$ in the lattice $\hat{\mathcal{V}}(P)$, where we have to add an additional top element $\hat{1}$.

Corollary 3.13. There is an algorithm that, given a vertex-facet incidence matrix of a polyhedron P, decides which facets of P are bounded. Its running time is bounded by a polynomial in $|\mathcal{V}(P)|$.

3.4 Computing the Euler Characteristic

From a computational point of view, the main question left open in the last section is the complexity of computing the Euler characteristic of $\mathcal{V}(P)$. We first give a sharpened result of Corollary 3.12.

Theorem 3.14. Given a vertex-facet incidence matrix of a polyhedron P, such that the collection of vertex sets of facets form a clutter, the Euler characteristic $\widetilde{\chi}(\mathcal{V}(P))$ of $\mathcal{V}(P)$ can be computed in $\mathcal{O}(\min\{n, m\} \cdot \alpha \cdot \varphi)$ time, where n is the number of vertices, m is the number of facets, α is the number of vertex-facet incidences of P, and φ is the size of $\mathcal{V}(P)$.

Proof. We first note that under this condition $\hat{\mathcal{V}}(P)$ is an atomic and coatomic lattice. It is coatomic by construction, since each face (element of $\hat{\mathcal{V}}(P)$) is the intersection (meet) of the vertex sets of the facets (coatoms). Furthermore, each face is described by its vertex set, i.e., $\hat{\mathcal{V}}(P)$ is atomic. As noted in Remark 4.7, Algorithm 1 of Chapter 4 can compute the Hasse diagram of any atomic and coatomic lattice from its atom-coatom incidences. Translated to the case of $\hat{\mathcal{V}}(P)$, its running time is $\mathcal{O}(\min\{n, m\} \cdot \alpha \cdot \varphi)$,

where n is the number of atoms (vertices), m is the number of coatoms (facets), α is the number of coatom-atom incidences, and φ is the size of the lattice.

Again by equation (3.2), it suffices to compute the Möbius function of $\hat{\mathcal{V}}(P)$, as defined in equation (3.1). This can be done in $\mathcal{O}(\min\{n,m\} \cdot \varphi)$ time as follows. Starting at the bottom node $\hat{0}$, perform a breadth-first search on the Hasse diagram of $\hat{\mathcal{V}}(P)$, which has size at most $\mathcal{O}(\min\{n,m\} \cdot \varphi)$. Initially, we set $\mu(\hat{0}) = 1$ and for each node $F \neq \hat{0}$ we let $\mu(F) = 0$. When processing node F, we distribute the current value $\mu(F)$ to each of its neighbors, that is, for a neighbor $G \supset F$ (i.e., there exists an edge (F,G) in the Hasse diagram) $\mu(G)$ is updated to $\mu(G) - \mu(F)$. When all nodes are processed, $\mu(\hat{1})$ stores the Möbius function of $\hat{\mathcal{V}}(P)$. In fact, in each node F, $\mu(F)$ gives the value of the Möbius function of F. This shows that we can compute $\mu(\mathcal{V}(P)) = \widetilde{\chi}(\mathcal{V}(P))$ in a total time of $\mathcal{O}(\min\{n,m\} \cdot \alpha \cdot \varphi)$. □

Remark 3.15. If the clutter condition in Theorem 3.14 is violated, then P cannot be bounded. Nonetheless, the Euler characteristic of $\mathcal{V}(P)$ can be computed by taking the inclusionwise maximal elements in the collection of vertex sets of facets and applying Theorem 3.14. Since the face lattices of polyhedra are coatomic the collection of sets of facets containing a vertex always forms a clutter.

It is an open question whether one can compute $\widetilde{\chi}(\mathcal{V}(P))$ without enumerating the Hasse diagram, or to be more specific, in polynomial time in the size of the incidences. This leads to the following considerations.

Let a vertex-facet incidence matrix of a polyhedron P be given. Define the simplicial complex $\Gamma(P)$ having the vertex sets of facets of P as its facets. The complex $\Gamma(P)$ is a so-called cross-cut complex; compare also Section 2.6. We have that $\Delta(\mathcal{V}(P))$ and $\Gamma(P)$ are homotopy equivalent by the nerve theorem, see [29, Theorem 10.7]. Hence, to compute $\widetilde{\chi}(\mathcal{V}(P))$ one can compute $\widetilde{\chi}(\Gamma(P))$, i.e., we are faced with the following problem:

Euler Characteristic of Simplicial Complexes: Given a simplicial complex Δ with vertex set $[n]$ that is defined by its facets $F_1, \ldots, F_m \subseteq [n]$, compute the reduced Euler characteristic $\widetilde{\chi}(\Delta)$.

Here we defined $[n] := \{1, \ldots, n\}$. The complexity of this problem is currently unknown. Hence, the question is whether one can compute $\widetilde{\chi}(\Delta)$ in polynomial time in n and m. See also Problem 31 of [76].

We note the following easy fact for future reference:

Lemma 3.16. If the problem to compute the Euler characteristic of a simplicial complex given by its facets is $\mathcal{NP}$-hard, then it is hard for pure simplicial complexes.

Proof. Let $F_1, \ldots, F_m$ be the facets of a simplicial complex Δ with vertex set $[n]$. Let $d := \dim(\Delta)$ be the dimension of Δ. If there exists a facet F_j with $\dim(F_j) < d$, we can construct a second simplicial complex Δ' with vertex set $[n+1]$ defined by the facets $F_1, \ldots, F_{j-1}, F_j \cup \{n+1\}, F_{j+1}, \ldots, F_m$. Since for each face $F \subseteq F_j$ of size i, $F \cup \{n+1\} \in \Delta'$ is a face of size $i+1$, it follows that

$$f_{i+1}(\Delta') = f_{i+1}(\Delta) + \binom{|F_j|}{i+1} \quad \text{for } i = 0, 1, \ldots, d-1.$$

Hence, this yields:

$$\begin{aligned}\widetilde{\chi}(\Delta') &= -1 + \sum_{i=0}^{d}(-1)^i f_i(\Delta') = -1 + \sum_{i=0}^{d}(-1)^i \left(f_i(\Delta) + \binom{|F_j|}{i} \right) \\ &= -1 + \sum_{i=0}^{d}(-1)^i f_i(\Delta) + \sum_{i=0}^{d}(-1)^i \binom{|F_j|}{i} = \widetilde{\chi}(\Delta) + 0.\end{aligned}$$

By performing at most $m \cdot d \leq m \cdot n$ such constructions we arrive at a pure simplicial complex $\widetilde{\Delta}$ with $\widetilde{\chi}(\Delta) = \widetilde{\chi}(\widetilde{\Delta})$. □

Since the Euler characteristic is determined by the number of faces of the simplicial complex, one can ask whether it is possible to compute the f-vector of Δ in polynomial time in n and m and then compute the Euler characteristic from it. The next proposition shows that this is unlikely.

Proposition 3.17. Let Δ be a pure simplicial complex with vertex set $[n]$, defined by its facets $F_1, \ldots, F_m$. Then the problem to compute $f_i(\Delta)$ for some $i \geq 1$ is $\#\mathcal{P}$-complete.

Proof. The following problem is in $\mathcal{NP}$: Given a set $S \subseteq [n]$ of size i, decide whether $S \in \Delta$? The number of times the answer to a "guess" S is "yes" is exactly $f_i(\Delta)$. Hence the corresponding counting problem is in $\#\mathcal{P}$ (see Garey and Johnson [62] for a definition).

To prove $\#\mathcal{P}$-hardness, consider an instance of the STABLE SET problem, i.e., a simple graph $G = (V, E)$ and an integer K. A *stable set* in G is a subset S of V such that S contains no edge of G, i.e., if $\{u, v\} \in E$ then $u \notin S$ or $v \notin S$. The corresponding counting problem is to compute the number $S^{\geq}(K)$ of stable sets of size at least K. This problem is $\#\mathcal{P}$-hard: It is well known that counting the vertex covers of size at most K is $\#\mathcal{P}$-complete (see [62, p. 169]). Since a set $V' \subseteq V$ is a vertex cover if and only if $V \setminus V'$ is a stable set, the claim follows. Let $S^{=}(k)$ be the number of stable sets of size k. Clearly, it suffices to compute $S^{=}(n), S^{=}(n-1), \ldots, S^{=}(K)$ to

compute $S^{\geq}(K)$. It follows that the problem to compute $S^{=}(K)$ is $\#\mathcal{P}$-hard (in fact, $\#\mathcal{P}$-complete).

Let n be the number of vertices and m be the number of edges of G. Let V be the vertex set of a simplicial complex Δ that is defined by the minimal non-faces $e \in E$. Clearly, $S \subseteq V$ is a stable set in G if and only if $S \in \Delta$. Hence, to solve the Stable Set problem, we just have to ask if $\dim(\Delta) + 1 \geq K$. Since we can compute $\dim(\Delta)$ by computing $f_i(\Delta)$ for at most n values of i, it follows that computing $f_i(\Delta)$ is $\#\mathcal{P}$-hard.

We now construct a simplicial complex $\overline{\Delta}$ (the *dual* complex), which is given by its facets. Define $\overline{\Delta}$ by the facets $V \setminus e$ for each edge $e \in E$. Note that $\overline{\Delta}$ is pure. We have that a set $S \subseteq V$ is a face of Δ if and only if $\overline{S} := V \setminus S$ is *not* a face of $\overline{\Delta}$. Hence, $f_i(\Delta) + f_{n-i-1}(\overline{\Delta}) = \binom{n}{i+1}$. It follows that one can efficiently compute $f_i(\Delta)$ from $f_{n-i-1}(\overline{\Delta})$, which completes the proof. □

Proposition 3.17 strengthens the result in [76, Problem 32] to pure simplicial complexes. To our knowledge no other proof of this fact appeared in the literature. Bayer and Stillman [23] use a similar idea as above to prove that computing the codimension of a monomial ideal in the ring of polynomials $k[x_1, \ldots, x_n]$ (for some field k) is $\mathcal{NP}$-hard.

If the number i is fixed, one can enumerate all $\binom{n}{i+1}$ faces of dimension i in polynomial time and hence solve the problem of computing $f_i(\Delta)$ in polynomial time.

3.5 Simple and Simplicial Polyhedra

In this section, we generalize the following well known fact about polytopes to not necessarily bounded polyhedra: A *polytope* which is both simple and simplicial is a simplex or a polygon (see Ziegler [115, Exercise 0.1]). We will prove the following theorem.

Theorem 3.18. For $d \geq 2$, every simple and simplicial d-polyhedron is a simplex or a polygon. In other words, if $d \geq 2$, then unbounded simple and simplicial polyhedra do not exist.

For $d = 0$, there are no unbounded polyhedra, and all one-dimensional polyhedra are simple and simplicial. Hence, we restrict our attention to the case $d \geq 2$ for the rest of this section.

If the dimension is known, simplicity as well as simpliciality are already determined by the vertex-facet incidences of a polyhedron (see Section 3.1). The first part of the proof of Theorem 3.18 considers 0/1-matrices M with

exactly d ones in each row and column – a class which includes the vertex-facet incidence matrices of simple and simplicial polyhedra. It is shown that the graph Γ_M, which is a natural generalization of the graph Γ_P of a polyhedron P, is either a complete graph or a cycle, if it is connected. Under this condition, the matrix M is isomorphic to a circulant matrix (see Section 3.5.2). In Section 3.5.3, we deduce that every simple and simplicial polyhedron has a circulant vertex-facet incidence matrix. The proof of Theorem 3.18 is then completed by showing that no unbounded d-polyhedron (with $d \geq 2$) can have a circulant vertex-facet incidence matrix.

Furthermore, Propositions 3.27 and 3.28 yield characterizations of those polyhedra that have circulant vertex-facet incidence matrices. The motivation for this characterization arose in the context of generalized antiwebs, whose incidence matrices are circulant matrices, see Section 1.3.3. In fact, it was this characterization from which Theorem 3.18 was deduced.

3.5.1 0/1-Matrices with Row and Column Sum d and Connected Graph

Throughout this section, let M be a 0/1 matrix of size $m \times n$ that has exactly $d \geq 2$ ones in each row and in column. Double counting yields that necessarily $n = m$. Hence, in the following we assume that M is of size $n \times n$. Let Γ_M be the graph with node set $C := \{0, 1, \ldots, n-1\}$, corresponding to the columns of M, and an edge between nodes i and j, if columns i and j differ in exactly two rows (have exactly $d-1$ ones in common rows). Throughout this section, we assume that Γ_M is connected. We then have $n > d$ (otherwise either Γ_M is not connected or $n = d = 1$).

Under this assumption, we investigate the structure of M in this section. It will turn out that Γ_M must be a cycle or a complete graph. It will follow from this fact that M is isomorphic to the circulant matrix $M(n, d)$, defined in Section 3.5.2 below.

Matrices like M arise as the vertex-facet incidence matrices of simple and simplicial polyhedra, which is our main motivation for studying such matrices. In this case, we have $\Gamma_M \cong \Gamma_P$, which is connected by Lemma 3.1. The reader might keep these examples in mind when reading the following proofs. In [74], the proof of Theorem 3.18 is formulated entirely in terms of polyhedra. The part of the proof given in this section is slightly more general: It works for arbitrary matrices M as above, which requires a more elaborate proof of Lemma 3.24.

Denote, as above, by $C := \{0, 1, \ldots, n-1\}$ the set of columns of M (the nodes of Γ_M) and by $R := \{0, 1, \ldots, n-1\}$ the set of rows of M. For $S \subseteq C$ let $R(S)$ be the set of all rows of M that have a one in each column of S.

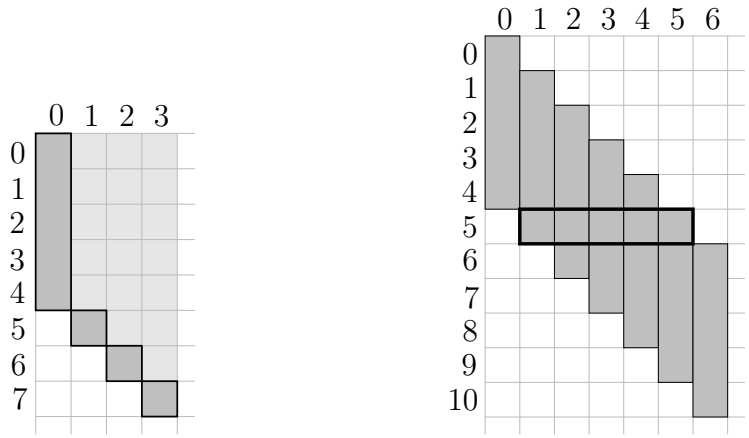


Figure 3.9: Left: Illustration of Lemma 3.20 for $d = 5$, $S = \{0, 1, 2, 3\}$, $\ell = 3$, and $\delta(S) = \{0, \ldots, 7\}$. Ones are indicated by the dark gray areas; zeros are not shown. **Right:** Illustration of Lemma 3.21 for $d = 5$, $|S| = 5$; here, $S = \{1, 2, 3, 4, 5\}$ and $R(S) = \{5\}$.

Likewise, for $S \subseteq R$ let $C(S)$ be the set of columns of M that have a one in each row of S. This means that $\{c, c'\}$ is an edge in Γ_M if and only if $|R(\{c, c'\})| = d - 1$. For $c \in C$, we often abbreviate $R(\{c\})$ by $R(c)$ and similarly for $C(\cdot)$.

Lemma 3.19. No two rows of M are the same.

Proof. Suppose that there are two rows $r_1, r_2 \in R$ of M ($r_1 \neq r_2$) such that $C(r_1) = C(r_2) =: S$. Because $n > d > 1$ and Γ_M is connected, there must be a column $c \notin S$ that is a neighbor of some node $c' \in S$ in Γ_M. Hence, we have $|R(\{c, c'\})| = d-1$. Since $|R(c')| = d$ and $r_1, r_2 \in R(c') \supseteq R(\{c, c'\})$, it follows that $r_1 \in R(\{c, c'\})$ or $r_2 \in R(\{c, c'\})$, which in both cases yields a contradiction to $c \notin S$. □

For $S \subseteq C$, define $\delta(S) \subseteq R$ to be the set of those rows of M that have a one in *at least* one column in S.

Lemma 3.20. Let $S \subseteq C$ with $|S| > 0$. Then $|\delta(S)| \geq \min\{n, d + |S| - 1\}$.

Proof. If $|\delta(S)| = n$, the claim is obviously correct. Therefore, assume that $|\delta(S)| < n$. Since Γ_M is connected, the nodes in $C \setminus S = \{z_1, \ldots, z_r\}$ (where $r = n - |S|$) can be ordered such that z_{i+1} is adjacent to some vertex of $S_i := S \cup \{z_1, \ldots, z_i\}$ for each $i \in \{0, \ldots, r - 1\}$ (additionally, we define $S_r = S \cup \{z_1, \ldots, z_r\}$). Clearly, $|\delta(S_i)| \leq |\delta(S_{i-1})| + 1$, for $i = 0, \ldots, r-1$, since column z_i has $d - 1$ ones in common rows with some column in S_{i-1} and hence δ can increase by at most one.

Define ℓ to be the last index i such that $|\delta(S_i)| = |\delta(S_{i-1})| + 1$, i.e., ℓ is the last index for which we encounter a new row (ℓ is well-defined due to $|\delta(S)| < n$). Since this row must have $d-1$ ones in columns from $C \setminus S_\ell$, we have $r - \ell \geq d - 1$, which yields $n - \ell \geq d + |S| - 1$.

Furthermore, we have $|\delta(S)| + \ell \geq n$, since at least one column in S_ℓ has a one in each row (S_ℓ covers the rows). Hence, $|\delta(S)| \geq n - \ell \geq d + |S| - 1$. □

For $S \subseteq C$ let $\Gamma_M(S)$ be the subgraph of Γ_M induced by S.

Lemma 3.21. Let $S \subset C$ with $0 < |S| \leq d$, such that $\Gamma_M(S)$ is connected. Then $|R(S)| = d - |S| + 1$ holds.

Proof. Since $\Gamma_M(S)$ is a connected subgraph of the connected graph Γ_M (which has $n > d$ nodes), there is a chain $\varnothing \subset C_1 \subset C_2 \subset \cdots \subset C_d$ with $C_{|S|} = S$, such that $|C_i| = i$ and $\Gamma_M(C_i)$ is connected for all i.

For every $1 < i \leq d$, the node c with $C_i \setminus C_{i-1} = \{c\}$ is connected to some node $c' \in C_{i-1}$. From $|R(c') \setminus R(c)| = 1$, we infer $|R(C_{i-1}) \setminus R(c)| \leq 1$ and thus $|R(C_i)| \geq |R(C_{i-1})| - 1$. Together with $|R(C_1)| = d$ (since each column has d ones) and $|R(C_d)| \leq 1$ (by Lemma 3.19), this implies $|R(C_i)| = d - i + 1$ for all $1 \leq i \leq d$. □

The next lemmas show that Γ_M has a very special structure.

Lemma 3.22. If Γ_M contains a cycle of size $k > d$, then Γ_M is the cycle itself or a complete graph on $n = d + 1$ nodes.

Proof. Let $(c_0, \ldots, c_{k-1}, c_0)$ be a cycle of size $k > d$ in Γ_M. Without loss of generality, the cycle is elementary, i.e., $c_0, c_1, \ldots, c_{k-1}$ are pairwise different. In this proof, all indices are taken modulo k. For $0 \leq i \leq k-1$ define the set $C_i := \{c_i, \ldots, c_{i+d-1}\}$ of size d. Clearly, $\Gamma_M(C_i)$ is connected, since it contains part of the cycle. By Lemma 3.21, there exists exactly one row r_i with $R(C_i) = \{r_i\}$. Conversely, we have $C(r_i) = C_i$, since there are d ones in each row. Hence, the rows $r_0, \ldots, r_{k-1}$ are pairwise distinct. This means that $R(c_i) = \{r_{i-d+1}, \ldots, r_i\}$ (since we have d ones in each column). Therefore, every column that is adjacent to one of the columns $c_0, \ldots, c_{k-1}$ must have a one in at least one (more precisely, $d-1$) of the rows $r_0, \ldots, r_{k-1}$ and thus lies in $\{c_0, \ldots, c_{k-1}\}$. Since Γ_M is connected, this implies $n = k$. For $n = d + 1$, this immediately yields that Γ_M is a complete graph on $n = d + 1$ nodes, while for $n > d + 1$ one finds that Γ_M is the cycle $(c_0, \ldots, c_{k-1}, c_0)$ (since, in this case, $|R(\{c_i, c_j\})| = d - 1$ if and only if $j \equiv i \pm 1 \bmod k$). □

Lemma 3.23. If Γ_M contains a cycle of length $k \leq d$, then Γ_M is a complete graph on $n = d + 1$ nodes.

Proof. Let $(c_0, \dots, c_{k-1}, c_0)$ be a cycle in Γ_M of length $k \leq d$. Then for $i = 1, \dots, k-1$ define $\widetilde{C}_i = \{c_0, \dots, c_i\}$. Taking all indices modulo k, we have $|R(\{c_i, c_{i+1}\})| = d - 1$ for each i. Hence, there are rows r_i and s_i with

$$R(c_i) \setminus R(c_{i+1}) = \{r_i\} \quad \text{and} \quad R(c_{i+1}) \setminus R(c_i) = \{s_i\}\,.$$

It follows that

$$R(\widetilde{C}_{k-1}) = R(\widetilde{C}_0) \setminus \{s_0, \dots, s_{k-1}\}\,. \tag{3.3}$$

If Γ_M is not complete, we have $n > d + 1$ and we infer from Lemma 3.20 that $|\delta(\widetilde{C}_2)| \geq d + 2$, which implies $s_0, s_1 \notin R(\widetilde{C}_0)$ (with $s_0 \neq s_1$). Due to $\{r_0, \dots, r_{k-1}\} = \{s_0, \dots, s_{k-1}\}$, Equation (3.3) implies

$$|R(\widetilde{C}_{k-1})| \geq |R(\widetilde{C}_0)| - (k-2) = d - k + 2\,,$$

contradicting Lemma 3.21. □

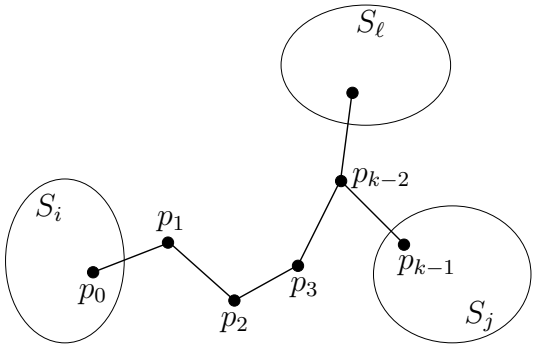
By the above two lemmas, Γ_M cannot contain any cycles, unless it is complete or a cycle itself. Thus, we are left with the case of Γ_M not containing any cycles at all.

Lemma 3.24. Γ_M cannot be a tree.

Proof. (a) We first consider the case that the induced graph $\Gamma_M(C(r))$ is connected for every $r \in R$. Assume Γ_M is a tree. Let $c \in C$ be a column of degree one in Γ_M and let $\{r_1, r_2, \dots, r_d\} = R(c)$ be the rows where column c has ones. Since Γ_M is connected, there exists (exactly) one edge to another column c'. W. l. o. g., let $R(\{c, c'\}) = \{r_2, \dots, r_d\}$. But then $\Gamma_M(C(r_1))$ is not connected, which is a contradiction.

(b) Consider the case that there exists $r \in R$ such that $\Gamma_M(C(r))$ is not connected. We will first show that $\Gamma_M(C(r))$ has at most two connected components for every $r \in R$. This is always true under our assumptions on M and Γ_M. After that, we will derive a contradiction if Γ_M is a tree.

Let $\Gamma_M(C(r))$ be disconnected and let $S_1, \dots, S_m \subseteq C$ be its connected components. Define the set $U_r := C \setminus C(r)$. Since Γ_M is connected, for every pair $1 \leq i \neq j \leq m$ there exists a path connecting S_i and S_j that uses only nodes of U_r, except for the end nodes. Let $P = (p_0, p_1, \dots, p_{k-1})$ be the shortest one among these paths. Observe that $p_0 \in S_i$, $p_{k-1} \in S_j$, and $k \geq 3$ (otherwise we would have an edge between S_i and S_j). In each step on the path from p_0 to p_{k-1}, $\delta(P)$ can increase by at most one. Because $r \notin R(p_{k-2})$ (since $p_{k-2} \in U_r$), but $r \in R(p_{k-1})$, $\delta(P)$ cannot increase in the last step. Since $|\delta(\{p_0\})| = d$, we have $\delta(P) \leq d + k - 2$.



The induced graph $\Gamma_M(P)$ is connected and hence, by Lemma 3.20, $\delta(P) \geq \min\{n, d+k-1\}$. Therefore, we have $d+k-2 \geq n$, i.e., $k \geq n-d+2$. The graph $\Gamma_M(C(r))$ has d nodes, two of which are on P. It follows that P contains all of the nodes outside $C(r)$, i.e., $\{p_1, p_2, \ldots, p_{k-2}\} = U_r$. Hence, we have $d + k - 2 = n$.

Assume for the sake of contradiction that $m \geq 3$ and let $1 \leq \ell \leq m$, $\ell \neq i$, and $\ell \neq j$. Let P' be a path from S_i to S_ℓ. Clearly, P' uses only nodes from P and $C(r)$, otherwise P would not contain all nodes outside $C(r)$. Hence, there exists an edge from some node $p \in \{p_1, \ldots, p_{k-2}\}$ to a node of S_ℓ. By the choice of P as the shortest path joining the two connected components, we have that $p = p_{k-2}$. By symmetry, it follows that $k = 3$ and hence $n = d + 1$. But then $\Gamma_M = K_n$, a contradiction since we assumed that $\Gamma_M(C(r))$ was disconnected.

So far, we proved that $\Gamma_M(C(r))$ has at most two connected components for every $r \in R$. Assume that Γ_M is a tree and hence $\Gamma_M(C(r))$ is a forest. Because Γ_M has $n-1$ edges and each edge "appears" in $d-1$ rows, double-counting the edges of the tree Γ_M yields:

$$(n-1) \cdot (d-1) = \sum_{r \in R} e_r,$$

where e_r is the number of edges in $\Gamma_M(C(r))$. Let A be the set of rows r for which $\Gamma_M(C(r))$ is disconnected. Since $\Gamma_M(C(r))$ is a forest on d nodes, it has exactly $d-1$ edges if it is connected and $d-2$ edges if it is not connected, because we have at most two connected components. Therefore,

$$\sum_{r \in R} e_r = (d-1) \cdot (n - |A|) + (d-2) \cdot |A|,$$

from which $|A| = d-1$ follows. Hence, there exist $d-1$ graphs $\Gamma_M(C(r))$ that are disconnected (with two connected components). For each $r \in A$, no leaf

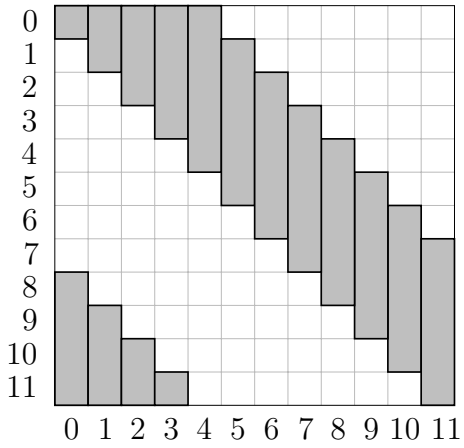


Figure 3.10: Illustration of the circulant matrix $M(12, 5)$.

of Γ_M can be in U_r. Therefore, each leaf of Γ_M has to be contained in $C(r)$ for every $r \in A$. It follows that any two leaves of Γ_M are contained in $d-1$ common rows. But then they should be adjacent in Γ_M. Contradiction. □

Altogether this proves the following:

Proposition 3.25. Let M be a 0/1 matrix of size $n \times n$ that has exactly $d \geq 2$ ones in each row and in each column. If Γ_M is connected, then it is an n-cycle or a complete graph on $n = d + 1$ nodes.

3.5.2 Circulant Matrices

Let n, d be integers satisfying $1 \leq d \leq n$. The (n, d)-*circulant* $M(n, d)$ is the $(n \times n)$-matrix with 0/1 entries whose coefficients m_{ij} $(i, j \in \{0, \ldots, n-1\})$ are defined as follows:

$$m_{ij} = \begin{cases} 1 & \text{if } j \in \{i, i+1 \bmod n, \ldots, i+d-1 \bmod n\} \\ 0 & \text{otherwise.} \end{cases}$$

See Figure 3.10 for an example. For $d \geq 1$, the $(d+1, d)$-circulant is a vertex-facet incidence matrix of the d-simplex, and for $n \geq 3$, the $(n, 2)$-circulant is a vertex-facet incidence matrix of the (2-dimensional) n-gon.

Two matrices M and M' are *isomorphic* if there exist permutations of the rows and columns of M such that the result equals M'.

Proposition 3.26. Let M be a 0/1 matrix of size $n \times n$ that has exactly $d \geq 2$ ones in each row and in each column. Then M is isomorphic to the circulant matrix $M(n, d)$ if and only if Γ_M is connected.

Proof. If M is isomorphic to $M(n,d)$, then clearly Γ_M is connected, since $\Gamma_M \cong \Gamma_{M(n,d)}$ and $\Gamma_{M(n,d)}$ is connected.

For the converse assume that Γ_M is connected. By Proposition 3.25, Γ_M either is a complete graph on $n = d+1$ nodes, or it is a cycle. In the first case, M is the complement of a permutation matrix, which can be transformed to $M(n,d)$ by a suitable permutation of its rows.

In the second case, assume that the columns of M are ordered according to the cycle Γ_M, beginning at an arbitrary node. Let m_{rc} be the entry of M in row r and column c. Then call two positions (r,j) and (r,k) in M *mates* if $k \equiv j \pm 1 \pmod{n}$ and $m_{rj} = m_{rk} = 1$, for $r \in R$, $j,k \in C$. Walking around the cycle Γ_M, we find that the total number of mates in M is $n\cdot(d-1)$, because we have n edges and every edge is "contained" in precisely $d-1$ rows. Since every row of M has only d ones, it follows that in each row the ones must appear consecutively (modulo n). Denote by $s(r)$ the starting position of the block of ones in row $r \in R$. By Lemma 3.19, there are no equal rows in M. Hence, s defines a permutation of the rows of M which determines how to transform M to $M(n,d)$. □

3.5.3 Simple and Simplicial Polyhedra

We turn to the structure of simple and simplicial polyhedra, applying the results of the previous sections.

Proposition 3.27. A polyhedron P is simple and simplicial if and only if it has a circulant $M(n,d)$ as a vertex-facet incidence matrix. In this case, $\dim(P) = d$.

Proof. First assume that P has $M(n,d)$ $(1 \le d \le n)$ as a vertex-facet incidence matrix. The cases $d = 1$ (implying $n \in \{1,2\}$) and $d = n$ (which implies $d = n = 1$) are trivial. Therefore, let $2 \le d < n$. Obviously, it suffices to show $\dim(P) = d$. To each row $r \in R$ of $M(n,d)$ there corresponds a facet F_r of P. For $0 \le r \le d-1$ define $G_r = F_0 \cap \cdots \cap F_r$. Clearly, $G_r \supseteq G_{r+1}$ holds for $0 \le r < d-1$. Due to $\operatorname{vert} G_r = \operatorname{vert} F_0 \cap \operatorname{vert} F_1 \cap \cdots \cap \operatorname{vert} F_r$ it follows that $\operatorname{vert} G_r \supset \operatorname{vert} G_{r+1}$ and therefore $G_r \supset G_{r+1}$. Then the chain $F_0 = G_0 \supset G_1 \supset \cdots \supset G_{d-1}$ is (decreasing) of length $d-1$ in the face poset of P. Hence we have $\dim P \ge d$. Since each vertex must be contained in at least $\dim(P)$ facets, it follows that $\dim(P) \le d$ (each vertex of P is contained in precisely d facets).

For the converse, assume that P is a simple and simplicial d-polyhedron. The case $d = 1$ is checked easily. Thus, assume $d \ge 2$. Every vertex-facet incidence matrix M of P is a 0/1-matrix of size $m \times n$, with exactly d ones in

each row and column. Double counting again yields $m = n$. Furthermore, Γ_M is connected by Lemma 3.1. By Proposition 3.26, M is isomorphic to $M(n, d)$ and hence P has $M(n, d)$ as a vertex-facet incidence matrix. □

The following result finishes the proof of Theorem 3.18 (via Proposition 3.27).

Proposition 3.28. If a polyhedron P has the circulant $M(n, d)$ $(2 \leq d < n)$ as a vertex-facet incidence matrix, then $n = d+1$ (P is a d-simplex) or $d = 2$ (P is an n-gon).

Proof. If $n = d + 1$, then $M(n, d)$ is a vertex-facet incidence matrix of a d-simplex. By Theorem 3.11, P cannot be unbounded, and thus it must be a d-simplex itself. Therefore, in the following we will assume $n > d + 1$.

Let us first treat the case $d + 1 < n < 2d - 1$. Consider the facets F and F' corresponding to rows 0 and $n - d + 1$, respectively. If we identify the vertices of P with the column indices $\{0, \ldots, n-1\}$ of $M(n, d)$, then the vertex set of the face $G = F \cap F'$ is $\{0\} \cup \{n - d + 1, \ldots, d - 1\}$, where $\{n - d + 1, \ldots, d - 1\} \neq \varnothing$ (due to $n < 2d - 1$). By Propositions 3.27, 3.26, and 3.25, Γ_P is an n-cycle (due to $n > d + 1$). Neither vertex 1 nor vertex $n - 1$, which are the only neighbors of 0 in Γ_P, are contained in G. We conclude that the subgraph of Γ_P induced by G is disconnected, which is a contradiction to Lemma 3.1.

Hence, we can assume $n \geq 2d - 1$. Taking all indices modulo n, we have

$$\mathcal{V}(P) = \big\{\{r, r + 1, \ldots, r + s - 1\} \,:\, r \in \{0, \ldots, n - 1\}, s \in \{1, \ldots, d\}\big\}.$$

This means that $\mathcal{V}(P)$ consists of all (cyclic) intervals of $\{0, \ldots, n - 1\}$ with at least one and at most d elements. We will compute the Möbius function μ (see Section 3.3) of the lattice $\hat{\mathcal{V}}(P)$ (that arises by adding artificial top and bottom elements $\hat{1}$ and $\hat{0}$ to $\mathcal{V}(P)$). Let $\mu(s) := \mu(\{0, 1, \ldots, s - 1\})$, for each $s \in \{1, \ldots, d\}$; note that $\{0, \ldots, s - 1\} \in \mathcal{V}(P)$. Obviously, for every element $F \in \mathcal{V}(P)$ with $|F| = s$ we have $\mu(F) = \mu(s)$. In particular, one readily deduces from (3.1) that $\mu(1) = -1$ and $\mu(2) = -(1+2\cdot(-1)) = 1$. For $3 \leq s \leq d$, we then infer by induction $\mu(s) = -(1+s\cdot(-1)+(s-1)\cdot(+1)) = 0$. We finally calculate

$$\mu(\mathcal{V}(P)) = \mu(\hat{1}) = -(1 + n \cdot (-1) + n \cdot (+1)) = -1,$$

which by (3.2) and Theorem 3.11 implies that P is bounded (and, hence, an n-gon).

(Alternatively, one could derive from the nerve lemma [29, Theorem 10.7] that $\mathcal{V}(P)$ is homotopy equivalent to a circle for $n \geq 2d - 1$, and thus, P must be a polygon.) □

Chapter 4

Computing the Face Lattice of a Polytope from its Vertex-Facet Incidences

In this chapter, we deal with the computational complexity of the problem to compute (the Hasse diagram of) the face lattice of a polytope P, if vertex-facet incidences of P are given.

We give an algorithm that constructs the Hasse diagram of the face lattice of a polytope P from its vertex-facet incidences in time $\mathcal{O}(\min\{n, m\} \cdot \alpha \cdot \varphi)$, where n is the number of vertices, m is the number of facets, α is the number of vertex-facet incidences, and φ is the total number of faces of P. In most cases this is faster than previously known methods. The algorithm can be applied to other atomic and coatomic lattices, if the atom-coatom incidences are given. This is used in Section 3.4 to distinguish between the vertex-facet incidences of polytopes and unbounded polyhedra.

In the first section we give a brief introduction to the problems and terms important for this chapter. Section 4.2.1 contains a rough sketch of the algorithm, which is followed by a more detailed description in Sections 4.2.2, 4.2.3, and 4.2.4. In Section 4.2.5 we analyze the algorithm. For simple or simplicial polytopes this algorithm can be specialized to run in time $\mathcal{O}(d \cdot \alpha \cdot \varphi)$, a case we deal with in Section 4.3.1. A variant that computes the k-skeleton appears in Section 4.3.2. Furthermore, in Section 4.3.3, a version that needs significantly less memory is described that enumerates just the faces together with their dimensions (i.e., without the edges of the Hasse diagram). Finally, a modification of the algorithm that computes the face lattice of an oriented matroid from its cocircuits (Section 4.3.4) is explained.

Most parts of this chapter are joint work with Volker Kaibel and appear in [76].

4.1 Introduction

Let P be a d-dimensional polytope (d-polytope). We again let $\mathcal{F}(P)$ be the set of non-trivial faces of P (excluding $\varnothing$ and P itself), ordered by inclusion (compare Section 3.1). We denote by $\hat{\mathcal{F}}(P)$ the poset $\mathcal{F}(P)$ with an artificial

top element $\hat{1}$ and an artificial bottom element $\hat{0}$ adjoined. Then $\hat{\mathcal{F}}(P)$ is a graded, atomic, and coatomic lattice, the *face lattice* of P (see also Section 2.5). In particular, each face can be identified with its set of vertices ($\hat{\mathcal{F}}(P)$ is atomic) or the set of facets it is contained in ($\hat{\mathcal{F}}(P)$ is coatomic). In this chapter, we usually identify a face with its vertex set. Define $\varphi := |\hat{\mathcal{F}}(P)|$ and denote by $\mathcal{L}$ the Hasse diagram (as an abstract graph) of the face lattice. Hence, $\mathcal{L}$ is a directed rooted acyclic graph whose nodes correspond to the elements of $\hat{\mathcal{F}}(P)$. If ℓ_H, ℓ_G are nodes in $\mathcal{L}$ and $H, G \in \hat{\mathcal{F}}(P)$ are the corresponding faces of P, then there is an arc (ℓ_H, ℓ_G) in $\mathcal{L}$ if and only if $H \subset G$ and $\dim(G) = \dim(H) + 1$.

We consider the following (compare Problem 13 in [77]):

Combinatorial face lattice enumeration problem: Given a vertex-facet incidence matrix of a polytope P, construct the Hasse diagram $\mathcal{L}$ of the face lattice.

See Section 3.1 for a definition of a vertex-facet incidence matrix, which we repeat in Section 4.2 below.

By definition, $\mathcal{L}$ is unlabeled. Nevertheless, it might be desired to label each node of $\mathcal{L}$ corresponding to a face F with the set of (indices of) vertices contained in F, the set of (indices of) facets containing F, or with the dimension of F.

Fukuda and Rosta [59] gave an algorithm to compute the face lattice of a d-polytope P, which runs in $\mathcal{O}(\min\{n, m\} \cdot d \cdot \varphi^2)$ time, where m is the number of facets and n is the number of vertices of P. This algorithm can easily be turned into an algorithm for the combinatorial face lattice enumeration problem with the same asymptotic running time.

Since φ can be exponential in n and m (consider the d-simplex, for instance), it is desirable to have an algorithm whose running time depends only linearly on φ (and polynomially on n and m). The main result of this chapter is such an algorithm.

Connected with the combinatorial face lattice enumeration problem is the following problem (compare Problem 5 in [77]).

Geometric face lattice enumeration problem: Given a polytope P described by inequalities, compute the face lattice of P.

There are algorithms for this problem whose running time depends only linearly on φ, e.g., Fukuda, Liebling, and Margot [58]. In our context, however, no geometric data are available.

Ganter [61] described an algorithm which, given the incidences of atoms and coatoms of a general atomic and coatomic lattice L, enumerates all elements of L in lexicographic order, where each element is identified with the set

of atoms below it (which are ordered arbitrarily). Specialized to our situation, one obtains an algorithm that computes the vertex sets of the faces of P in $\mathcal{O}(\min\{n,m\} \cdot \alpha \cdot \varphi)$ steps, where α is the number of vertex-facet incidences of P. Note that $d \cdot \max\{n,m\} \leq \alpha \leq n \cdot m$, in particular, α is bounded polynomially in n and m. This algorithm, however, does not compute the inclusion relations between the faces, i.e., the edges of the Hasse diagram of the face lattice. Of course, once all (vertex sets of) faces are computed, one may construct the Hasse diagram in an obvious way afterwards, but this would require a number of steps which is quadratic in the total number φ of faces.

Inspired by Ganter's algorithm, we developed the (quite different) algorithm presented in this chapter, which computes the entire Hasse diagram in the same running time of $\mathcal{O}(\min\{n,m\} \cdot \alpha \cdot \varphi)$, see Theorem 4.6. It requires $\mathcal{O}(\varphi \cdot \min\{n,m\})$ memory (without output storage). In our algorithm, the vertex set of each face or the set of facets it is contained in, as well as its dimension, is readily available (or can be computed without increasing the asymptotic running time). Of course, this may increase the (output) storage requirements significantly.

Fukuda and Rosta [59] also considered the combinatorial face lattice enumeration problem for the special case of simple or simplicial polytopes. They presented an algorithm that computes the face lattice of a simple or simplicial polytope in $\mathcal{O}(d \cdot \varphi)$ steps, provided that in addition to the vertex-facet incidences an acyclic orientation of the graph of the polytope is given that induces precisely one sink on every nonempty face. Such an orientation is called a *good orientation* or *abstract objective function orientation* (AOF) (see Section 2.3.2). Unfortunately, no polynomial-time algorithm is known that computes a good orientation of a simple polytope P – neither if P is given by its vertex-facet incidences nor if it is specified by its whole face lattice (compare Problem 17 of [77]).

For simple or simplicial polytopes, our algorithm can be specialized such that it computes the Hasse diagram of the face lattice in $\mathcal{O}(d \cdot \alpha \cdot \varphi)$ steps from the vertex-facet incidences, where no good orientation is required (see Section 4.3.1).

The basic concepts from the theory of algorithms and data structures that play a role in this chapter can be found in any corresponding textbook (e.g., in the one by Cormen, Leiserson, Rivest, and Stein [50]).

Throughout this chapter, our running time estimates refer to the uniform time measure, i.e., every arithmetic operation/comparison takes one unit of time. However, we decided to let our statements on memory requirements refer to the bit model, since this reflects the "real" situation more closely, e.g., this introduces a term $\min\{n,m\}$ into the estimates of the working space.

4.2 The Algorithm

Define m to be the number of facets and n the number of vertices of the d-polytope P. Let $A = (a_{fv}) \in \{0,1\}^{m \times n}$ be a *vertex-facet incidence matrix* of P. Hence the facets of P can be identified with $F := \{1, \ldots, m\}$ and its vertices can be identified with $V := \{1, \ldots, n\}$, such that $a_{fv} = 1$ if facet f contains vertex v, and $a_{fv} = 0$ otherwise. Denote by α the number of vertex-facet incidences, i.e., the number of ones in A. For $S \subseteq V$, define $\mathrm{F}(S) := \{f \in F : a_{fs} = 1 \text{ for all } s \in S\}$, the set of facets containing all vertices of S. For $T \subseteq F$, define $\mathrm{V}(T) := \{v \in V : a_{tv} = 1 \text{ for all } t \in T\}$, the set of vertices contained in all facets of T.

For $S \subseteq V$, the set $\mathrm{cl}(S) := \mathrm{V}(\mathrm{F}(S))$ is the (vertex set of) the smallest face of P containing S (in lattice theoretic terms, the *join* of the elements in S). One can check easily that this defines a *closure map* on the subsets of V, i.e., for all $S, S' \subseteq V$ we have:

$$S \subseteq \mathrm{cl}(S), \qquad \mathrm{cl}(\mathrm{cl}(S)) = \mathrm{cl}(S), \qquad S \subseteq S' \;\Rightarrow\; \mathrm{cl}(S) \subseteq \mathrm{cl}(S').$$

The faces of P correspond exactly to the *closed sets* of V with respect to this closure map (i.e., sets $S \subseteq V$ with $\mathrm{cl}(S) = S$). Our algorithm crucially relies on the fact that closures can be computed fast (see Section 4.2.2).

4.2.1 The Skeleton of the Algorithm

The strategy is to build up the Hasse diagram $\mathcal{L}$ of the face lattice from bottom ($\varnothing$) to top (P). Consequently, $\mathcal{L}$ is initialized with the single face $\varnothing$ and then enlarged iteratively by adding out-neighbors of nodes that have already been constructed. We will say that a face has been *seen*, once its corresponding node in $\mathcal{L}$ has been constructed.

During the algorithm, we keep a set $\mathcal{Q}$ containing those faces that we have seen so far, but for which we have not yet inserted their out-arcs into the Hasse diagram. At each major step, we remove a face H from the set $\mathcal{Q}$ and construct the set $\mathcal{G}$ of all faces G with $H \subset G$ and $\dim(G) = \dim(H)+1$. For each face $G \in \mathcal{G}$ we check whether it has already been seen. If this is not the case, then a new node in $\mathcal{L}$ representing G is constructed, and G is added to $\mathcal{Q}$. In any case, an arc from the node corresponding to H to the node corresponding to G is inserted into $\mathcal{L}$. See Figure 4.1 for an illustration of the algorithm.

In order to compute the set $\mathcal{G}$, we exploit the fact that $\mathcal{G}$ consists of the inclusion minimal faces among the ones that properly contain H. Since the face lattice of a polytope is atomic, each face $G \in \mathcal{G}$ must be of the

form $H(v) := \mathrm{cl}(H \cup \{v\})$ for some vertex (atom) v; in particular, the Hasse diagram has at most $n \cdot \varphi$ arcs. Thus, we first construct the collection $\mathcal{H}$ of all sets $H(v)$, $v \in V \setminus H$, and then compute $\mathcal{G}$ as the set of inclusion minimal sets of $\mathcal{H}$.

Computing $H(v)$ for some $v \in V \setminus H$ requires determining a closure. In Section 4.2.2, we describe a method to perform this task in $\mathcal{O}(\alpha)$ steps. Determining the inclusion minimal sets in the collection $\mathcal{H}$ clearly could be done in $\mathcal{O}(n^3)$ steps by pairwise comparisons, since $\mathcal{H}$ has at most n elements, each of size at most n. In Section 4.2.3 we show that this can even be performed in $\mathcal{O}(n^2)$ time.

Another crucial ingredient is a data structure, described in Section 4.2.4, that allows us to locate the node in $\mathcal{L}$ representing a given face G or to assert that G has not yet been seen. This can be performed in $\mathcal{O}(\alpha)$ steps.

A summary of the analysis of the time complexity of the algorithm, along with a pseudo-code description of it, is given in Section 4.2.5.

4.2.2 Computing Closures

In order to be able to compute closures fast, we store the incidence matrix A in a *sorted sparse format* both in a row and column based way. For each

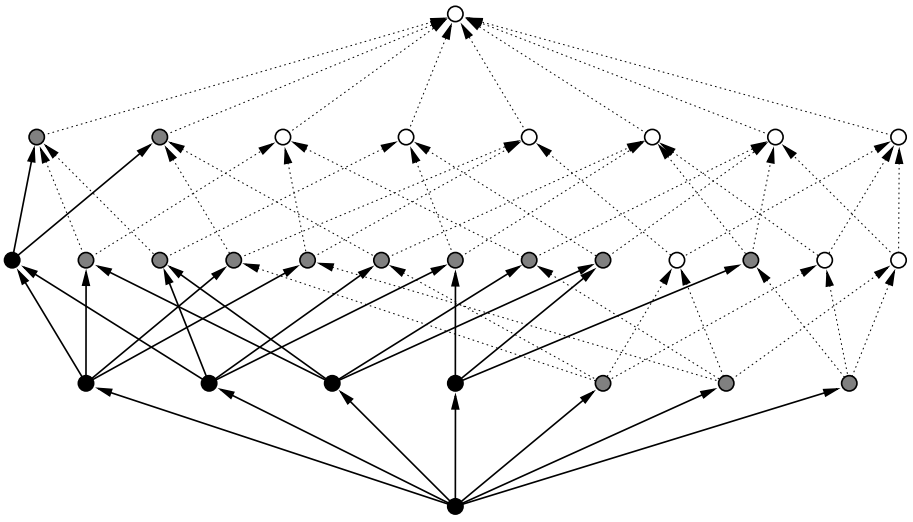


Figure 4.1: Illustration of the Hasse diagram $\mathcal{L}$ during a run of Algorithm 1 on the face lattice of the polytope on the right of Figure 3.1 on page 67. Unseen vertices are white, vertices in $\mathcal{Q}$ are marked grey, and the vertices completely processed are black. Black edges are already inserted in $\mathcal{L}$, the other edges are dotted.

vertex $v \in V$, the elements in $\mathrm{F}(\{v\}) \subseteq \{1, \ldots, m\}$ are stored increasingly in a list. Similarly, for each facet $f \in F$, we store the sorted set $\mathrm{V}(\{f\})$ in a list. This preprocessing can be performed in $\mathcal{O}(n \cdot m)$ time (which is dominated by $\mathcal{O}(n \cdot \alpha)$ and thus does not influence the estimate of the asymptotic running time in Proposition 4.5 below). The sorted sparse format uses $\mathcal{O}(\alpha \cdot \log \max\{n, m\})$ storage.

Whenever we want to compute the closure of a set $S \subseteq V$, the first step is to compute $\mathrm{F}(S)$, i.e., the intersection of the lists $\mathrm{F}(\{v\})$, $v \in S$. Since the intersection of two sorted lists can be computed in time proportional to the sum of the lengths of the two lists and because the intersection of two lists is at most as long as the shorter one, $\mathrm{F}(S)$ can be computed in time $\mathcal{O}\big(\sum_{v \in S} |\mathrm{F}(\{v\})|\big) \subseteq \mathcal{O}(\alpha)$. Similarly, $\mathrm{V}(T)$ can be computed in time $\mathcal{O}(\alpha)$ for a set $T \subseteq F$.

Lemma 4.1. The closure $\mathrm{cl}(S)$ of a set $S \subseteq V$ can be computed in $\mathcal{O}(\alpha)$ steps (provided that the vertex-facet incidence matrix is given in the sorted sparse format).

4.2.3 Identifying the Minimal Sets

Suppose that $H \subset V$ is a face of P and let $\mathcal{H}$ be the collection of the faces $H(v) := \mathrm{cl}(H \cup \{v\}) \subseteq V$, for $v \in V \setminus H$.

Our procedure to identify the set $\mathcal{G}$ of minimal sets in the collection $\mathcal{H}$ starts by assigning a label *candidate* to each vertex in $V \setminus H$. Subsequently, the label *candidate* of each vertex will either be removed or replaced by a label *minimal*. We keep the following three invariants: For each vertex v that is labeled *minimal* we have $H(v) \in \mathcal{G}$; if two different vertices v and w both are labeled *minimal*, then we have $H(v) \neq H(w)$; $\mathcal{G}$ is contained in the set of all $H(v)$ for which v is labeled *minimal* or *candidate*. Clearly, if no vertex is labeled *candidate* anymore, the set of vertices labeled *minimal* is in one-to-one correspondence to $\mathcal{G}$ via $H(\cdot)$.

Suppose there is still some v labeled *candidate* available. If $H(v) \setminus \{v\}$ contains some vertex w, then we have $H(w) \subseteq H(v)$, because $H(w)$ is the intersection of all faces containing H and w, and one of these faces is $H(v)$. Hence, if w is labeled *minimal* or *candidate*, we remove the label *candidate* from v; if there exists no such w, we label v *minimal*.

It follows by induction that the three invariants are satisfied throughout the procedure. Moreover, at each major step (choosing a *candidate* v) the number of *candidate* labels decreases by one. Since each such step takes $\mathcal{O}(n)$ time, the entire procedure has complexity $\mathcal{O}(n^2)$.

Lemma 4.2. The set $\mathcal{G}$ of inclusion minimal sets in $\mathcal{H} = \{H(v) : v \in V \setminus H\}$ can be identified in $\mathcal{O}(n^2)$ steps.

4.2.4 Locating Nodes

During the algorithm, we have to keep track of the faces that we have seen so far and the corresponding nodes in $\mathcal{L}$. To this end, we maintain a special data structure, the *face tree*. In this data structure, a face $S = \{s_1, \ldots, s_k\} \subseteq V$ (with $s_1 < \cdots < s_k$) is represented by the lexicographically smallest set $C(S) \subseteq S$ that generates S, i.e., $\mathrm{cl}(C(S)) = S$. We call $C(S)$ the *canonical spanning set* of the face S. The map $C(\cdot)$ is one-to-one; its inverse map is the closure map. See also Table 4.1 and Example 4.4 below.

The set $C(S)$ can be computed efficiently as follows. For $k = 1$ and $k = 2$, set $C(S) := S$. For $k \geq 3$, $C(S)$ is computed iteratively: Initialize $C(S)$ with the set $\{s_1, s_2\}$; at each iteration extend $C(S)$ by the smallest s_i such that $\mathrm{cl}(C(S)) \subset \mathrm{cl}(C(S) \cup \{s_i\})$. Note that $|C(S)| \leq \dim(S) + 1 \leq d + 1$. Recall that we stored the vertex-facet incidences in the sorted sparse format (see Section 4.2.2). Similarly to the method for computing closures, this computation can be performed in $\mathcal{O}(\alpha)$ steps, since just the intersections $\mathrm{F}(\{s_1\}) \cap \cdots \cap \mathrm{F}(\{s_i\})$, $i = 1, \ldots, k$, have to be computed iteratively. Then, $C(S)$ is obtained as the set of those s_i for which the intersection becomes smaller.

We now explain the structure of the face tree. Its arcs are directed away from the root. They are labeled with vertex numbers, such that no two arcs leaving the same node have the same label and on every directed path in the tree the labels are increasing. Via the sets of labels on the paths from the root, the nodes of the tree correspond to the sorted sets $C(S)$ for the faces $S \subseteq V$ that are on the path to some face S' (i.e., $C(S) \subseteq C(S')$) that has already been accessed. In particular, the root node represents the face $\varnothing$. Each node has a pointer to the corresponding node of $\mathcal{L}$. By construction, the depth of the tree is bounded by $d + 1$.

Suppose we want to find the node ℓ_S corresponding to some face $S \subseteq V$ in the part of $\mathcal{L}$ that we have already constructed or to assert that this face has not yet been seen. We first sort S (a subset of $\{1, \ldots, n\}$) increasingly in $\mathcal{O}(n)$ steps (by counting or bucket sort, see [50, Chap. 8]) and compute $C(S)$ in $\mathcal{O}(\alpha)$ steps. Then, starting from the root, we proceed (as long as possible) downwards in the face tree along arcs labeled by the successive elements of $C(S)$. Either we find an existing node in the tree which corresponds to S, or we have to introduce new labeled arcs (and nodes) into the tree until we have constructed a node representing S.

Table 4.1: List of faces $H(v)$ for the 4-cube given in Figure 4.2, where $H = \{5\}$ and $v \in \{0, \ldots, 15\} \setminus \{5\}$. See Example 4.4 and Figure 4.3.

v	$H(v)$	$C(H(v))$
0	$\{0, 1, 4, 5\}$	$\{0, 1, 4\}$
1	$\{1, 5\}$	$\{1, 5\}$
2	$\{0, 1, 2, 3, 4, 5, 6, 7\}$	$\{0, 1, 2, 4\}$
3	$\{1, 3, 5, 7\}$	$\{1, 3, 5\}$
4	$\{4, 5\}$	$\{4, 5\}$
6	$\{4, 5, 6, 7\}$	$\{4, 5, 6\}$
7	$\{5, 7\}$	$\{5, 7\}$
8	$\{0, 1, 4, 5, 8, 9, 12, 13\}$	$\{0, 1, 4, 8\}$
9	$\{1, 5, 9, 13\}$	$\{1, 5, 9\}$
10	$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$	$\{0, 1, 2, 4, 8\}$
11	$\{1, 3, 5, 7, 9, 11, 13, 15\}$	$\{1, 3, 5, 9\}$
12	$\{4, 5, 12, 13\}$	$\{4, 5, 12\}$
13	$\{5, 13\}$	$\{5, 13\}$
14	$\{4, 5, 6, 7, 12, 13, 14, 15\}$	$\{4, 5, 6, 12\}$
15	$\{5, 7, 13, 15\}$	$\{5, 7, 13\}$

In the latter case, it might be necessary to construct an entire new path in the tree. The definition of the canonical spanning sets $C(S)$ ensures that all "intermediate nodes" on that path will correspond to canonical spanning sets of faces as well. Hence, the number of nodes in the face tree always will be bounded by φ, the total number of faces of the polytope. The faces represented by intermediate nodes will be seen later in the algorithm. Consequently, the corresponding pointers to $\mathcal{L}$ are set to `nil` for the meantime. Later in the algorithm, when we are searching for the face represented by such a tree-node for the first time, the `nil`-pointer will indicate that this face is not yet represented in $\mathcal{L}$. The `nil`-pointer is then replaced by a pointer to a newly created node representing the face in $\mathcal{L}$.

In any case, since the face tree has depth at most $d+1$ and the out-degree of each node is at most n, we need a total time of $\mathcal{O}(n + \alpha + (d+1) \cdot n) = \mathcal{O}(\alpha)$ to either locate or create the tree-node representing a certain face.

Lemma 4.3. Using the face tree, it is possible to locate or create the node in $\mathcal{L}$ representing a face in $\mathcal{O}(\alpha)$ steps (provided the vertex-facet incidence matrix is stored in the sorted sparse format).

Example 4.4. Figure 4.2 shows a Schlegel diagram of the 4-cube and Figure 4.3 part of a face tree during a run of Algorithm 1.

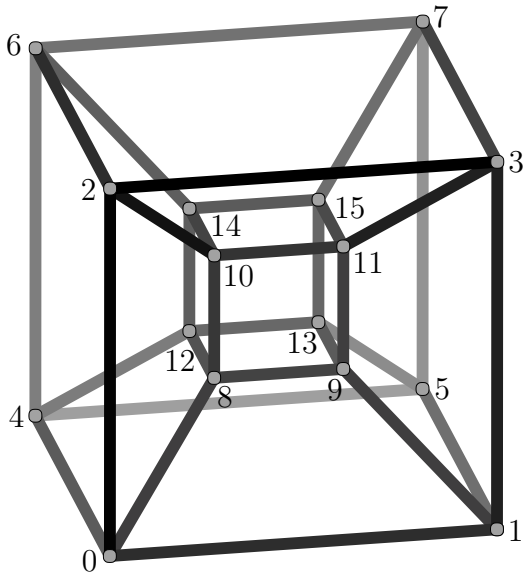


Figure 4.2: Schlegel diagram of the 4-cube used in Example 4.4.

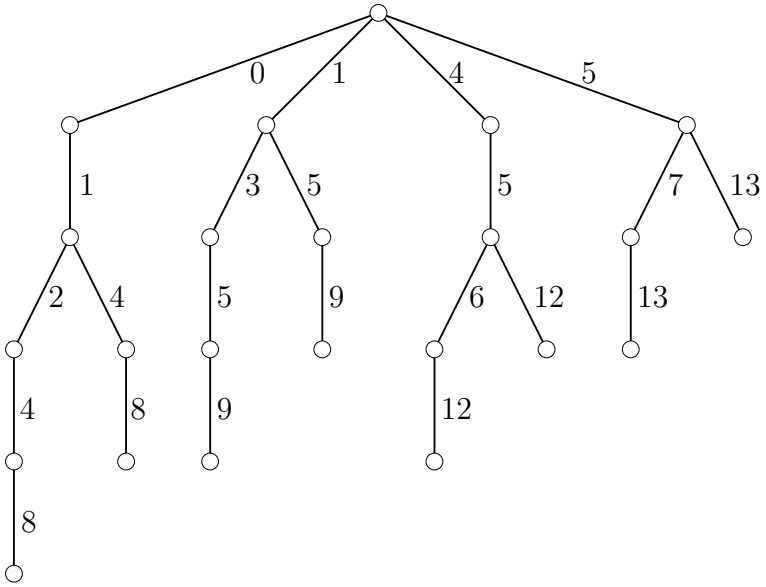


Figure 4.3: Illustration of the face tree during a run of Algorithm 1 for the face lattice of the 4-cube, see Figure 4.2. The requests to the face tree are the sets $C(H(v))$ as they appear in Table 4.1. See also the notes in Example 4.4.

As in every run of Algorithm 1, first the face $\varnothing$ is considered. The sets $H(v)$, for $H = \varnothing$ and $v \in V$, are exactly the vertices $0, 1, \ldots, 15$.

We assume that the next face is vertex $H := \{5\}$ and $H(v)$ is computed for all $v \in V \setminus \{5\}$. This results in 15 faces, which are shown in Table 4.1 together with the corresponding sets $C(H(v))$.

To illustrate the face tree, Figure 4.3 shows the result if all faces of Table 4.1 are requested in turn. Of course, in an actual execution of Algorithm 1 the face tree is larger (e.g., it includes nodes for all vertices).

In the description given above, we have assumed that for each node in the face tree the out-arcs are stored in a list which is searched linearly for a certain label when walking down the tree. Of course, one can store the set of out-arcs in a balanced search tree (see, e.g., [50, Chap. 13]), allowing to perform the search for a certain label in logarithmic time. After computing $C(S)$ for a face S (in $\mathcal{O}(\alpha)$ time), this allows to locate or create the node corresponding to S in the face tree in $\mathcal{O}((d+1) \cdot \log n)$ steps. The total running time remains $\mathcal{O}(\alpha)$; nevertheless this might speed up the algorithm in practice.

Instead of using the face tree, one can also store the faces in a balanced search tree. Again, the faces are represented by their canonical spanning sets, which are ordered lexicographically. Once $C(S)$ is computed for a face S, searching S can be performed in $\mathcal{O}((d+1) \cdot \log \varphi) \subseteq \mathcal{O}((d+1) \cdot \min\{n, m\})$ steps (since $\varphi \leq 2^{\min\{n,m\}}$). This yields the same total asymptotic running time, but searching the tree takes more (or the same) time compared to the variant of the face tree with balanced search trees at its nodes, since $\log n \leq \min\{n, m\}$.

4.2.5 The Analysis

We summarize the algorithm in pseudo-code (Algorithm 1) below.

Proposition 4.5. Algorithm 1 computes the Hasse diagram of the face lattice of a polytope P from its vertex-facet incidences in $\mathcal{O}(n \cdot \alpha \cdot \varphi)$ time. It can be implemented such that its space requirements (without output space) are bounded by $\mathcal{O}(\varphi \cdot n)$.

Proof. Algorithm 1 works correctly by the discussion above.

Step 7 can be performed in $\mathcal{O}(n \cdot \alpha)$ steps by Lemma 4.1. Lemma 4.2 shows that Step 8 can be executed in $\mathcal{O}(n^2) \subseteq \mathcal{O}(n \cdot \alpha)$ time. Hence, Steps 7 and 8 in total contribute at most $\mathcal{O}(n \cdot \alpha \cdot \varphi)$ to the running time (since the while-loop is executed once per face).

The body of the for-loop has to be executed for each of the $\mathcal{O}(n \cdot \varphi)$ arcs in the Hasse diagram $\mathcal{L}$. Lemma 4.3 implies that each execution of the body

Algorithm 1: Computing the face lattice of a polytope from its incidences.

Input: incidence matrix of a polytope P
Output: Hasse diagram $\mathcal{L}$ of the face lattice of P
initialize $\mathcal{L}$ and a face tree with $\ell_\varnothing$ corresponding to the empty face
initialize a set $\mathcal{Q} \subseteq \{\text{nodes of } \mathcal{L}\} \times \{\text{subsets of } V\}$ by $(\ell_\varnothing, \varnothing)$
while $\mathcal{Q} \neq \varnothing$ **do**
  choose some $(\ell_H, H) \in \mathcal{Q}$ and remove it from $\mathcal{Q}$
  compute the collection $\mathcal{H}$ of all $H(v)$, $v \in V \setminus H$
  compute the set $\mathcal{G}$ of minimal sets in $\mathcal{H}$
  for each $G \in \mathcal{G}$ **do**
    locate/create the node ℓ_G corresponding to G in $\mathcal{L}$
    if ℓ_G was newly created **then**
      add (ℓ_G, G) to $\mathcal{Q}$
    end if
    add the arc (ℓ_H, ℓ_G) to $\mathcal{L}$
  end for
end while

of the for-loop can be performed in $\mathcal{O}(\alpha)$ steps. Thus, the claim on the running time follows.

Since each node of the face tree corresponds to a face of P, the face tree has $\mathcal{O}(\varphi)$ nodes. Each label on an edge of the face tree needs at most $\mathcal{O}(\log n)$ bits, and we can estimate the space requirements of any of the (internal and external) pointers by $\mathcal{O}(\log \varphi) \subseteq \mathcal{O}(\min\{n, m\})$. Hence, the face tree needs no more than $\mathcal{O}(\varphi \cdot \min\{n, m\})$ bits.

The space required for the storage of $\mathcal{Q}$ is bounded by $\mathcal{O}(\varphi \cdot n)$, if for each pair $(\ell_H, H) \in \mathcal{Q}$ the set H is stored as a *bit set*, i.e., the characteristic vector of $H \subseteq V$ is stored bit by bit. □

If $m < n$, then it is more efficient to apply Algorithm 1 to the incidences of the dual polytope, i.e., to the transpose of the incidence matrix. Of course, after the computations the roles of vertices and facets have to be exchanged again. This yields the main result of this chapter.

Theorem 4.6. The Hasse diagram of the face lattice of a polytope P can be computed from the vertex-facet incidences of P in $\mathcal{O}(\min\{n, m\} \cdot \alpha \cdot \varphi)$ time, where n is the number of vertices, m is the number of facets, α is the number of vertex-facet incidences, and φ is the total number of faces of P. The space requirements of the algorithm (without output space) can be bounded by $\mathcal{O}(\varphi \cdot \min\{n, m\})$.

Whenever a new node representing a face G in the Hasse diagram $\mathcal{L}$ is constructed, we can label that node with the vertex set of G, the set of facets containing G, or with the dimension of G without (asymptotically) increasing the running time of the algorithm. The output, however, might become much larger with such labelings. For instance, labeling the Hasse diagram of the d-cube by vertex labels requires $\Omega(4^d \cdot d)$ output storage space, while the Hasse diagram with facet labels needs only $\mathcal{O}\big(d \cdot 3^d \cdot \log d\big)$ space.

In practice, the computation can be speeded up by exploiting that every vertex that is contained in a face G with $H \subset G$ and $\dim G = \dim H + 1$ must be contained in some facet which contains H. Thus, it suffices to consider only the sets $H(v)$, $v \in \big(\bigcup_{f \in \mathrm{F}(H)} \mathrm{V}(\{f\})\big) \setminus H$ in Step 7.

Remark 4.7. Algorithm 1 can be used to compute the Hasse diagram of an atomic lattice, if a subroutine is available that computes the join of a set of atoms. In Section 4.3.4 we provide an application of this idea to oriented matroids. Another example are atomic and coatomic lattices where the atom-coatom incidences are given; in this case one can compute the joins of atoms similarly to the case of face lattices of polytopes.

4.3 EXTENSIONS

4.3.1 Simple or Simplicial Polytopes

For a simple d-polytope P with n vertices, the above procedure can be implemented to run more efficiently. We have $\alpha = n \cdot d$ in this case. From the incidences (stored in the sorted sparse format), the graph $G(P)$ of P (i.e., all one-dimensional faces) can be computed in time $\mathcal{O}(n^2 \cdot d)$, since a pair of vertices forms an edge if and only if it is contained in $d-1$ common facets.

After initialization with the vertices instead of $\varnothing$ (in Steps 3 and 4), Steps 7 and 8 can now be simplified. Consider an arbitrary vertex $w \in H$. For each neighbor $v \notin H$ of w in $G(P)$, $H(v)$ yields the other end node of an arc in the Hasse diagram; and each out-arc of H is produced this way. Thus, we can avoid constructing non-minimal faces in Step 7. Hence, Step 8 can be skipped. The total running time for simple d-polytopes decreases to $\mathcal{O}(d \cdot \alpha \cdot \varphi)$ (since the body of the for-loop is executed at most $d \cdot \varphi$ times).

The space complexity stays $\mathcal{O}(\varphi \cdot n)$ (see Proposition 4.5). It can, however, be reduced to $\mathcal{O}(\varphi \cdot m)$ (we have $m \leq n$ for simple polytopes): instead of storing pairs (ℓ_H, H) in the set $\mathcal{Q}$, we store the pairs $(\ell_H, \mathrm{F}(H))$, since $|\mathrm{F}(H)| \leq m$. Converting between H and $\mathrm{F}(H)$ can be performed in $\mathcal{O}(\alpha)$ steps and hence does not increase the asymptotic total running time.

Using duality, the same running times and space requirements can be achieved for simplicial polytopes.

Similarly to the situation with general polytopes, for both simple and simplicial polytopes we can also output for each face its vertices, the facets containing it, or its dimension without (asymptotically) increasing the running time.

4.3.2 The k-Skeleton

A variant of Algorithm 1 computes the Hasse diagram of the k-skeleton (all faces of dimension at most k) of a polytope P. One simply prevents the computation of faces of dimensions larger than k by not inserting any $(k-1)$-face into the list $\mathcal{Q}$. This leads to an $\mathcal{O}\big(n \cdot \alpha \cdot \varphi^{\leq k}\big)$ time algorithm, where $\varphi^{\leq k}$ is the number of faces of P of dimension at most k. The space requirements are $\mathcal{O}\big(n \cdot \varphi^{\leq k}\big)$.

4.3.3 Computing the "Hasse Diagram without Edges"

If we only want to compute the faces of P together with their descriptions and dimensions (i.e., the "Hasse diagram without edges"), there exists a variant of Algorithm 1 with the same asymptotic running time, but significantly reduced space requirements. The difference is that no face tree is used, and the set $\mathcal{Q}$ is organized as a stack, i.e., the faces are investigated in a depth-first search manner. At each step, we take a face H from the stack, output it, and compute the set $\mathcal{G}$ of $(\dim H + 1)$-faces containing H, like in Steps 7 and 8 of Algorithm 1. This needs time $\mathcal{O}(n \cdot \alpha)$ for each H. The for-loop beginning at Step 10, including the search in the face tree, is replaced by an efficient way to decide which of the faces in $\mathcal{G}$ is put onto the stack $\mathcal{Q}$, such that every face appears exactly once on the stack during the algorithm. For this, we compute for each face $G \in \mathcal{G}$ a unique canonical facet H' of it. We put G onto the stack if and only if $H = H'$. This ensures that each face is picked exactly once.

We take H' as the closure of a set $D(G)$, which is computed similar to the set $C(G)$ of Section 4.2.4, except that we reject vertices which would produce G. More precisely, let $G = \{g_1, g_2, \ldots, g_l\}$, with $g_1 < g_2 < \cdots < g_l$. Initialize $D(G)$ with $\varnothing$ and in each iteration extend $D(G)$ by the smallest g_i such that $\mathrm{cl}(D(G)) \subset \mathrm{cl}(D(G) \cup \{g_i\})$ and $\mathrm{cl}(D(G) \cup \{g_i\}) \neq G$. After the computation, H', the closure of $D(G)$, clearly is a proper face of G. Moreover, it is a facet of G, since otherwise there exists a vertex $g \in G \setminus H'$, such that $\mathrm{cl}(H' \cup \{g\}) \subset G$. But then g would have been included into $D(G)$ when it was considered. Hence, $D(G)$ is the lexicographically smallest subset

of G which spans a facet of G. It can be computed in time $\mathcal{O}(\alpha)$, and hence checking for all faces $G \in \mathcal{G}$ whether H is the canonical facet $D(G)$ of G can be performed in $\mathcal{O}(n \cdot \alpha)$ time.

Altogether, this leads to an $\mathcal{O}(n \cdot \alpha \cdot \varphi)$ time algorithm (see the proof of Proposition 4.5). The algorithm needs $\mathcal{O}(n^2 \cdot d \cdot \log n)$ space for $\mathcal{Q}$; since the depth of $\mathcal{Q}$ is at most $d+1$, there are never more than $n \cdot (d+1)$ sets on the stack, each of size at most n. Additionally, we need $\mathcal{O}(\alpha \cdot \log \max\{n, m\})$ space for storing the incidences in the sorted sparse format. Applying this method to the dual polytope, if necessary, we obtain an $\mathcal{O}(\min\{n, m\} \cdot \alpha \cdot \varphi)$ time algorithm.

4.3.4 Oriented Matroids

As noted in Remark 4.7, Algorithm 1 can be used to compute the Hasse diagram of any atomic lattice provided a subroutine is available that computes the join of a set of atoms. In this section, we describe such an application of our algorithm to oriented matroids.

The covectors of an oriented matroid with groundset $\{1, \ldots, k\}$ are elements of $\{-, 0, +\}^k$ that satisfy certain axioms. We refer to the book of Björner, Las Vergnas, Sturmfels, White, and Ziegler [30, Chap. 4] for the definitions and concepts that are relevant in the following.

A specific, but illustrative, example of an oriented matroid arises from any finite set X of points in $\mathbb{R}^d$ as follows. For every linear functional $\varphi \in (\mathbb{R}^d)^\star$ denote by $\textsc{sign}(\varphi) \in \{-, 0, +\}^X$ the vector whose component corresponding to $x \in X$ encodes the sign of $\varphi(x)$. Then the set $\{\textsc{sign}(\varphi) : \varphi \in (\mathbb{R}^d)^\star\}$ is the set of covectors of an oriented matroid $\mathcal{O}(X)$.

For $\mathrm{V}, \mathrm{W} \in \{-, 0, +\}^k$ the *separation set* $\mathrm{S}(\mathrm{V}, \mathrm{W})$ of V and W contains all indices i such that one of V_i, W_i is $+$, and the other one is $-$. The *composition* $\mathrm{V} \circ \mathrm{W}$ of V and W is defined by $(\mathrm{V} \circ \mathrm{W})_i := \mathrm{V}_i$ if $\mathrm{V}_i \neq 0$ and $(\mathrm{V} \circ \mathrm{W})_i := \mathrm{W}_i$ otherwise.

We define a partial order $\preceq$ on $\{-, 0, +\}^k$, where $\mathrm{V} \preceq \mathrm{W}$ holds if and only if for all i we have $\mathrm{V}_i = 0$ or $\mathrm{V}_i = \mathrm{W}_i$. The $\preceq$-minimal elements among the nonzero covectors of an oriented matroid are called its *cocircuits*. If one adjoins an artificial maximal element $\hat{1}$ to the poset formed by the covectors of an oriented matroid (ordered by $\preceq$), then one obtains its (*big*) *face lattice*.

If, in the above example, X is the vertex set of a polytope $P \subset \mathbb{R}^d$, then the faces of P correspond to the *positive covectors* (i.e., the covectors with no component equal to $-$) of $\mathcal{O}(X)$. The facets of P correspond to the positive cocircuits of $\mathcal{O}(X)$. The face lattice of P is anti-isomorphic to a sublattice of the face lattice of $\mathcal{O}(X)$.

The face lattice of an oriented matroid is atomic and coatomic; its atoms are the cocircuits, and its coatoms are called *topes*. Hence, as in the case of polytopes, we can compute the (Hasse diagram of the) face lattice of an oriented matroid if the abstract atom-coatom incidences are given.

However, this is not the usual way to encode an oriented matroid. It is far more common to specify an oriented matroid by its cocircuits. In this case, we can use the fact that the join of two covectors simply is their composition, if their separation set is empty, or $\hat{1}$ otherwise. Such a composition can be computed in $\mathcal{O}(k)$ steps, which enables us to compute the (Hasse diagram of the) face lattice (efficiently) from its cocircuits by a variant of Algorithm 1, which we describe now.

In Step 6, H now is a face of the oriented matroid, i.e., a covector. In Step 7, one has to compute the joins of H with every cocircuit $\mathrm{V} \not\preceq H$. Thus, Step 7 can be performed in $\mathcal{O}(n \cdot k \cdot \varphi)$ steps altogether (where φ is the total number of covectors and n is the number of cocircuits). We do not know any method to perform Step 8 faster than by pairwise comparisons, which take $\mathcal{O}(n^2 \cdot k \cdot \varphi)$ time in total.

The face tree is organized similarly to the description in Section 4.2.4. One first fixes an (arbitrary) ordering $\mathrm{C}_1, \ldots, \mathrm{C}_n$ of the cocircuits. Then for a covector S let $\{i_1, \ldots, i_r\}$ $(i_1 < \cdots < i_r)$ be the set of indices of cocircuits $\mathrm{C}_{i_j} \preceq \mathrm{S}$. To compute a "canonical spanning set", we iteratively form the joins of $\mathrm{C}_{i_1}, \ldots, \mathrm{C}_{i_r}$, and let $C(\mathrm{S})$ consist of all those indices for which the "joins change". Computing $C(\mathrm{S})$ from S takes $\mathcal{O}(n \cdot k)$ steps. Note that $|C(\mathrm{S})| \leq k$.

Using this modified face tree, the node corresponding to a given covector S can now be located or created in the same way as in the case of face lattices of polytopes. The depth of the face tree is bounded by k. Hence, location/creation of a covector can be performed in $\mathcal{O}(n \cdot k)$ time. The rest of the analysis is similar to the proof of Proposition 4.5. We conclude that by this variant of Algorithm 1, the Hasse diagram of the face lattice of an oriented matroid can be computed in $\mathcal{O}(n^2 \cdot k \cdot \varphi)$ steps, requiring $\mathcal{O}(\varphi \cdot k)$ working space (since $\varphi \leq 3^k$).

Finschi [56] describes a different algorithm that computes the covectors of an oriented matroid from its cocircuits in $\mathcal{O}(n \cdot k^2 \cdot \varphi)$ time. His algorithm, however, does not produce the edges of the Hasse diagram.

The case where the topes (i.e., the $\preceq$-maximal covectors) of an oriented matroid are given is a bit different. Here, the number of faces is bounded by m^2, where m is the number of topes. Hence, the size of the face lattice is polynomial in m. Fukuda, Saito, and Tamura [60] give an $\mathcal{O}(k^3 \cdot m^2)$ time algorithm for constructing the face lattice from the maximal covectors.

4.3.5 Computational Experience

In this section we report on computational experience with Algorithm 1. The goal is to provide an idea of the running times of this algorithm for different classes of polytopes. No detailed investigation is intended here.

Algorithm 1 is compared to a naive algorithm to compute the face lattice of a polytope (given the vertex-facet incidences), which we call the *level algorithm*. This algorithm proceeds by levels in the face lattice, starting at the level of the facets. As before, we identify a face with its vertex set. At each step, every element of a level (face) is intersected with every facet. The inclusionwise maximal sets among the resulting sets constitute the faces of the level below the current one. This algorithm needs $\mathcal{O}(d \cdot n \cdot \varphi^2)$ time and $\mathcal{O}(\varphi \cdot n)$ space in the worst case to compute the face lattice. It can be easily modified to produce the Hasse diagram, too.

Below, we compare the running times of implementations of the level algorithm and of Algorithm 1. Both algorithms were implemented and tested by Nikolaus Witte using the `polymake` framework of Gawrilow and Joswig [63, 64]. For this comparison, both algorithms just produce the face lattice (without the edges of the Hasse diagram). Algorithm 1 issues requests to the face tree data structure to test if a face has already been output, but does not create the edges of the Hasse diagram. The specializations for simple and simplicial polytopes described in Section 4.3.1 are not implemented, although all tested polytopes below are simplicial.

Table 4.2 gives the results for the d-dimensional simplex, for $4 \leq d \leq 13$. Indicated are the number of vertices n, the number of facets m, the number

Table 4.2: Running times in seconds for the level algorithm (T_{la}) and Algorithm 1 (T_{A1}) for the d-simplex.

d	n	m	α	φ	T_{la}	T_{A1}	$T_{A1}/(n \cdot \alpha \cdot \varphi)$
4	5	5	20	31	0.00	0.00	0.00
5	6	6	30	63	0.00	0.00	0.00
6	7	7	42	127	0.01	0.04	$1.07 \cdot 10^{-6}$
7	8	8	56	255	0.02	0.06	$5.25 \cdot 10^{-7}$
8	9	9	72	511	0.06	0.15	$4.53 \cdot 10^{-7}$
9	10	10	90	1023	0.27	0.38	$4.13 \cdot 10^{-7}$
10	11	11	110	2047	1.20	0.91	$3.67 \cdot 10^{-7}$
11	12	12	132	4095	4.89	2.27	$3.50 \cdot 10^{-7}$
12	13	13	156	8191	20.32	5.31	$3.20 \cdot 10^{-7}$
13	14	14	182	16383	85.04	12.64	$3.03 \cdot 10^{-7}$

Table 4.3: Running times in seconds for the level algorithm (T_{la}) and Algorithm 1 (T_{A1}) for the d-cross polytope.

d	n	m	α	φ	T_{la}	T_{A1}	$T_{A1}/(n \cdot \alpha \cdot \varphi)$
4	8	16	64	81	0.01	0.01	$2.41 \cdot 10^{-7}$
5	10	32	160	243	0.02	0.06	$1.54 \cdot 10^{-7}$
6	12	64	384	729	0.17	0.43	$1.28 \cdot 10^{-7}$
7	14	128	896	2187	1.67	2.55	$9.30 \cdot 10^{-8}$
8	16	256	2048	6561	17.56	17.17	$8.00 \cdot 10^{-8}$
9	18	512	4608	19683	213.73	119.56	$7.32 \cdot 10^{-8}$
10	20	1024	10240	59049	2844.25	821.09	$7.00 \cdot 10^{-8}$
11	22	2048	22528	177147	37601.80	5701.34	$6.49 \cdot 10^{-8}$

of vertex-facet incidences α, the size of the face lattice φ, the running time T_{la} of the level algorithm in seconds, the running time T_{A1} of Algorithm 1 in seconds, and the quotient $T_{A1}/(n \cdot \alpha \cdot \varphi)$. Similarly, Table 4.3 provides results for the d-dimensional cross polytope, for $4 \leq d \leq 11$. Table 4.4 gives results for random d-polytopes, which are produced by placing n vertices randomly on the $(d-1)$-sphere. Such polytopes are generated for $d = 4, 5, 6$ and $n = 10, 20, \ldots, 100$. All values in Table 4.4 are mean values over 10 runs.

The data of Table 4.2 and 4.3 show the rapid growth of φ and the running times with increasing dimension. In Table 4.4, the dimensions are small, but the growth of φ and the running times for an increasing number of vertices already becomes visible. In any case, the data shows the limitations on the practical applicability of both the level algorithm and of Algorithm 1; for larger dimensions and higher number of vertices the computation times and space requirements become too large.

For small d and n, the running times of Algorithm 1 are higher than the times of the level algorithm. This could be caused by the overhead of the data structures. For larger d, i.e., $d \geq 10$ for the simplex and cross polytope, Algorithm 1 is clearly superior to the level algorithm (as the theoretical results indicate). For random polytopes, Algorithm 1 is better if the number of vertices (and facets) is large. Not surprisingly, the higher the dimension, the better Algorithm 1 is compared to the level algorithm.

Furthermore, the quotient of the running time T_{A1} of Algorithm 1 and the number $n \cdot \alpha \cdot \varphi$ is surprisingly stable. This indicates that the asymptotic theoretical running times are already a good estimate of the asymptotic running times in practice, even for small values of n, α, and φ.

Table 4.4: Running times in seconds of the level algorithm (T_{la}) and Algorithm 1 (T_{A1}) for random d-polytopes for $d = 4, 5, 6$. *All results are mean values over 10 trials.*

d	n	m	α	φ	T_{la}	T_{A1}	$T_{A1}/(n \cdot \alpha \cdot \varphi)$
4	10	26	104	125	0.004	0.018	$1.38 \cdot 10^{-7}$
	20	79	357	316	0.038	0.088	$3.90 \cdot 10^{-8}$
	30	136	544	605	0.113	0.177	$1.79 \cdot 10^{-8}$
	40	196	785	866	0.243	0.302	$1.11 \cdot 10^{-8}$
	50	259	1034	1135	0.405	0.448	$7.63 \cdot 10^{-9}$
	60	323	1292	1413	0.634	0.638	$5.82 \cdot 10^{-9}$
	70	385	1538	1679	0.902	0.867	$4.79 \cdot 10^{-9}$
	80	443	1771	1932	1.193	1.147	$4.19 \cdot 10^{-9}$
	90	512	2048	2229	1.619	1.446	$3.52 \cdot 10^{-9}$
	100	573	2291	2492	2.008	1.803	$3.16 \cdot 10^{-9}$
5	10	36	180	267	0.022	0.079	$1.64 \cdot 10^{-7}$
	20	175	873	1159	0.435	0.74	$3.66 \cdot 10^{-8}$
	30	349	1745	2265	1.671	1.814	$1.53 \cdot 10^{-8}$
	40	549	2743	3523	4.083	3.407	$8.82 \cdot 10^{-9}$
	50	766	3828	4885	7.791	5.389	$5.76 \cdot 10^{-9}$
	60	982	4910	6243	12.665	7.107	$3.86 \cdot 10^{-9}$
	70	1219	6093	7723	19.376	9.388	$2.85 \cdot 10^{-9}$
	80	1448	7241	9160	27.211	11.768	$2.22 \cdot 10^{-9}$
	90	1691	8454	10676	36.557	14.437	$1.78 \cdot 10^{-9}$
	100	1941	9704	12236	47.930	17.165	$1.44 \cdot 10^{-9}$
6	10	40	241	479	0.064	0.214	$1.85 \cdot 10^{-7}$
	20	351	2106	3439	3.902	22.104	$3.89 \cdot 10^{-8}$
	30	873	5240	8277	23.104	22.104	$1.70 \cdot 10^{-8}$
	40	1508	9047	14105	69.994	49.053	$9.61 \cdot 10^{-9}$
	50	2266	13593	21020	169.801	87.838	$6.15 \cdot 10^{-9}$
	60	3064	18382	28301	320.993	134.079	$4.30 \cdot 10^{-9}$
	70	3954	23722	36377	634.941	193.344	$3.20 \cdot 10^{-9}$
	80	4871	29226	44704	987.710	257.754	$2.47 \cdot 10^{-9}$
	90	5716	34298	52413	1303.643	319.822	$1.98 \cdot 10^{-9}$
	100	6804	40823	62235	1923.358	402.331	$1.58 \cdot 10^{-9}$

Chapter 5

Branch-and-Cut for the Min IIS Cover Problem

In this chapter we report on experience with a preliminary implementation of a branch-and-cut algorithm for the Min IIS Cover Problem. It relies on a generalization of the integer programming formulation (1.1) of the Min IIS Cover problem, given in Section 1.1.1. We refer to this section for the basic definitions needed in the sequel.

This chapter is organized as follows. In Section 5.1, we give an overview of the branch-and-cut approach for Min IIS Cover and discuss basic methods needed in the rest of the chapter. We explain the usage of a generalized alternative polyhedron to check whether a given set of inequalities corresponds to an IIS-cover, give heuristics to separate IIS-inequalities, and introduce primal heuristics. We then briefly discuss two types of additional cutting planes (Section 5.2). One type comprises a special case of inequalities characterized by Balas and Ng. The second type consists of Gomory cuts. In Section 5.3, we report on the performance of the branch-and-cut implementation on three different sets of test problems. The first set contains infeasible systems collected in the Netlib library. The second set includes random inequality systems and the third involves classification problems in machine learning.

We assume that the reader is familiar with branch-and-cut algorithms and the basic concepts of integer programming. We refer to Nemhauser and Wolsey [90], Padberg and Rinaldi [92], Thienel [112], and Caprara and Fischetti [39] for more information.

5.1 Basic Techniques

The outline of the branch-and-cut implementation is as follows. We allow general infeasible instances of Min IIS Cover of the following form:

$$\Sigma : \{C\boldsymbol{x} \leq \boldsymbol{c},\ D\boldsymbol{x} = \boldsymbol{d},\ \boldsymbol{\ell} \leq \boldsymbol{x} \leq \boldsymbol{u}\}, \tag{5.1}$$

where $C \in \mathbb{R}^{m_1 \times n}$, $D \in \mathbb{R}^{m_2 \times n}$, $\boldsymbol{c} \in \mathbb{R}^{m_1}$, $\boldsymbol{d} \in \mathbb{R}^{m_2}$, and $\boldsymbol{\ell}$, $\boldsymbol{u} \in \mathbb{R}^n$. We define $m := m_1 + m_2 + 2n$, so that Σ has m constraints. Furthermore, let

$S(\Sigma) := [m_1]\times[m_2]\times[n]\times[n]$; hence we have $|S(\Sigma)| = 2^{m_1}\cdot 2^{m_2}\cdot 2^n\cdot 2^n = 2^m$. A subset $S = (S_1,\ S_2,\ S_3,\ S_4) \subseteq S(\Sigma)$ indexes a subset of the constraints of Σ, where S_1 and S_2 correspond to subsets of the rows of C and D, respectively. The sets S_3 and S_4 correspond to subsets of the lower $\boldsymbol{\ell}$ and upper bounds $\boldsymbol{u}$, respectively. Sometimes we identify $S(\Sigma)$ with $[m]$ and implicitly use the obvious translation between subsets of the two representations.

As an instance of MIN IIS COVER, Σ is infeasible and we want to delete as few of its m constraints as possible to obtain a feasible subsystem. The definition of an *Irreducible Inconsistent Subsystem* (IIS) directly carries over to this general case, i.e, an infeasible subsystem Σ' of Σ is an IIS if and only if all proper subsystems of Σ' are feasible. We usually identify the constraints with their indices and also call the index set of an IIS an IIS. A set of constraints that contains at least one constraint of each IIS is called an *IIS-cover*. To obtain a feasible solution, we have to delete at least one constraint out of each IIS, i.e., remove an IIS-cover from Σ. This observation leads to the following set covering formulation similar to (1.1).

To avoid the generation of all, or almost all, IISs we only work with a partial list of IISs. We introduce a variable y_i for the ith constraint of Σ $(i = 1, 2, \ldots, m)$ and consider the following partial set covering problem:

$$\begin{array}{lll} \min & \sum_{i=1}^{m} y_i & \\ \text{s.t.} & \sum_{i\in C} y_i \geq 1 & \text{for all } C \in \mathcal{C}' \\ & \boldsymbol{y} \in \{0,1\}^m, & \end{array} \tag{5.2}$$

where $\mathcal{C}'$ is a subset of the set of IISs $\mathcal{C}(\Sigma)$. Weighted versions of this formulation can be solved by the branch-and-cut implementation with only slight changes.

Since $\mathcal{C}'$ might not be complete, i.e., $\mathcal{C}' \neq \mathcal{C}(\Sigma)$, we have to be able to deal with the following situation: Given an integer solution of (5.2), check whether it is the incidence vector of an IIS-cover. If this is not the case, we should produce an IIS which is not covered. A method that can perform this task is explained in Section 5.1.1. We then can add the resulting IIS to $\mathcal{C}'$ and iterate.

Parker and Ryan [95] solve (5.2) by exact methods, check whether the solution corresponds to an IIS-cover, and iterate (if necessary) until a minimum cardinality IIS-cover is found. They also discuss the use of heuristics to solve the integer program (5.2), where of course the last solution has to be computed exactly. For both approaches, however, it can happen that the number of IISs that have to be covered gets very large and consequently the corresponding set covering problem is very hard to solve. Hence, the idea evolved to incorporate the solution of the set covering problems into a

branch-and-cut algorithm as follows. In fact, Parker [94] began the investigation of polyhedral properties of the corresponding polytope P_{IISC}, see also Section 1.3.2.

We start with the empty set $\mathcal{C}'$ and then iteratively solve the LP relaxation of (5.2). Additionally, we add cutting planes after each iteration and perform a branch-and-bound process. If we obtain a 0/1-solution, we check if it is an IIS-cover as mentioned above (feasibility test). So far, we use the following cutting planes in our implementation: inequalities arising from IISs as in (5.2) (see Section 1.3.2), special cases of inequalities characterized by Balas and Ng (Section 5.2.1), and Gomory cuts (Section 5.2.2). All cuts are stored in a pool, and at the beginning of each iteration the pools are checked for violated cuts, which are then included into the current LP relaxation.

The fact that we start with the empty set $\mathcal{C}'$ is not a problem, since testing if the empty set is an IIS-cover, i.e., the whole system is feasible, can be done by linear programming. If it is not an IIS-cover, we can obtain an IIS that is not covered by the method mentioned above.

For $S \subseteq [m]$ and a vector $\boldsymbol{x} \in \mathbb{R}^n$, in the following we write

$$x(S) = \sum_{i \in S} x_i \,.$$

We now explain the above mentioned methods in more detail.

5.1.1 Checking for an IIS-Cover

We consider the following basic problem: Given an integer solution of (5.2), check whether it is the incidence vector of an IIS-cover. If it does not correspond to an IIS-cover, we have to produce a witness, i.e., an IIS which is not covered.

We can use the statement of Theorem 1.13 to perform this task: IISs of Σ are in one-to-one correspondence with supports of the vertices of the *alternative polyhedron*

$$\begin{aligned} P(\Sigma) = \{(\boldsymbol{y}, \boldsymbol{v}, \boldsymbol{w}, \boldsymbol{z}) \,:\; & \boldsymbol{y}C + \boldsymbol{v}D + \boldsymbol{w} - \boldsymbol{z} = \boldsymbol{0} \\ & \boldsymbol{y}\boldsymbol{c} + \boldsymbol{v}\boldsymbol{d} + \boldsymbol{w}\boldsymbol{u} - \boldsymbol{z}\boldsymbol{\ell} = -1 \\ & \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z} \geq \boldsymbol{0},\ \boldsymbol{v} \text{ free}\}. \end{aligned} \tag{5.3}$$

For $S = (S_1,\ S_2,\ S_3,\ S_4) \subseteq S(\Sigma)$, define the polyhedron

$$\begin{aligned} P_S(\Sigma) := \{(\boldsymbol{y}, \boldsymbol{v}, \boldsymbol{w}, \boldsymbol{z}) \in P(\Sigma) \,:\; & y_i = 0 \quad i \in S_1, \qquad v_i = 0 \quad i \in S_2, \\ & w_i = 0 \quad i \in S_3, \qquad z_i = 0 \quad i \in S_4\ \}, \end{aligned}$$

which might be empty. We use the following useful fact:

Lemma 5.1. The set $S \subseteq S(\Sigma)$ corresponds to an IIS-cover if and only if $P_S(\Sigma) = \varnothing$.

Proof. The system 5.3 in which all variables indexed by S are removed has no solution if and only if $P_S(\Sigma) = \varnothing$. By the Farkas lemma (Proposition 1.3), the former is the case if and only if the system (5.1) with inequalities indexed by S removed is feasible, i.e., S is an IIS-cover. □

If $\boldsymbol{y} \in \{0,1\}^m$ is a feasible solution to (5.2), we let $S := \{i \in [m] : y_i = 1\}$. Then we test if $P_S(\Sigma) = \varnothing$ by an LP-solver that returns a vertex in the case where $P_S(\Sigma) \neq \varnothing$. By Lemma 5.1, S is an IIS-cover if $P_S(\Sigma) = \varnothing$. Otherwise, let $\boldsymbol{v}$ be a vertex of $P_S(\Sigma)$. We clearly have $\operatorname{supp} \boldsymbol{v} \cap S = \varnothing$. Then by Theorem 1.13, $\operatorname{supp} \boldsymbol{v}$ corresponds to an IIS that has empty intersection with S, i.e., is uncovered.

5.1.2 Preprocessing

We perform a preprocessing step to find "small" IISs before starting the branch-and-cut algorithm. Such IISs are of interest since their corresponding IIS-inequalities provide the "strongest" cuts. Furthermore, small IISs are hard to find by other methods, e.g., by the ones explained in the next section.

In a first step, one can check if there are IISs of cardinality one, although they rarely occur in practice. In principle, these IISs can be removed, since any IIS-cover includes these inequalities. In a second step, we look for IISs of cardinality two. Such IISs are easy to find by just comparing their normal vectors and right hand sides. We furthermore check for IISs that involve one inequality and bounds on the variables. To find more complicated IISs would need more elaborate methods and would be too time consuming.

5.1.3 Separation of IIS Facets

Clearly, the most important cuts one can use in a branch-and-cut algorithm for the MIN IIS COVER problem are IIS-inequalities:

$$y(C) = \sum_{i \in C} y_i \geq 1,$$

for an IIS C. If Σ is of the form $\{A\boldsymbol{x} \leq \boldsymbol{b}\}$ and $|C| > 1$, applying the connection between P_{IS} and P_{FS} (see Section 1.3.1), Theorem 1.26 shows that these IIS-inequalities define facets of the polytope

$$P_{IISC} = \{\boldsymbol{y} \in \{0,1\}^m : y(C) \geq 1 \text{ for all IISs } C\}.$$

This result is not necessarily true for our case where Σ contains equations, i.e., $m_2 > 0$; in this case, IISs are not necessarily closed, see Example 1.28. Nevertheless, IIS-inequalities are still important valid inequalities.

As mentioned above, after the solution of each LP relaxation we want to find IIS-inequalities that violated by the current LP solution. By Proposition 1.29, finding such IIS-inequalities (or deciding that there exists none) is $\mathcal{NP}$-hard. Hence, it is unlikely that there exists an efficient algorithm to solve this problem. Consequently, all of the (polynomial-time) methods to find violated IIS-inequalities introduced in this section can fail to produce any such inequality, even if many exist.

The key to find IISs is the alternative polyhedron $P(\Sigma)$. As the IISs correspond to the vertices of $P(\Sigma)$, it suffices to work on this polyhedron in order to find new IISs. Let $\boldsymbol{x}^* \in \mathbb{R}^m$ be the solution of the current LP relaxation in the branch-and-cut algorithm. Hence, the problem to find an IIS-inequality violated by $\boldsymbol{x}^*$ can in principle be solved by first determining

$$\lambda := \min\{x^*(S) \,:\, S = \operatorname{supp}(\boldsymbol{v}),\ \boldsymbol{v} \text{ vertex of } P(\Sigma)\}. \tag{5.4}$$

If $\lambda < 1$, the support of a vertex at which the minimum is attained corresponds to an IIS C whose corresponding inequality $x^*(C) \geq 1$ is violated. As noted above, the problem to find λ is $\mathcal{NP}$-hard. Hence, we introduce the following heuristics for it.

Method 1: The first intuitive way to separate an IIS-inequality is to approximate (5.4) by the following LP:

$$\min\{\boldsymbol{x}^*\boldsymbol{y} \,:\, \boldsymbol{y} \in P(\Sigma)\},$$

where again $\boldsymbol{x}^*$ is the current LP solution of the branch-and-cut algorithm. This approach finds only one IIS-inequality per iteration, which is not necessarily violated by $\boldsymbol{x}^*$, even if there exist many violated IIS-inequalities.

We also experimented with solving this LP by a simplex solver and then testing whether the support of each vertex on the path to the optimum is an IIS whose inequality is violated. This works, but is inferior to the methods presented below.

Method 2: An extension of the idea of method (1) is the following. Let S be the support of $\boldsymbol{x}^*$. Applying Lemma 5.1, we can check if S is an IIS-cover by finding a vertex solution of

$$\min\{\boldsymbol{x}^*\boldsymbol{y} \,:\, \boldsymbol{y} \in P_S(\Sigma)\},$$

if there exists one. If the LP is feasible, the result gives us a vertex which corresponds to an IIS, otherwise we found an IIS-cover, which may be a good primal solution to the Min IIS Cover problem.

Let $\mathcal{C}(S)$ be the set of all IISs which are covered by S, i.e., we have $C \cap S \neq \varnothing$ for each $C \in \mathcal{C}(S)$. By definition, the polyhedron $P_S(\Sigma)$ only has vertices whose components corresponding to S are zero. This excludes all IISs in $\mathcal{C}(S)$, since each IIS of Σ corresponds to the support of a vertex of $P(\Sigma)$. Hence, if we obtain a vertex solution of the above problem, then the corresponding IIS is not contained in $\mathcal{C}(S)$.

Usually, the current LP relaxation includes all known IIS-inequalities that are satisfied with equality or were violated by the solution of the previous LP relaxation (e.g., if they were separated from a pool). Clearly, $\boldsymbol{x}^*$ satisfies $x^*(C) \geq 1$ for each IIS C whose inequality is included the LP relaxation. Hence, for each such IIS C there exists $i \in C$ such that $x_i^* > 0$. In particular, we have $C \in \mathcal{C}(S)$.

It follows that the IIS found by the above process is new, i.e., not included in the current LP relaxation. Moreover, the corresponding IIS-inequality is clearly violated by $\boldsymbol{x}^*$.

If $\operatorname{supp}(\boldsymbol{x}^*)$ is not an IIS-cover, e.g., if during the above process we found a violated IIS-inequality, we can even start a greedy heuristic to produce an IIS-cover. Initially, let S be the support of $\boldsymbol{x}^*$. At each step, check if S is an IIS-cover using Lemma 5.1. If S is an IIS-cover, we stop. Otherwise, we found a new IIS C. In this case, we enlarge S greedily by an arbitrary element of C and iterate. This procedure yields an IIS-cover at the end and new IISs in each step (except in the last). In our implementation, we choose an element of C by taking an element that is contained in a maximal number of the IISs we have found so far. This is essentially a heuristic of Ryan [103].

Performing such a greedy extension is the default method for separating IIS-inequalities in the implementation of the branch-and-cut implementation. For instance, it is used in all computational experiments of Section 5.3.

Method 3: We can extend this idea further. We fix $\alpha \in [0, 1]$ and initially let $S = S(\alpha) := \{i : x_i^* \geq \alpha\}$. Then we can proceed as in method (2). At each step we either obtain an IIS-cover or an IIS which is not covered by $S(\alpha)$. In contrast to the above case, these IISs are not necessarily new and their corresponding inequalities may not be violated by $\boldsymbol{x}^*$.

The motivation for this approach is based on the following observation: It often happens that during the branch-and-cut process the fractional LP solutions are "smeared" over a large number of variables, i.e., many variables have about the same small nonzero value. In this case, method (2) is likely to stop in the first iteration, since the set of fractional variables is so large that it constitutes an IIS-cover. If α is large enough and $\boldsymbol{x}^*$ is not an integer solution, the set $S(\alpha)$ from above is not an IIS-cover. Consequently, some IIS-inequalities are found – although they might not be violated.

As a side-effect, when performing these methods, we obtain a rather big list of IISs at the end of the execution of the algorithm. Of course, we want to keep this list as small as possible. The experiments of Section 5.3.3 indicate that this is achieved by our branch-and-cut implementation and that there are far too many IISs to be enumerated completely, cf. Table 5.4. Nevertheless, the IISs that are found during the execution may provide additional information about the instance, e.g., in the application of Min IIS Cover to linear programming, see Section 1.1.2.

5.1.4 Primal Heuristics

Since the methods explained in the previous section generate (many) IIS-covers, we implemented a primal heuristic that improves these IIS-covers greedily. Let S be an IIS-cover to be improved. Initially set $S' := S$. To guide the search for a good primal solution, we sort the elements of S' according to increasing fractional value of the components of the current LP solution $\boldsymbol{x}^*$. We then remove each element in this order from S' and check if the result is an IIS-cover, using Lemma 5.1. If S' is not an IIS-cover, we reinclude the element otherwise we consider the next element. At the end of this process we obtain a possibly smaller IIS-cover S'.

To limit the amount of work spent for this heuristic, we apply it at the end of every kth branch-and-bound node, where k is a user defined number. In the computational experiments given below, k usually was set to 50. Additionally, we apply it once we reach a level for the first time; this helps to find better primal solutions at the beginning.

We could also apply any primal heuristic at the beginning of the whole branch-and-cut process. We experimented with the solution of an elastic LP, as explained in Section 1.1.3. Then one can take the set of slack variables with non-zero support as the initial set S and then improve this solution as above. Since the results of this approach are inferior to the results obtained by improvement during the run of the branch-and-cut code, we decided not to use it.

5.2 Facet-defining Inequalities

Many facet-defining inequalities for the set covering polytope P_{SC} and the independence system polytope P_{IS} are known. See the overview chapter of Ceria, Nobili, and Sassano [40] for a list of references about polyhedral results for P_{SC}. As a special case we have the vertex cover polytope P_{VC} or stable set polytope P_{ST} belonging to a graph (compare Section 1.3.1). The

understanding of the structure of these polytopes is (maybe not surprisingly) better than that of P_{SC} and P_{IS}. Moreover, hardly any facet-defining inequalities are known for P_{SC} and P_{IS} that can be separated in polynomial time; this is in contrast to the situation for P_{VC} and P_{ST}. To our knowledge, the only special inequalities for P_{SC} and P_{IS} that can be (heuristically) separated in polynomial time are the *conditional cuts* of Balas [16] and Balas and Ho [19], the *aggregated cuts* of Borndörfer [35], and the *k-projection* cuts of Nobili and Sassano [91]. Additionally, we have cutting planes that are generally applicable like Gomory cuts (see, e.g., Schrijver [107] and Balas, Ceria, Cornuéjols, and Natraj [18]) and lift-and-project cutting planes (see Balas, Ceria, and Cornuéjols [17]).

Up to writing of this thesis, we could not find any problem specific cuts for the MIN IIS COVER problem. Therefore, we decided on the following list of cutting planes to be used in the implementation of the branch-and-cut algorithm:

- IIS-inequalities (separation methods (2) and (3) of Section 5.1.3)
- Special cuts of Balas and Ng, explained in Section 5.2.1
- Gomory cut (see Section 5.2.2)

We also experimented with the aggregated cuts of Borndörfer [35]. Unfortunately, on our test problems this approach almost never found a violated inequality, so we don't use these cuts in the implementation and skip the discussion of this very interesting approach.

Of course, the limited number of cutting planes that we use in the implementation raises the question of how well a branch-and-cut code can perform beyond the computational results presented below, i.e., can additional cutting planes be found that help to significantly improve the performance of the branch-and-cut implementation. This question has to be left open for future research.

5.2.1 Balas and Ng Cuts

A class of inequalities we use in the branch-and-cut algorithm are of the form $\boldsymbol{a}\boldsymbol{y} \geq 2$, where $\boldsymbol{a} \in \{0,1,2\}^m$. Balas and Ng [20] consider these inequalities and characterize the cases when they define facets of the set covering polytope.

In this section, let $P_{SC}(M) = \{\boldsymbol{y} \in \mathbb{R}^m : M\boldsymbol{y} \geq \mathbb{1}\}$ be the set covering polytope, where $M = (m_{ij}) \in \{0,1\}^{k \times m}$. Assume $\boldsymbol{a}\boldsymbol{y} \geq \beta$, with $\boldsymbol{a} \in \mathbb{Z}^m$ and $\beta \in \mathbb{Z}$, defines a facet of $P_{SC}(M)$. It is well known that if $\beta > 0$ then $\boldsymbol{a} \geq \boldsymbol{0}$, and if $\beta = 1$, then $\boldsymbol{a}$ is a row of M (see [20]).

If $\beta = 2$, Balas and Ng show that $\boldsymbol{a}\boldsymbol{y} \geq \beta$ can be obtained in the following way. Let $S \subseteq [k]$ and define $\boldsymbol{a}^S$ as follows:

$$a_j^S = \begin{cases} 0 & \text{if } m_{ij} = 0 \text{ for all } i \in S, \\ 2 & \text{if } m_{ij} = 1 \text{ for all } i \in S, \\ 1 & \text{otherwise} \end{cases} \qquad \text{for } j = 1, \ldots, m.$$

Hence, the result of Balas and Ng is: If $\boldsymbol{a}\boldsymbol{y} \geq 2$ with $\boldsymbol{a} \in \mathbb{Z}^m$ defines a facet of P_{SC}, there exists a set $S \subseteq [k]$ such that $\boldsymbol{a} = \boldsymbol{a}^S$. They also show that this class of inequalities is in the elementary closure of P_{SC}. Furthermore, they discuss conditions on S, such that $\boldsymbol{a}^S\boldsymbol{y} \geq 2$ defines a facet of P_{SC}.

The complexity status of the corresponding separation problem is unknown, but it seems difficult to find a violated inequality of this type. In fact, we tried to separate such cuts by randomly picking S and then greedily trying to improve the violation of $\boldsymbol{a}^S\boldsymbol{x}^* \geq 2$, where $\boldsymbol{x}^*$ denotes the current solution of the LP relaxation. This approach was almost always unsuccessful. Therefore, in our implementation we enumerate sets $S \subseteq [k]$ of *cardinality three* and check if the inequalities $\boldsymbol{a}^S\boldsymbol{y} \geq 2$ are violated by the current LP solution. Note that the sets $S \subseteq [k]$ of cardinality two are uninteresting, since in this case $\boldsymbol{a}^S\boldsymbol{x} \geq 2$ is just the sum of two IIS-inequalities and hence is never violated if the IIS-inequalities are not violated.

To keep computation times of our approach small, we take the currently active IIS-inequalities as the rows of the matrix M and consider a randomly chosen subset of these as candidates for elements of S. The size of this subset is user defined. For the results presented in Section 5.3, it is 200.

It turns out that this approach yields a successful method in practice.

5.2.2 Gomory Cuts

Gomory cutting planes are a general class of cutting planes that can be separated in polynomial time and always yield a violated inequality (if the solution is not integer), see, e.g., Nemhauser and Wolsey [90] and Schrijver [107]. One can prove finite termination of a cutting plane algorithm that uses Gomory cut, but they were rarely used since they were introduced by Gomory in the 1950's. The main reason is that the precision needed for the coefficients of the generated inequalities can get large after a couple of iterated applications of the cutting plane generation. Lately, Gomory cuts regained interest, see, e.g., Balas, Ceria, Cornuéjols, and Natraj [18]. One reason is that today's LP solvers are much more robust; see [18] for a more detailed analysis.

In our context we have to introduce slack variables to obtain the following equality-form of the set covering formulation, derived from (5.2):

$$\begin{aligned} &M\boldsymbol{y} + \boldsymbol{s} = \mathbb{1} \\ &\boldsymbol{y} \in \{0,1\}^m \\ &\boldsymbol{s} \leq \mathbf{0},\ \boldsymbol{s} \in \mathbb{Z}^k, \end{aligned}$$

where $M \in \mathbb{Z}^{k \times m}$ is the constraint matrix of the current LP relaxation. Hence, we assume that all cutting planes used here have integer coefficients. We can then apply the Gomory procedure to this formulation which yields a cutting plane, from which the slack variables have to be eliminated and which has to be made integer.

As suggested in [18], in each round we generate a Gomory cut for every fractional variable. Furthermore, as mentioned above, we store Gomory cuts in a pool and try to separate from there. Since violated Gomory cuts for a fractional solution can always be found, the user has to set limits on the number of Gomory rounds that are performed in each node. In the computational tests presented below, 25 rounds were performed in the root node and 15 in every other node. For higher values numerical problems occur, i.e., the basis matrix gets numerically almost singular.

5.3 Computational Experience

In the following sections we report on the results of the branch-and-cut implementation of the branch-and-cut algorithm for different problem sets and different combinations of the methods proposed above.

The branch-and-cut algorithm was implemented using the framework ABACUS of Thienel [112]. We used ABACUS version 2.2 on a 200 MHz Sun Ultra Sparc 2 machine running under Solaris 5.8 with 512 MByte memory. As an LP solver we used ILOG CPLEX version 6.6. All running times are measured in seconds.

5.3.1 Numerical Issues

Before discussing the computational results, we give a disclaimer concerning numerical errors occurring during the computation and the limited precision of LP solvers. This has rarely been addressed in the literature so far.

At the beginning of solving a particular MIN IIS COVER or MAX FS instance, we have to determine whether it is infeasible. At the end, a claimed solution to MAX FS has to be verified, i.e., tested for feasibility. Furthermore, most algorithms for MIN IIS COVER/MAX FS have to perform (many)

Table 5.1: Abbreviations used in the Tables of Section 5.3.

n	dimension of the space
m	number of inequalities
sub	number of subproblems solved
LP	total number of LPs solved
D	depth of the branch-and-cut tree
CPU	CPU time for the whole optimization process in seconds
sep	total CPU time for the separation process in seconds
IIS	number of IISs found
C_{bn}	number of Balas and Ng cuts found
C_{go}	number of Gomory cuts found
bnd	lower bound obtained in the root node*
sol	optimal objective value of the problem
s	number of random instances successfully solved for each size

* Since the objective function value is integer, lower bounds are always rounded up.

tests for infeasibility before terminating. In almost all cases, an LP solver working with fixed precision (e.g., doubles) is used to perform these tests. Currently, using exact solvers is out of question because this would result in a prohibitively slow code. Moreover, in some applications the user is restricted to a specific LP solver; for instance, in the linear programming application of Min IIS Cover, the user might want to know why his or her favorite LP solver reports an infeasibility.

Hence, in most practical cases infeasibility is decided with a fixed limited precision. The LP-solver CPLEX, for instance, only works with a precision of 10^{-5}. In most cases this is sufficient for finding a (near) optimal solution and most LP solvers are tuned to find such an solution very fast. Usually, it doesn't matter whether the cost of a solution is off the optimal cost by 10^{-5}. For testing feasibility, however, limited precision can lead to completely wrong decisions. This can already happen if the system is mildly ill conditioned. Even more, the decisions could differ when using the alternative system to check feasibility or when testing a subsystem directly (see Section 5.1.1). The decision could also be different if one uses different starting bases for the dual simplex algorithm.

These are principal problems that cannot be avoided without making the solution process very slow. The consequence is that one should not trust the results of algorithms in this area too much, as one should not trust LP solvers too much if they report infeasibility. On the other hand, some problems might be solved (correctly) by tuning the numerical behavior of the LP solver.

5.3.2 Results for the Netlib Problems

We turn to the first set of test instances for the branch-and-cut implementation for MIN IIS COVER. In the Netlib library a well known set of infeasible linear inequality systems are available (at "`http://www.netlib.org`"). It provides a list of 29 infeasible problems. Additionally, there are three more problems (`gams10am`, `gams30am`, `gams60am`), which are available at "`ftp://ftp.sztaki.hu/pub/oplab/LPTESTSET/INFEAS/`".

We removed problem `gran` from the test set, since it provides severe numerical problems. It also could not be tackled by other known approaches, i.e., the ones of Parker and Ryan [95] and Chinneck [48].

Table 5.2 gives results of the branch-and-cut algorithm for this test set. For these problems it suffices to only separate IIS-inequalities by method (2) of Section 5.1.3. Additionally the primal heuristic, explained in Section 5.1.4, is run at every new level of the branch-and-bound tree. We used a breadth-first search strategy for the tree. The abbreviations used in the tables are explained in Table 5.1.

The results show that these problems are very easy to solve. In only two cases the algorithm had to do a branching step. All other problems could be solved in the root node. Interestingly, for both exceptional problems the optimal lower bound could be obtained in the root node, i.e., these two problems could have been solved in the root node if an optimal primal solution had been found earlier. Moreover, almost all problems could be solved within a few seconds. The only exceptions are problem `gosh`, which was solved in 76 seconds, and problem `greanbea`, which needed 169 seconds to be solved; this is due to the greater effort needed to solve the intermediate linear programs during the algorithm. We furthermore observe that only very few IISs are found/needed to solve the problems.

The explanation for this good overall performance seems to be the following: For 23 problems only one inequality needs to be removed to make them feasible, for four additional problems removing two inequalities suffice. The highest number of inequalities that have to be removed were 12 for `bgdbg1`. Thus, our branch-and-cut algorithm seems to perform well on problems that have a small MIN IIS COVER objective function.

Parker [94] and Parker and Ryan [95] report comparable results; their approach is described in Section 5.1. Unfortunately, the only computational results available for this approach are for the Netlib problems. Hence, a fair comparison to our branch-and-cut implementation is not possible.

Finally, the heuristics of Chinneck [48] find the optimal solution for almost all these problems in a very short time. For the problems in Section 5.3.4, the performance of Chinneck's heuristics is also very good.

Table 5.2: Results of the branch-and-cut algorithm on the *Netlib problems*. The only cutting planes used are IIS-inequalities, separated by method (2) of Section 5.1.3. Additionally the primal heuristic is run. See Table 5.1 for an explanation of the abbreviations.

Name	m	n	sub	LP	D	CPU	sep	IIS	bnd	sol
bgdbg1	349	407	1	5	1	0	0	30	12	12
bgetam	401	688	1	2	1	1	0	8	1	1
bgindy	2672	10116	1	2	1	14	0	1	1	1
bgprtr	21	34	1	2	1	0	0	1	1	1
box1	232	261	1	2	1	0	0	3	1	1
ceria3d	3577	824	1	3	1	7	0	40	1	1
chemcom	289	720	1	2	1	0	0	1	1	1
cplex1	3006	3221	1	2	1	7	0	1	1	1
cplex2	225	221	1	18	1	10	0	20	1	1
ex72a	198	215	1	2	1	0	0	3	1	1
ex73a	194	211	1	2	1	0	0	4	1	1
forest6	67	95	1	2	1	0	0	2	1	1
galenet	9	8	1	2	1	0	0	1	1	1
gams10am	114	61	1	3	1	0	0	4	4	4
gams30am	354	181	1	3	1	0	0	5	5	5
gams60am	714	361	1	2	1	0	0	1	1	1
gosh	3793	10733	1	2	1	76	0	1	1	1
greenbea	2505	5405	3	5	2	169	51	27	2	2
itest2	10	4	1	3	1	0	0	4	2	2
itest6	12	8	1	3	1	0	0	4	2	2
klein1	55	54	1	2	1	0	0	1	1	1
klein2	478	54	1	3	1	1	0	27	1	1
klein3	995	88	1	4	1	21	0	107	1	1
mondou2	313	604	3	7	2	1	0	16	3	3
pang	362	460	1	2	1	0	0	1	1	1
pilot4i	411	1000	1	2	1	0	0	1	1	1
qual	324	464	1	2	1	1	0	8	1	1
reactor	319	637	1	2	1	0	0	2	1	1
refinery	324	464	1	6	1	6	0	36	1	1
vol1	324	464	1	4	1	2	0	11	1	1
woodinfe	36	89	1	2	1	0	0	2	2	2

5.3.3 Results for Random Inequality Systems

As reported in the last section, the infeasible inequality systems in the Netlib library are easily solved by our branch-and-cut implementation. Hence, these problems are not helpful for testing the practical limits of the implementation. In this section we report on the performance of our branch-and-cut implementation on random inequality systems. We generated random instances for different combinations of the number of inequalities and the dimension of the space. This collection turns out to provide a good test set for the implementation.

We generated these random inequality systems as follows: Each coefficient of an inequality was chosen to be an integer in the range $-K$ to K, where K was set to 100 in our tests. The other parameters that could be chosen were the dimension of the space n, the number of inequalities m, and a seed value for the pseudo-random number generator. After the inequality system was generated, it was tested for infeasibility; of course, only infeasible problems were accepted. If m was large enough compared to n (about twice as big), almost every system was infeasible.

The hyperplanes defined by these inequalities are in general position with high probability. It follows that the corresponding alternative polyhedra are simple/nondegenerate (with high probability). This leads to the question, whether MIN IIS COVER restricted to such systems is $\mathcal{NP}$-hard. This question seems to be open, see Section 2.3.2. Nevertheless, if m and n are large enough, solving random inequality systems is hard for our branch-and-cut implementation.

Below we report on results on such random problems. We altogether consider 25 combinations of values for n and m, see, e.g., Table 5.5. For each combination *three* problems were generated. The dimension of the space n was chosen to be 5, 10, 15, and 20. The number of inequalities m varied between 10 and 80 and depends on n, because m has to be large enough for the random systems to be infeasible. Furthermore, m was chosen such that at least one problem of each size could be solved.

Each value in the tables below is the average over the three problems of each size. This should be a fair choice between providing a good estimate of the average performance and reasonable running times. Nevertheless, the values given can only provide a rough trend. We will discuss this issue below in more detail.

We turn to a more thorough discussion of the results of the branch-and-cut algorithm. Tables 5.5 to 5.14 provide results of different strategies for the separation of cuts, which are explained in Section 5.1 and 5.2. Table 5.3

Table 5.3: Overview of the separation methods used in Tables 5.5 to Table 5.14. In column "ph" we indicate whether the primal heuristic is used, see Section 5.1.4. For IIS-inequalities we refer to Section 5.1.3, for Balas and Ng cuts we refer to Section 5.2.1, and for Gomory cuts we refer to Section 5.2.2.

Table	ph	Separation Methods
5.5	n	IIS-inequalities (method (2))
5.6	y	IIS-inequalities (method (2))
5.9	y	IIS-inequalities (methods (2) and (3))
5.11	y	Balas and Ng cuts
5.12	y	Gomory cuts (25 in root node, 15 in other nodes)
5.13	y	IIS-inequalities (methods (2) and (3)) and Balas and Ng cuts
5.14	y	IIS-inequalities (methods (2) and (3)), Balas and Ng cuts, and Gomory cuts (25 root, 15 other)

gives an overview. The abbreviations used in the tables are summarized in Table 5.1. At the bottom of most tables the sums over the columns are given. Here, the last row gives sums of columns over the rows that are marked with "⋆". The corresponding problems are chosen such that almost all variants of the branch-and-cut implementation terminated with the optimal value for all three problems of this size. Hence, the last row can be used for a fair comparison between the variants. These values are also plotted in Figure 5.1 on page 138.

The column indicated by s (if present) gives the number of the three problems of each size that could be solved to optimality, given a time constraint of 12 hours and a limit of the available main memory of 512 MByte. We always used *depth first search* in the tree to save memory space. The partial results of problems that could not be solved to optimality are not included in the tables, except in Table 5.7. If $s = 0$, we leave the remaining columns empty.

Moreover, due to a memory leakage in either ABACUS or CPLEX, the most severe constraint was the memory size. After roughly 4000 nodes of the branch-and-cut tree the memory was exhausted. The time when this occurs depends on the particular problem. We think that the tables nonetheless provide a good overview of the performance of the implementation.

We now give an analysis of the results.

Primal Heuristic. Tables 5.5 and 5.6 give results of the "plain" algorithm, i.e., using only the IIS-inequality separation method (2) of Section 5.1.3. The latter table additionally uses the heuristic, explained in Section 5.1.4, to find primal solutions; it is applied every 50 nodes of the branch-and-bound

tree. The performance of this primal heuristic is examined in Table 5.7, where the results of the former two tables are compared. This comparison clearly shows the advantages of using the primal heuristic. The number of subproblems and LPs needed to solve the problems is much smaller with the primal heuristic. The same is true for the running times.[1] We conclude that the additional time needed to use the heuristic is well invested. For the larger problems (with longer computation times) the primal heuristic finds a relatively good solution in the root node. Moreover, in most cases the optimal solution is found very early during the optimization process. This seems to be an indication that if one would stop the branch-and-cut algorithm before termination, one can hope for a very good primal solution; compare also the results in Section 5.3.4. Because of these results, we used the primal heuristic in all of the following tables.

Basic Observations. Several not surprising observations can be made about the results of Table 5.6. Clearly, the larger the dimension of the space, the more time is needed to solve the LPs during the algorithm. Hence, the total CPU-time needed to solve problems for different dimensions of the space, with the same number of inequalities, increases with growing n. On the other hand, we observe that the average size of an optimal solution of those problems that could be solved to optimality decreases with growing n. This seems to strengthen the conclusion that problems are tractable for the branch-and-cut implementation if the objective value is small. We also observe that the number of IISs found by the implementation clearly grows with m, but the trend for increasing n is unclear.

Number of IISs. In Table 5.4 we list the number of IISs for several small random inequality problems. This table was computed as follows: First the alternative polyhedron P was constructed using the `polymake` framework of Gawrilow and Joswig [63, 64]. The vertices of the (simple) polyhedron P were enumerated by `lrs` [11, 12], Avis's implementation of the reverse search method of Avis and Fukuda [14]; we refer to Section 2.2 for an explanation of the connection between vertices of the alternative polyhedron and IISs. The number of IISs for larger problems with $n = 5$ and $n = 10$ could not be computed since 1 GB of memory did not suffice.[2]

In Table 5.4, for each combination of n and m given, we took the first

[1] Using the primal heuristic one additional problem of the ones marked with "$\star$" could be solved; it has size $n = 5$ and $m = 60$. This is the reason for the total sums in the last row of Table 5.6 being larger than the corresponding total sums in Table 5.5. If the sums are corrected accordingly, the values are in favor for using the primal heuristic.

[2] Dyer [53] proved that the following problem is $\mathcal{NP}$-hard, even when restricted to simple polyhedra: Given a polyhedron P and a number k, does P have at least k vertices? We also refer to Problem 7 of [77].

Table 5.4: The number of IISs for a couple of small *random inequality systems*.

n	m	IISs
5	10	16
5	20	1733
5	30	56036
5	40	400015
10	20	314
10	30	47538

of the three problems of this size. Hence, the numbers of IISs given in this table can be compared to the number of IISs found during the branch-and-cut algorithm in Table 5.8, where the results for the complete list of problems is given; also see the next topic. We conclude that only a very small fraction of the total number of IISs are actually needed or found by the branch-and-cut implementation.

Variance. Table 5.8 lists results of the implementation using separation methods (2) and (3) of Section 5.1.3 for the *complete* set of all 75 problems. The mean values of these data are given in Table 5.9, and the corresponding standard deviations are provided in Table 5.10. We observe that the standard deviation gets larger with growing n and m. The large variance is mainly due to problems that are much easier to solve than the other two problems of the same size, compare, e.g., the results in Table 5.8 for $n = 15$ and $m = 50$ or $m = 60$. Furthermore, we can suspect that for problems of larger size we actually could only solve the easier problems, while the harder problems could not be solved within the memory and time constraints as they were given for our tests. We conclude that the mean values that are given in the tables provide only a rough estimate of the "real values". Nevertheless, the data seems to be significant enough to evaluate the performance of our branch-and-cut implementation.

Performance of the Cuts. We now compare six different combinations of the cutting plane generation strategies we mentioned in the beginning of Section 5.2. See Table 5.3 for an overview. To estimate the performance of these variants, we compare their results to the data given in Table 5.6, which provides results of a "plain method". This table gives the results for the most simple strategy: Only IIS-inequalities are separated with method (2) of Section 5.1.3, and we use the primal heuristic as explained in Section 5.1.4. Method (2) is used in all of the variants discussed below.

Figure 5.1 shows a comparison of the total sums of important values given in the tables for these six variants; only the problems marked with "⋆" are considered.

- We first observe that applying method (3) of Section 5.1.3 for separating IIS-inequalities consistently reduces the number of subproblems and LPs needed when compared to the "plain method", but in general increases the CPU-time, cf. Table 5.9. The number of found IISs drastically increase. Moreover, one additional problem could be solved.
- Applying the cuts of Balas and Ng, as explained in Section 5.2.1, in total decreases the number of subproblems needed compared to the "plain method". But it also sometimes increases them, cf. Table 5.11. The number of needed LPs on average roughly stays the same. The CPU-time is consistently higher than for the plain version.
- When Gomory cuts are used, in total the number of subproblems decreases, but the number of LPs increases, cf. Table 5.12. Moreover, the total CPU-time increases extremely. Furthermore, because of memory constraints, this variant could solve three problems less, compared to the "plain method".
- When combining method (3) for separating IIS-inequalities and the cuts of Balas and Ng, the numbers of subproblems and LPs once more decrease when compared to the results when only one of the methods was used. The CPU-time increases compared to using only one method, see Table 5.13. Some of the larger problems could not be solved within 12 hours.
- If we combine all three methods, i.e., method (3) for separating IIS-inequalities, the cuts of Balas and Ng, and Gomory cuts, the total number of subproblems are in general the lowest of all. The number of LPs is only higher for the version using Gomory cuts only. The CPU-time needed is the highest of all, see Table 5.14. For all problems for which no optimal solution could be obtained, the implementation terminated because of the 12 hour CPU time constraint.

Interestingly, using different and more additional cuts does not imply that more of the larger problems could be solved. This may be due to the imposed memory and time constraints.

Table 5.5: Results of the branch-and-cut algorithm on *random inequality systems*. Given are mean values over the three problems for each size. See Table 5.1 for an explanation of the abbreviations. The last two rows give sums over the columns, where the last row gives sums over the rows marked with "⋆".
The only cutting planes used are IIS-inequalities, separated by method (2) of Section 5.1.3. No primal heuristic is used.

n	m	sub	LP	D	CPU	sep	IIS	bnd	sol	s	
5	10	1.0	2.3	1.0	0.0	0.0	2.3	1.0	1.0	3	⋆
5	20	1.0	4.7	1.0	0.1	0.0	13.3	2.7	2.7	3	⋆
5	30	23.0	84.0	5.7	0.5	0.1	88.0	4.3	5.7	3	⋆
5	40	109.7	418.0	10.7	3.8	1.6	180.7	5.3	7.7	3	⋆
5	50	587.0	2586.0	14.3	45.7	26.0	384.7	7.0	11.0	3	⋆
5	60	311.0	1302.5	16.0	19.4	9.4	382.5	7.5	12.5	2	⋆
5	70	1331.0	6053.0	18.5	161.4	99.0	675.0	9.5	16.0	2	
5	80									0	
10	20	1.0	2.3	1.0	0.1	0.0	3.0	1.0	1.0	3	⋆
10	30	5.0	17.3	2.7	0.1	0.0	23.0	2.0	2.7	3	⋆
10	40	82.3	279.0	8.0	2.2	0.9	136.0	3.0	4.0	3	⋆
10	50	593.0	2313.7	14.7	40.2	21.6	402.7	4.0	7.0	3	⋆
10	60	721.0	2924.0	18.0	54.1	26.0	495.0	5.0	9.0	1	
10	70									0	
15	30	1.0	3.3	1.0	0.0	0.0	4.3	1.0	1.0	3	⋆
15	40	7.7	24.3	3.7	0.3	0.0	28.7	1.7	2.0	3	⋆
15	50	107.7	340.3	10.0	3.6	1.5	150.0	2.7	3.7	3	⋆
15	60	1184.3	4202.7	14.7	90.7	54.4	540.3	2.7	5.3	3	⋆
15	70	1.0	8.0	1.0	0.3	0.1	57.0	3.0	3.0	1	
15	80	765.0	2617.0	18.0	54.2	31.3	683.0	3.0	7.0	1	
20	40	1.0	4.0	1.0	0.1	0.0	9.0	1.0	1.0	3	⋆
20	50	95.7	292.7	7.3	3.6	1.5	98.0	2.0	2.7	3	⋆
20	60	89.0	249.3	15.0	3.3	1.5	126.3	2.0	4.0	3	⋆
20	70	1513.7	5253.0	21.0	154.2	101.2	693.3	3.0	5.3	3	
20	80	1187.0	3869.0	22.0	87.6	51.8	617.0	3.0	6.0	1	
$\sum$		8719.1	32850.4	226.3	725.5	427.9	5793.1	77.4	121.3	59	
		3200.4	12126.4	127.8	213.7	118.5	2572.8	50.9	75.0	50	⋆

Table 5.6: Results for the situation as in Table 5.5, except that the *primal heuristic* is called every 50 nodes of the branch-and-cut tree.

n	m	sub	LP	D	CPU	sep	IIS	bnd	sol	s	
5	10	1.0	2.3	1.0	0.0	0.0	2.3	1.0	1.0	3	$\star$
5	20	1.0	4.7	1.0	0.0	0.0	13.3	2.7	2.7	3	$\star$
5	30	15.7	58.0	4.7	0.4	0.1	74.7	4.3	5.7	3	$\star$
5	40	87.0	329.7	7.0	2.8	1.2	175.3	5.3	7.7	3	$\star$
5	50	519.7	2324.3	12.0	44.1	25.4	360.0	7.0	11.0	3	$\star$
5	60	902.3	4332.3	14.3	115.8	70.2	484.0	8.0	13.7	3	$\star$
5	70	1146.0	5296.0	18.5	139.2	83.0	642.0	9.5	16.0	2	
5	80	2081.0	10168.0	20.0	370.5	220.9	857.0	12.0	20.0	1	
10	20	1.0	2.3	1.0	0.0	0.0	3.0	1.0	1.0	3	$\star$
10	30	4.3	16.7	2.7	0.1	0.0	22.0	2.0	2.7	3	$\star$
10	40	65.0	201.7	7.7	1.6	0.7	115.0	3.0	4.0	3	$\star$
10	50	263.7	1030.7	12.3	15.9	7.5	301.3	4.0	7.0	3	$\star$
10	60	661.0	2729.0	19.0	55.5	27.0	534.0	5.0	9.0	1	
10	70	1931.0	8822.0	17.0	290.1	164.8	868.0	6.0	11.0	1	
15	30	1.0	3.3	1.0	0.1	0.0	4.3	1.0	1.0	3	$\star$
15	40	5.0	16.0	3.0	0.2	0.0	24.0	1.7	2.0	3	$\star$
15	50	90.3	289.3	10.0	3.2	1.5	143.0	2.7	3.7	3	$\star$
15	60	897.0	3152.3	15.3	63.0	36.2	451.7	2.7	5.3	3	$\star$
15	70	1.0	8.0	1.0	0.3	0.1	57.0	3.0	3.0	1	
15	80	461.0	1662.0	15.0	34.6	19.6	558.0	3.0	7.0	1	
20	40	1.0	4.0	1.0	0.1	0.0	9.0	1.0	1.0	3	$\star$
20	50	94.3	262.3	7.7	3.4	1.6	85.0	2.0	2.7	3	$\star$
20	60	82.3	235.0	15.7	3.6	1.9	124.3	2.0	4.0	3	$\star$
20	70	735.0	2451.3	20.7	55.0	31.3	454.3	3.0	5.3	3	
20	80	695.0	2336.0	24.0	57.4	32.1	520.0	3.0	6.0	1	
$\sum$		10742.6	45737.2	252.6	1256.9	725.1	6882.5	95.9	153.5	62	
		3031.6	12264.9	117.4	254.3	146.3	2392.2	51.4	76.2	51	$\star$

Table 5.7: In this table the performance of the primal heuristic for *random inequality systems* is investigated. The left part of the table shows data similar to Table 5.5, where no primal heuristic is used. The right part displays the data as in Table 5.6, i.e., the primal heuristic is called every 50 nodes in the branch-and-bound tree. In both cases method (2) for separating IIS-inequalities was used. The values in the column labeled "root" show the objective function value of the best primal solution found in the root node, "best" the objective function value of the best primal solution found in the whole optimization process (not all problems could be solved optimally), "bestsub" gives the number of the subproblem where the best solution was found, "sub" gives the total number of subproblems, and the column labeled "%" gives the fraction in percent of subproblems solved until the best solution was found. Again all values are means over the three problems of each size. In this table also problems that could not be solved to optimality are counted.

		no primal heuristic					with primal heuristic				
n	m	root	best	bestsub	sub	%	root	best	bestsub	sub	%
5	10	1.0	1.0	1.0	1.0	100.0	1.0	1.0	1.0	1.0	100.0
5	20	2.7	2.7	1.0	1.0	100.0	2.7	2.7	1.0	1.0	100.0
5	30	11.0	5.7	21.0	23.0	91.3	7.0	5.7	13.7	15.7	87.2
5	40	13.0	7.7	102.0	109.7	93.0	8.7	7.7	83.7	87.0	96.2
5	50	19.3	11.0	388.0	587.0	66.1	13.0	11.0	278.3	519.7	53.6
5	60	26.7	13.7	1168.0	1308.7	89.3	17.7	13.7	671.7	902.3	74.4
5	70	30.3	18.0	1333.3	1905.0	70.0	23.0	17.7	440.3	1745.7	25.2
5	80	36.3	23.7	1648.0	2959.7	55.7	26.0	21.7	116.0	2590.0	4.5
10	20	1.0	1.0	1.0	1.0	100.0	1.0	1.0	1.0	1.0	100.0
10	30	3.0	2.7	1.7	5.0	33.3	2.7	2.7	1.0	4.3	23.1
10	40	8.0	4.0	79.3	82.3	96.4	4.7	4.0	58.3	65.0	89.7
10	50	15.3	7.0	477.3	593.0	80.5	9.3	7.0	45.3	263.7	17.2
10	60	24.7	12.3	759.0	2759.3	27.5	15.7	11.0	1018.0	2533.7	40.2
10	70	28.3	15.0	315.3	3293.0	9.6	18.0	13.7	9.7	2696.3	0.4
10	80	40.0	22.3	2761.0	3391.3	81.4	26.3	18.7	194.3	2893.0	6.7
15	30	1.0	1.0	1.0	1.0	100.0	1.0	1.0	1.0	1.0	100.0
15	40	3.0	2.0	2.7	7.7	34.8	2.0	2.0	1.0	5.0	20.0
15	50	6.0	3.7	44.0	107.7	40.9	5.0	3.7	34.3	90.3	38.0
15	60	10.7	5.3	351.7	1184.3	29.7	7.7	5.3	10.3	897.0	1.2
15	70	15.7	7.7	1078.3	2504.7	43.1	9.7	7.7	288.7	2365.0	12.2
15	80	23.7	12.7	1888.3	2771.7	68.1	16.3	11.3	8.0	2364.7	0.3
20	40	1.0	1.0	1.0	1.0	100.0	1.0	1.0	1.0	1.0	100.0
20	50	4.3	2.7	70.3	95.7	73.5	3.3	2.7	62.0	94.3	65.7
20	60	8.0	4.0	4.7	89.0	5.2	4.7	4.0	1.3	82.3	1.6
20	70	11.7	5.3	1101.7	1513.7	72.8	7.0	5.3	50.3	735.0	6.8
20	80	20.0	8.0	633.7	2730.3	23.2	11.3	7.7	31.7	2524.7	1.3

Table 5.8: Results of the branch-and-cut algorithm on *random inequality systems*. Here, *methods (2) and (3)* of Section 5.1.3 are used to separate IIS-inequalities. Additionally, the primal heuristic is run every 50 nodes. We give the complete results on *all 75 problems* considered.

n	m	sub	LP	D	CPU	sep	IIS	bnd	sol
5	10	1	3	1	0.03	0.00	4	1	1
5	10	1	2	1	0.02	0.00	2	1	1
5	10	1	2	1	0.05	0.00	1	1	1
5	20	1	5	1	0.03	0.00	14	3	3
5	20	1	3	1	0.04	0.00	9	2	2
5	20	1	6	1	0.07	0.00	17	3	3
5	30	17	57	7	0.40	0.17	94	5	6
5	30	3	12	2	0.12	0.01	56	4	5
5	30	29	92	7	0.68	0.35	97	5	6
5	40	19	78	6	0.77	0.41	130	7	8
5	40	5	21	3	0.22	0.08	59	5	6
5	40	69	334	10	3.72	1.76	264	6	9
5	50	55	308	10	4.80	2.29	340	8	11
5	50	19	92	5	0.89	0.40	112	6	8
5	50	511	2893	13	66.74	40.62	755	8	14
5	60	305	1663	13	40.92	23.66	579	9	13
5	60	113	537	12	9.17	4.74	389	9	12
5	60	537	3411	14	112.29	68.67	938	10	16
5	70	875	4674	14	160.01	99.97	863	11	17
5	70	425	2365	11	65.58	37.36	663	10	15
5	70	0	0	0	0.00	0.00	0	0	0
5	80	903	5643	16	285.43	185.95	1217	13	20
5	80	1373	8359	18	462.18	320.40	1356	12	20
5	80	0	0	0	0.00	0.00	0	0	0
10	20	1	2	1	0.02	0.00	2	1	1
10	20	1	2	1	0.05	0.00	4	1	1
10	20	1	3	1	0.04	0.00	3	1	1
10	30	1	4	1	0.08	0.00	8	2	2
10	30	7	26	4	0.24	0.10	37	2	3
10	30	5	20	3	0.18	0.07	25	2	3
10	40	5	24	3	0.33	0.16	52	3	4
10	40	139	533	14	6.25	3.66	358	4	5
10	40	9	27	5	0.30	0.16	40	3	3
10	50	299	1665	16	71.72	54.20	1974	5	7
10	50	219	1871	13	201.55	167.79	5095	5	8
10	50	23	175	7	4.25	2.73	481	4	6
10	60	0	0	0	0.00	0.00	0	0	0

Table 5.8: continued. Results on all 75 problems.

n	m	sub	LP	D	CPU	sep	IIS	bnd	sol
10	60	0	0	0	0.00	0.00	0	0	0
10	60	235	2127	13	282.81	238.17	6186	6	9
10	70	0	0	0	0.00	0.00	0	0	0
10	70	0	0	0	0.00	0.00	0	0	0
10	70	925	8775	15	4274.72	3976.56	21321	6	11
15	30	1	2	1	0.05	0.00	3	1	1
15	30	1	6	1	0.09	0.00	9	1	1
15	30	1	2	1	0.06	0.00	1	1	1
15	40	13	37	7	0.51	0.28	44	2	3
15	40	3	4	2	0.12	0.01	11	2	2
15	40	1	5	1	0.10	0.00	10	1	1
15	50	123	734	14	36.06	29.59	2095	3	5
15	50	1	5	1	0.10	0.00	13	2	2
15	50	23	179	9	5.41	4.27	505	3	4
15	60	607	3929	17	627.67	559.06	9163	4	7
15	60	3	5	2	0.22	0.01	33	2	2
15	60	575	3864	16	593.03	523.99	8427	4	7
15	70	0	0	0	0.00	0.00	0	0	0
15	70	3	15	2	0.63	0.31	91	3	3
15	70	0	0	0	0.00	0.00	0	0	0
15	80	0	0	0	0.00	0.00	0	0	0
15	80	353	2094	15	174.49	146.99	3907	4	7
15	80	0	0	0	0.00	0.00	0	0	0
20	40	1	5	1	0.11	0.00	11	1	1
20	40	1	4	1	0.15	0.00	11	1	1
20	40	1	3	1	0.07	0.00	5	1	1
20	50	49	364	16	23.57	20.17	1615	3	4
20	50	1	6	1	0.17	0.00	14	2	2
20	50	5	23	3	0.46	0.17	34	2	2
20	60	21	203	11	16.75	14.48	1143	3	4
20	60	23	247	11	23.59	20.83	1433	3	4
20	60	31	194	12	8.19	6.75	497	2	4
20	70	447	3336	18	997.15	926.65	14848	3	6
20	70	129	882	15	87.78	76.94	2941	3	4
20	70	1149	6801	24	1705.46	1575.93	16962	3	6
20	80	0	0	0	0.00	0.00	0	0	0
20	80	647	4335	23	1365.63	1270.37	15830	3	6
20	80	0	0	0	0.00	0.00	0	0	0

Table 5.9: Results of the branch-and-cut algorithm on *random inequality systems*. Given are mean values of the data in Table 5.8.
The *two separation methods (2) and (3)* of Section 5.1.3 are used to separate IIS-inequalities. Additionally, the primal heuristic is run every 50 nodes.

n	m	sub	LP	D	CPU	sep	IIS	bnd	sol	s	
5	10	1.0	2.3	1.0	0.0	0.0	2.3	1.0	1.0	3	*
5	20	1.0	4.7	1.0	0.0	0.0	13.3	2.7	2.7	3	*
5	30	16.3	53.7	5.3	0.4	0.2	82.3	4.7	5.7	3	*
5	40	31.0	144.3	6.3	1.6	0.8	151.0	6.0	7.7	3	*
5	50	195.0	1097.7	9.3	24.1	14.4	402.3	7.3	11.0	3	*
5	60	318.3	1870.3	13.0	54.1	32.4	635.3	9.3	13.7	3	*
5	70	650.0	3519.5	12.5	112.8	68.7	763.0	10.5	16.0	2	
5	80	1138.0	7001.0	17.0	373.8	253.2	1286.5	12.5	20.0	2	
10	20	1.0	2.3	1.0	0.0	0.0	3.0	1.0	1.0	3	*
10	30	4.3	16.7	2.7	0.2	0.1	23.3	2.0	2.7	3	*
10	40	51.0	194.7	7.3	2.3	1.3	150.0	3.3	4.0	3	*
10	50	180.3	1237.0	12.0	92.5	74.9	2516.7	4.7	7.0	3	*
10	60	235.0	2127.0	13.0	282.8	238.2	6186.0	6.0	9.0	1	
10	70	925.0	8775.0	15.0	4274.7	3976.6	21321.0	6.0	11.0	1	
15	30	1.0	3.3	1.0	0.1	0.0	4.3	1.0	1.0	3	*
15	40	5.7	15.3	3.3	0.2	0.1	21.7	1.7	2.0	3	*
15	50	49.0	306.0	8.0	13.9	11.3	871.0	2.7	3.7	3	*
15	60	395.0	2599.3	11.7	407.0	361.0	5874.3	3.3	5.3	3	*
15	70	3.0	15.0	2.0	0.6	0.3	91.0	3.0	3.0	1	
15	80	353.0	2094.0	15.0	174.5	147.0	3907.0	4.0	7.0	1	
20	40	1.0	4.0	1.0	0.1	0.0	9.0	1.0	1.0	3	*
20	50	18.3	131.0	6.7	8.1	6.8	554.3	2.3	2.7	3	*
20	60	25.0	214.7	11.3	16.2	14.0	1024.3	2.7	4.0	3	*
20	70	575.0	3673.0	19.0	930.1	859.8	11583.7	3.0	5.3	3	
20	80	647.0	4335.0	23.0	1365.6	1270.4	15830.0	3.0	6.0	1	
$\sum$		5820.2	39436.8	218.4	8135.7	7331.5	73306.6	104.7	153.5	63	
		1294.2	7897.3	101.9	620.8	517.3	12338.4	56.7	76.2	51	*

Table 5.10: Results of the branch-and-cut algorithm on *random inequality systems*. Given are *standard deviations* for the data in Table 5.8. The corresponding mean values are presented in Table 5.9.

n	m	sub	LP	D	CPU	sep	IIS	bnd	sol	s	
5	10	0.0	0.5	0.0	0.0	0.0	1.2	0.0	0.0	3	$\star$
5	20	0.0	1.2	0.0	0.0	0.0	3.3	0.5	0.5	3	$\star$
5	30	10.6	32.7	2.4	0.2	0.1	18.7	0.5	0.5	3	$\star$
5	40	27.5	136.1	2.9	1.5	0.7	85.0	0.8	1.2	3	$\star$
5	50	223.9	1272.6	3.3	30.2	18.5	266.2	0.9	2.4	3	$\star$
5	60	173.4	1182.4	0.8	43.1	26.8	227.6	0.5	1.7	3	
5	70	225.0	1154.5	1.5	47.2	31.3	100.0	0.5	1.0	2	
5	80	235.0	1358.0	1.0	88.4	67.2	69.5	0.5	0.0	2	
10	20	0.0	0.5	0.0	0.0	0.0	0.8	0.0	0.0	3	$\star$
10	30	2.5	9.3	1.2	0.1	0.0	11.9	0.0	0.5	3	$\star$
10	40	62.2	239.2	4.8	2.8	1.6	147.2	0.5	0.8	3	$\star$
10	50	115.9	755.6	3.7	81.9	69.0	1922.3	0.5	0.8	3	$\star$
10	60	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1	
10	70	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1	
15	30	0.0	1.9	0.0	0.0	0.0	3.4	0.0	0.0	3	$\star$
15	40	5.2	15.3	2.6	0.2	0.1	15.8	0.5	0.8	3	$\star$
15	50	53.1	310.9	5.4	15.8	13.1	888.5	0.5	1.2	3	$\star$
15	60	277.5	1834.7	6.8	288.0	255.7	4141.4	0.9	2.4	3	$\star$
15	70	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1	
15	80	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1	
20	40	0.0	0.8	0.0	0.0	0.0	2.8	0.0	0.0	3	$\star$
20	50	21.7	164.9	6.6	11.0	9.5	750.0	0.5	0.9	3	$\star$
20	60	4.3	23.2	0.5	6.3	5.8	391.2	0.5	0.0	3	$\star$
20	70	426.1	2428.1	3.7	662.1	613.8	6171.9	0.0	0.9	3	
20	80	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1	

Table 5.11: Results of the branch-and-cut algorithm on *random inequality systems*. Given are mean values for the three problems of each size.
The cuts of *Balas and Ng*, described in Section 5.2.1, are used. Additionally, the primal heuristic is run every 50 nodes.

n	m	sub	LP	D	CPU	sep	IIS	C_{bn}	bnd	sol	s	
5	10	1.0	2.3	1.0	0.0	0.0	2.3	0.0	1.0	1.0	3	*
5	20	1.0	4.7	1.0	0.1	0.0	13.3	1.0	2.7	2.7	3	*
5	30	13.0	56.3	4.3	0.6	0.3	71.0	87.7	4.7	5.7	3	*
5	40	51.0	251.3	7.0	4.4	2.9	136.3	222.0	5.7	7.7	3	*
5	50	422.3	2325.0	10.7	130.2	107.2	312.7	776.0	7.0	11.0	3	*
5	60	464.3	2558.3	13.7	174.0	144.6	387.3	1040.3	8.7	13.7	3	*
5	70	810.0	4392.5	15.5	355.1	298.3	537.5	1813.0	9.5	16.0	2	
5	80	2445.0	14408.0	18.0	1829.8	1594.4	853.0	993.0	12.0	20.0	1	
10	20	1.0	2.3	1.0	0.1	0.0	3.0	0.0	1.0	1.0	3	*
10	30	3.7	17.0	2.3	0.1	0.0	21.7	38.7	2.0	2.7	3	*
10	40	31.7	161.3	5.7	2.4	1.3	81.7	483.3	3.0	4.0	3	*
10	50	384.3	2506.3	11.7	189.3	156.6	319.7	930.0	4.3	7.0	3	*
10	60	539.0	3365.0	15.0	296.0	241.1	501.0	2024.0	5.0	9.0	1	
10	70	2115.0	13612.0	18.0	2123.7	1862.2	823.0	1764.0	6.0	11.0	1	
15	30	1.0	3.3	1.0	0.1	0.0	4.3	0.0	1.0	1.0	3	*
15	40	4.3	16.7	2.7	0.2	0.0	21.3	73.7	1.7	2.0	3	*
15	50	59.7	324.7	8.3	8.9	5.3	118.7	631.0	2.7	3.7	3	*
15	60	651.7	4111.3	13.3	354.4	287.7	452.7	1030.7	3.0	5.3	3	*
15	70	3.0	24.0	2.0	0.8	0.3	63.0	456.0	3.0	3.0	1	
15	80	397.0	2429.0	16.0	189.7	151.4	525.0	1364.0	3.0	7.0	1	
20	40	1.0	4.0	1.0	0.1	0.0	9.0	0.0	1.0	1.0	3	*
20	50	23.0	117.7	6.3	3.4	1.8	76.3	383.3	2.0	2.7	3	*
20	60	32.3	213.7	13.0	6.1	3.3	122.3	1317.3	2.7	4.0	3	*
20	70	1149.7	6905.3	17.7	637.0	506.9	659.0	1013.3	3.0	5.3	3	
20	80	555.0	3282.0	19.0	227.8	163.8	514.0	460.0	3.0	6.0	1	
$\sum$		10160.0	61094.0	225.2	6534.3	5529.4	6629.1	16902.3	98.7	153.5	62	
		2146.3	12676.2	104.0	874.4	711.0	2153.6	7015.0	54.2	76.2	51	*

Table 5.12: Results of the branch-and-cut algorithm on *random inequality systems*. Given are mean values for the three problems of each size.
Here we separate at most 25 *Gomory cuts* in the root node and at most 15 Gomory cuts are separated in the other nodes. As usual, the primal heuristic is run every 50 nodes.

n	m	sub	LP	D	CPU	sep	IIS	C_{go}	bnd	sol	s	
5	10	1.0	2.3	1.0	0.0	0.0	2.3	0.0	1.0	1.0	3	⋆
5	20	1.0	4.7	1.0	0.0	0.0	13.3	0.3	2.7	2.7	3	⋆
5	30	7.7	153.7	3.0	8.0	6.9	70.7	93.0	5.0	5.7	3	⋆
5	40	23.7	683.3	5.7	50.2	43.5	146.7	234.7	5.7	7.7	3	⋆
5	50	169.7	6506.3	8.7	784.2	692.7	332.0	330.0	7.7	11.0	3	⋆
5	60	232.3	9005.3	11.0	1696.1	1532.8	414.7	321.0	9.0	13.7	3	⋆
5	70	293.0	10869.0	12.0	2619.3	2397.6	535.5	125.0	10.0	16.0	2	
5	80										0	
10	20	1.0	2.3	1.0	0.1	0.0	3.0	0.0	1.0	1.0	3	⋆
10	30	3.0	46.3	2.0	1.9	1.5	23.7	157.3	2.3	2.7	3	⋆
10	40	18.3	415.0	4.3	30.9	26.7	89.0	172.7	3.3	4.0	3	⋆
10	50	339.7	11519.0	10.7	1252.4	1090.2	459.3	263.7	4.7	7.0	3	⋆
10	60	359.0	12582.0	17.0	2051.4	1827.5	587.0	173.0	5.0	9.0	1	
10	70										0	
15	30	1.0	3.3	1.0	0.1	0.0	4.3	0.0	1.0	1.0	3	⋆
15	40	2.3	35.3	1.7	1.2	0.9	24.0	144.0	1.7	2.0	3	⋆
15	50	99.7	2643.3	8.7	306.4	265.3	218.0	288.7	2.7	3.7	3	⋆
15	60	393.7	11935.3	11.7	1674.1	1481.5	507.3	188.0	3.3	5.3	3	⋆
15	70	3.0	64.0	2.0	10.4	9.0	67.0	483.0	3.0	3.0	1	
15	80	249.0	6608.0	13.0	1280.2	1153.6	571.0	182.0	4.0	7.0	1	
20	40	1.0	4.0	1.0	0.1	0.0	9.0	0.0	1.0	1.0	3	⋆
20	50	25.0	535.7	6.7	67.5	58.9	86.0	151.3	2.0	2.7	3	⋆
20	60	35.0	853.3	12.0	144.5	126.6	144.7	532.3	2.3	4.0	3	⋆
20	70	402.0	12271.5	16.5	2219.2	1948.7	553.0	605.5	3.0	5.0	2	
20	80	299.0	8451.0	17.0	1453.0	1306.0	570.0	252.0	3.0	6.0	1	
$\sum$		2960.1	95193.9	168.7	15651.2	13969.9	5431.5	4697.5	84.4	122.2	59	
		1355.1	44348.4	91.2	6017.7	5327.5	2548.0	2877.0	56.4	76.2	51	⋆

Table 5.13: Results of the branch-and-cut algorithm on *random inequality systems*. Given are mean values for the three problems of each size.
IIS-inequalities are separated by *methods (2) and (3)* of Section 5.1.3. Additionally, the cuts of *Balas and Ng* (see Section 5.2.1) are used. The primal heuristic is run every 50 nodes.

n	m	sub	LP	D	CPU	sep	IIS	C_{bn}	bnd	sol	s	
5	10	1.0	2.3	1.0	0.0	0.0	2.3	0.0	1.0	1.0	3	*
5	20	1.0	4.7	1.0	0.1	0.0	13.3	1.0	2.7	2.7	3	*
5	30	7.0	38.7	3.3	0.5	0.2	68.3	81.7	5.0	5.7	3	*
5	40	35.0	175.0	5.7	4.2	3.0	147.0	209.7	6.0	7.7	3	*
5	50	155.0	1041.0	9.3	85.6	74.1	383.3	297.3	7.7	11.0	3	*
5	60	163.0	1223.7	10.0	146.4	129.2	539.7	910.7	9.0	13.7	3	*
5	70	588.0	3741.5	14.5	470.5	415.4	752.5	738.0	11.0	16.0	2	
5	80	1356.0	9651.0	17.5	2023.2	1834.5	1353.0	2887.0	13.0	20.0	2	
10	20	1.0	2.3	1.0	0.0	0.0	3.0	0.0	1.0	1.0	3	*
10	30	2.3	16.3	1.7	0.2	0.1	22.3	61.7	2.3	2.7	3	*
10	40	12.3	92.0	4.7	2.0	1.3	91.0	438.7	3.3	4.0	3	*
10	50	109.0	1204.0	10.0	467.3	441.8	2814.7	871.3	5.0	7.0	3	*
10	60	161.0	1896.0	13.0	894.6	843.6	6158.0	1289.0	6.0	9.0	1	
10	70	391.0	6272.0	12.0	6691.4	6453.0	21471.0	1796.0	7.0	11.0	1	
15	30	1.0	3.3	1.0	0.1	0.0	4.3	0.0	1.0	1.0	3	*
15	40	1.7	10.0	1.3	0.2	0.1	21.7	97.0	2.0	2.0	3	*
15	50	41.0	360.3	7.0	40.6	34.7	947.7	927.7	2.7	3.7	3	*
15	60	261.0	2634.0	11.0	1301.4	1228.7	7338.7	1266.3	3.3	5.3	3	*
15	70	1.0	32.0	1.0	2.0	1.2	116.0	812.0	3.0	3.0	1	
15	80	217.0	2049.0	14.0	815.8	765.1	4507.0	3186.0	4.0	7.0	1	
20	40	1.0	4.0	1.0	0.1	0.0	9.0	0.0	1.0	1.0	3	*
20	50	13.7	113.0	6.0	10.0	7.9	491.3	124.7	2.3	2.7	3	*
20	60	19.7	212.7	10.3	24.2	20.2	1065.7	2099.0	3.0	4.0	3	*
20	70	345.0	3459.0	15.7	2146.3	2034.6	15410.7	1921.3	3.0	5.3	3	
20	80	205.0	2427.0	18.0	1558.6	1468.6	16521.0	1013.0	4.0	6.0	1	
$\sum$		4089.7	36664.8	191.0	16685.3	15757.3	80252.5	21029.1	109.3	153.5	63	
		825.7	7137.3	85.3	2082.9	1941.3	13963.3	7386.8	58.3	76.2	51	*

Table 5.14: Results of the branch-and-cut algorithm on *random inequality systems*. Given are mean values for the three problems of each size.
Here we use all separation methods: IIS-inequalities are separated by *methods (2) and (3)* of Section 5.1.3, the cuts of *Balas and Ng* (see Section 5.2.1) are used, and *Gomory cuts* (25 in the root node, 15 elsewhere) are separated. The primal heuristic is run every 50 nodes.

n	m	sub	LP	D	CPU	sep	IIS	C_{bn}	C_{go}	bnd	sol	s	
5	10	1.0	2.3	1.0	0.0	0.0	2.3	0.0	0.0	1.0	1.0	3	⋆
5	20	1.0	4.7	1.0	0.1	0.0	13.3	1.0	0.3	2.7	2.7	3	⋆
5	30	1.7	56.3	1.3	2.1	1.7	76.3	106.0	104.3	5.3	5.7	3	⋆
5	40	14.3	622.7	4.0	53.3	46.2	189.7	417.7	296.3	6.7	7.7	3	⋆
5	50	148.3	8699.0	6.7	1677.2	1529.4	543.7	868.7	203.7	8.0	11.0	3	⋆
5	60	110.3	6928.3	8.7	1725.6	1584.5	692.3	1194.3	126.7	10.0	13.7	3	⋆
5	70	310.0	18144.0	11.0	5710.0	5285.3	978.0	1682.5	399.0	11.5	16.0	2	
5	80											0	
10	20	1.0	2.3	1.0	0.1	0.0	3.0	0.0	0.0	1.0	1.0	3	⋆
10	30	1.7	54.7	1.3	1.6	1.2	38.7	205.7	83.0	2.7	2.7	3	⋆
10	40	10.3	503.7	3.7	39.1	32.5	230.3	829.7	241.3	3.7	4.0	3	⋆
10	50	64.3	5196.7	8.3	5883.2	5723.5	9219.7	592.3	294.0	5.0	7.0	3	⋆
10	60	73.0	7034.0	9.0	9183.4	8910.8	18534.0	1240.0	526.0	6.0	9.0	1	
10	70											0	
15	30	1.0	3.3	1.0	0.1	0.0	4.3	0.0	0.0	1.0	1.0	3	⋆
15	40	1.0	38.7	1.0	1.6	1.2	39.3	244.3	131.0	2.0	2.0	3	⋆
15	50	26.3	1554.7	6.7	535.7	498.7	2640.7	1071.3	162.7	2.7	3.7	3	⋆
15	60	138.3	11051.0	9.3	18403.5	17986.3	28890.3	1401.3	151.0	3.3	5.3	3	⋆
15	70	1.0	63.0	1.0	5.8	4.5	166.0	1017.0	159.0	3.0	3.0	1	
15	80	125.0	9365.0	13.0	9866.2	9546.7	16674.0	2431.0	536.0	4.0	7.0	1	
20	40	1.0	4.0	1.0	0.1	0.0	9.0	0.0	0.0	1.0	1.0	3	⋆
20	50	7.0	456.3	4.0	143.3	130.5	2565.3	461.7	74.7	2.3	2.7	3	⋆
20	60	13.7	949.7	7.3	300.8	271.9	4131.7	1481.0	346.7	3.0	4.0	3	⋆
20	70	9.0	714.0	5.0	264.4	240.0	5383.0	2182.0	327.0	3.0	4.0	1	
20	80	127.0	11543.0	15.0	26759.2	26173.3	73458.0	791.0	426.0	4.0	6.0	1	
$\sum$		1187.3	82991.3	121.3	80556.3	77968	164483.0	18218.5	4588.7	92.8	121.0	58	
		542.2	36128.4	67.3	28767.4	27807.6	49289.9	8875.0	2215.7	61.4	76.2	51	⋆

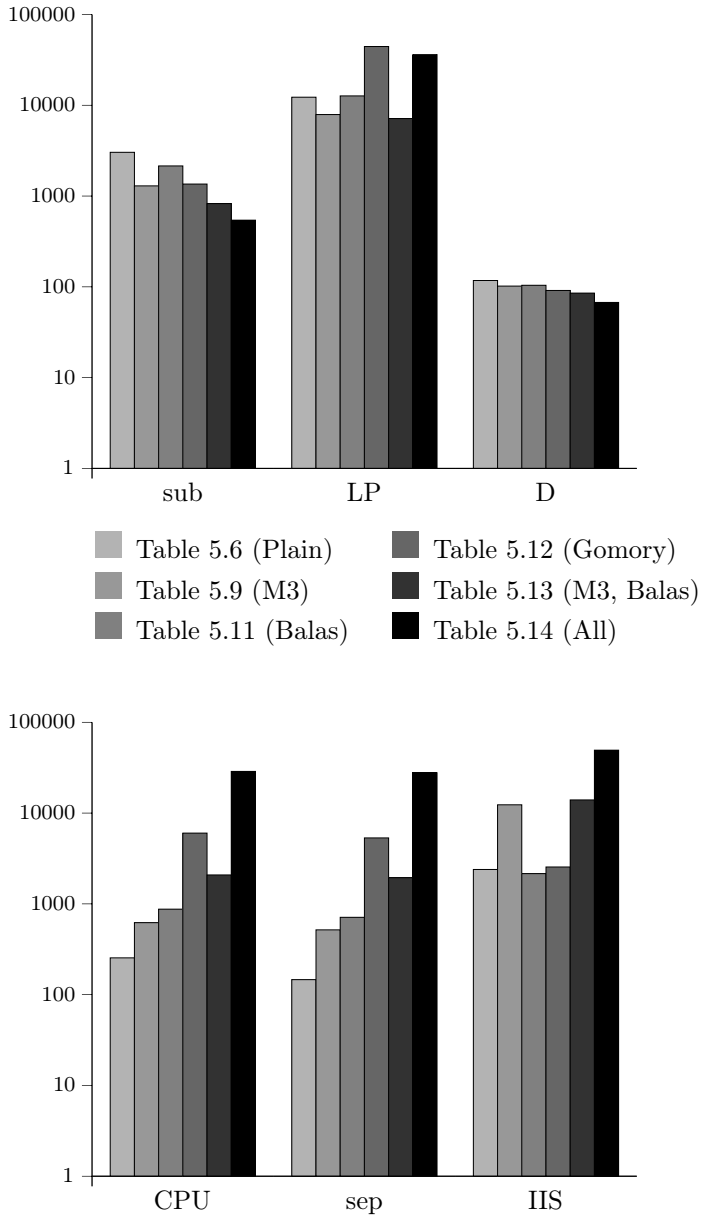


Figure 5.1: Comparison of the six different combinations of separation methods used in Tables 5.6 to 5.14; see Table 5.3 for an overview. The charts compare the entries of the six most important column sums over the rows marked with "⋆" of the tables listed in the legend; Table 5.1 explains the abbreviations used for the column titles. "Plain" stands for using method (2) and "M3" for method (3) for the separation of IIS-inequalities. "Balas" is short for the separation method of Balas and Ng. The values are plotted in *logarithmic scale*.

5.3.4 Results for Machine Learning Problems

In this section we discuss results of the branch-and-cut implementation for machine learning problems. These problems arise in a "classical" application of Min IIS Cover, which we already briefly presented in Section 1.1.2.

In this application, one is given m points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$ in $\mathbb{R}^{n-1}$, each belonging to one of two possible *classes* P_1 and P_2, i.e., P_1 and P_2 partition the set $\{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\}$. Each component of the points stores a measurement of an attribute relevant for the concrete application. We want to classify these points in $\mathbb{R}^{n-1}$ by an oriented hyperplane defined by $\boldsymbol{a}\boldsymbol{x} \leq \beta$, with $\boldsymbol{a} \in \mathbb{R}^{n-1}$ and $\beta \in \mathbb{R}$. The points in P_1 should satisfy the inequality $\boldsymbol{a}\boldsymbol{x} \leq \beta$ and the points in P_2 should violate it. Hence, we are looking for $(\boldsymbol{a}, \beta) \in \mathbb{R}^n$ such that

$$|\{\boldsymbol{p} \in P_1 \,:\, \boldsymbol{a}\boldsymbol{p} > \beta\}| + |\{\boldsymbol{p} \in P_2 \,:\, \boldsymbol{a}\boldsymbol{p} \leq \beta\}|$$

is minimized, i.e., the number of misclassified points should be minimized. This minimization is performed in order to maximize the chance that a new point can be correctly classified. This problem can be written as a Min IIS Cover problem with variables $(\boldsymbol{a}, \beta) \in \mathbb{R}^n$ and the following inequalities

$$\boldsymbol{p}\boldsymbol{a} - \beta \begin{cases} \leq 0 & \text{if } \boldsymbol{p} \in P_1 \\ > 0 & \text{if } \boldsymbol{p} \in P_2 \end{cases} \qquad \text{for each } \boldsymbol{p} \in \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\}.$$

In practice, this system is replaced by the system

$$\boldsymbol{p}\boldsymbol{a} - \beta \begin{cases} \leq 0 & \text{if } \boldsymbol{p} \in P_1 \\ \geq \varepsilon & \text{if } \boldsymbol{p} \in P_2 \end{cases} \qquad \text{for each } \boldsymbol{p} \in \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m\},$$

where $\varepsilon > 0$ is a small constant[3], depending on the points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$. Note that it might happen that this system is feasible, i.e., the points are completely separable (in which case Min IIS Cover reduces to solving one linear program).

We tested our branch-and-cut implementation on such classification problems from the UCI Repository of Machine Learning Databases (see Blake and Merz [32]). We picked problems from this database that are of the above type (i.e., have two classes) and are of reasonable size ($m < 1000$). Exceptions are the problems `glass`, `iris`, and `new-thyroid`; for these problems one tries to separate one class from the others. Most of these twelve problems are also

[3]For the test problems, discussed in the following, ε was 0.001.

Table 5.15: Characteristics of the *machine learning problems*. As usual n denotes the dimension of the space (number of attributes). The column labeled $m^\star$ gives the number of original data sets and m the number of data sets remaining after removing incomplete ones. The right column gives additional notes, e.g., the names of the problems in the UCI database.

Name	n	m	$m^\star$	Notes
breast-cancer	10	683	699	breast-cancer-wisconsin
bupa	7	345	345	liver-disorders
echo	9	61	132	echocardiogram
glass	10	214	214	type 2 vs. others
heart	14	297	303	heart-disease (Cleveland)
ionosphere	35	351	351	
iris.1	5	150	150	Versicolor vs. others
iris.2	5	150	150	Virginica vs. others
new-thyroid	6	215	215	normal vs. others
pima	9	768	768	pima-indians-diabetes
tic-tac-toe	10	958	958	
wpbc	33	194	198	Wisconsin breast-cancer database

used by Chinneck [48] for testing his heuristic for MAX FS/MIN IIS COVER. Below, we compare our results to the results he obtained.

In our context, we are not interested in the performance of the above linear classification approach compared to other classification methods; this is investigated elsewhere, see, e.g., Bennett and Bredensteiner [24], and Bennett and Mangasarian [25]. Hence, we use all data sets of these problems as points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m$ – in contrast to using part of the points to find the hyperplane and the other part to test whether these points are correctly classified. This is also the case in [48].

Table 5.15 gives the characteristics of the chosen problems. For some problems we had to remove incomplete data sets. The dimension of the space is denoted by n, as above, and gives the number of attributes for each data set. A complete description of the particular contexts in which these instances arise is available at the UCI Repository.

In Tables 5.16 and 5.17 the results of the branch-and-cut implementation for these problems are reported. In Table 5.16 only method (2) of Section 5.1.3 to separate IIS-inequalities is used. The primal heuristic is called every 50 nodes of the tree. We also use the usual abbreviations (see Table 5.1 on page 119). In Table 5.17 additionally the cuts of Balas and Ng (Section 5.2.1) and method (3) of Section 5.1.3 are used. The primal heuristic is called every five nodes of the tree. Furthermore, "tailing off" control

Table 5.16: Results of the branch-and-cut algorithm on *machine learning problems*. The only cutting planes used are IIS-inequalities, separated by method (2) of Section 5.1.3. Additionally, the primal heuristic is run every 50 nodes. The last two columns give lower ("lbd") and upper ("ps") bounds at termination, i.e., the first gives a dual bound and the second a primal solution. Not all problems could be solved to optimality, in which case we only have values in the last two columns.

name	sub	LP	D	CPU	sep	IIS	sol	lbd	ps
breast-cancer	105	455	11	30	18	390	11	11	11
bupa								49	87
echo	17	75	7	0	0	115	6	6	6
glass								26	36
heart								19	31
ionosphere	2639	9684	27	1864	1671	1895	6	6	6
iris.1	291	1845	19	124	73	1013	25	25	25
iris.2	1	2	1	0	0	52	1	1	1
new-thyroid	17	92	7	3	1	428	11	11	11
pima								62	163
tic-tac-toe								65	92
wpbc								7	16

Table 5.17: Results of the branch-and-cut algorithm on *machine learning problems*. IIS-inequalities are separated by *methods (2) and (3)* of Section 5.1.3. Additionally, the cuts of *Balas and Ng* (see Section 5.2.1) are used. The primal heuristic is run every 5 nodes. Tailing off control was on. The last two columns give lower ("lbd") and upper ("ps") bounds at termination, i.e., the first gives a dual bound and the second a primal solution. Not all problems could be solved to optimality.

name	sub	LP	D	CPU	sep	IIS	sol	lbd	ps
breast-cancer	79	733	12	211	193	399	11	11	11
bupa								49	85
echo	11	98	5	2	1	89	6	6	6
glass								23	36
heart								16	29
ionosphere								4	6
iris.1	365	2593	20	564	503	1151	25	25	25
iris.2	1	2	1	0	0	52	1	1	1
new-thyroid	13	106	6	21	20	458	11	11	11
pima								62	157
tic-tac-toe								64	94
wpbc								5	15

Table 5.18: Best results for the *machine learning problems*. The values give the fraction of correctly classified points in percent. The column labeled "Chinneck" gives the best results obtained by Chinneck [48]. Column "MISMIN" lists results of an algorithm by Bennett and Bredensteiner [24], as they are given in [48]. Column "B&C" provides the best primal solution of Tables 5.16 and 5.17, where values that are proved to be optimal are marked with "⋆".

Name	Chinneck	MISMIN	B&C
breast-cancer	98.4	98.2	98.4 ⋆
bupa	75.9	73.9	75.4
echo			90.2 ⋆
glass	81.8	76.6	83.2
heart			90.2
ionosphere	98.3	98.3	98.3 ⋆
iris.1	83.3	82.0	83.3 ⋆
iris.2	99.3	99.3	99.3 ⋆
new-thyroid	94.9	93.5	94.9 ⋆
pima	80.6	80.5	79.6
tic-tac-toe			90.4
wpbc	94.6	91.2	92.3

was on, i.e., we perform a branching step if the progress in the value of the LP relaxation is not good enough.

Since so far all available solutions for these instances were obtained by heuristics, we are also interested in the best solution available for these problems. The last two columns give the lower bound "lbd" obtained in the branch-and-bound tree and the best primal solution "ps" found during the optimization. To obtain a good lower bound we perform a breadth-first search in the branch-and-cut tree.

With the first variant half of the twelve instances could be solved to optimality. Three of these problems are very easy to solve; only one (`ionosphere`) takes quite long (around 30 minutes). The other half of the problems could not be solved to optimality (within 12 hours of computation time or because of memory constraints). For three of these instances the difference of the final bounds is quite large and an optimal solution seems to be out of reach for the current implementation.

The second variant could not solve `ionosphere` within 12 hours. It also obtained worse lower bounds. It seems that within 12 hours of computation time speed is more important than advanced separation techniques (at least with the current cutting planes). The upper bounds are always better than those of the first variant, except for `tic-tac-toe`, mainly because we called

the primal heuristic every five nodes, instead of every 50 nodes.

Table 5.18 compares the best primal solutions found by our branch-and-cut implementation to the best results obtained by the heuristics of Chinneck [48]. The results are given as the fraction of correctly classified points to the total number of points in percent. We additionally add the results of a heuristic called MISMIN by Bennett and Bredensteiner [24], as they appear in [48]. For the results of our branch-and-cut implementation we take the best primal solution given in Tables 5.16 and 5.17.

The results show that both the heuristics of Chinneck and MISMIN generate very good solutions, where the best solutions found by the heuristics of Chinneck are better (or equal) than the solutions found by MISMIN. Since neither the exact solution nor lower bounds were known previously, this shows the advantage of an exact method like branch-and-cut to provide reliable data to evaluate the performance of heuristics. We also conclude that, compared to the two heuristics, the primal solutions that are generated by the branch-and-cut implementation are relatively good and sometimes better.

Bibliography

[1] C. C. Aggarwal, R. K. Ahuja, J. Hao, and J. B. Orlin, *Diagnosing infeasibilities in network flow problems*, Math. Program. **81**, no. 3 (1998), pp. 263–280. [24]

[2] S. Agmon, *The relaxation method for linear inequalities*, Can. J. Math. **6** (1954), pp. 382–392. [14]

[3] E. Amaldi, *On the complexity of training perceptrons*, in Artificial Neural Networks, T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, eds., Elsevier, Amsterdam, 1991, pp. 55–60. [11]

[4] E. Amaldi, *From Finding Maximum Feasible Subsystems of Linear Systems to Feedforward Neural Network Design*, PhD thesis, Dep. of Mathematics, EPF-Lausanne, 1994. [10, 11, 15]

[5] E. Amaldi and R. Hauser, *Randomized relaxation methods for the maximum feasible subsystem problem*, Tech. Report 2001-90, DEI, Politecnico di Milano, 2001. [15]

[6] E. Amaldi and V. Kann, *The complexity and approximability of finding maximum feasible subsystems of linear relations*, Theor. Comput. Sci. **147**, no. 1–2 (1995), pp. 181–210. [1, 13]

[7] E. Amaldi and V. Kann, *On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems*, Theor. Comput. Sci. **209**, no. 1–2 (1998), pp. 237–260. [13, 23]

[8] E. Amaldi and M. Mattavelli, *The* Min PCS *problem and piecewise linear model estimation*, Discrete Appl. Math. **118** (2002), pp. 115–143. [11]

[9] E. Amaldi, M. E. Pfetsch, and L. E. Trotter, Jr., *On the maximum feasible subsystem problem, IISs, and IIS-hypergraphs*, Math. Program. (2002). To appear. [7, 37]

[10] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, *The hardness of approximate optima in lattices, codes, and systems of linear equations*, J. Comput. Syst. Sci. **54**, no. 2 (1997), pp. 317–331. [13]

[11] D. Avis, *A revised implementation of the reverse search vertex enumeration algorithm*, in Polytopes – Combinatorics and Computation, G. Kalai and G. M. Ziegler, eds., DMV Seminar 29, Birkhäuser, Basel, 2000, pp. 177–198. [124]

[12] D. Avis, `lrs` *home page*. Available at: `http://cgm.cs.mcgill.ca/~avis/C/lrs.html`, 2001. [124]

[13] D. Avis, D. Bremner, and R. Seidel, *How good are convex hull algorithms?*, Comput. Geom. **7**, no. 5–6 (1997), pp. 265–301. [41, 60]

[14] D. Avis and K. Fukuda, *A pivoting algorithm for convex hull and vertex enumeration of arrangements and polyhedra*, Discrete Comput. Geom. **8**, no. 3 (1992), pp. 295–313. [41, 124]

[15] D. Avis and K. Fukuda, *Reverse search for enumeration*, Discrete Appl. Math. **65**, no. 1–3 (1996), pp. 21–46. [44]

[16] E. Balas, *Cutting planes from conditional bounds: a new approach to set covering*, Math. Program. Stud. **12** (1980), pp. 19–36. [116]

[17] E. Balas, S. Ceria, and G. Cornuéjols, *A lift-and-project cutting plane algorithm for mixed* 0-1 *programs*, Math. Program. **58**, no. 3 (1993), pp. 295–324. [116]

[18] E. Balas, S. Ceria, G. Cornuéjols, and N. Natraj, *Gomory cuts revisited*, Oper. Res. Lett. **19**, no. 1 (1996), pp. 1–9. [116, 117, 118]

[19] E. Balas and A. Ho, *Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study*, Math. Program. Stud. **12** (1980), pp. 37–60. [116]

[20] E. Balas and S. M. Ng, *On the set covering polytope: I. All the facets with coefficients in* $\{0, 1, 2\}$, Math. Program. **43**, no. 1 (1989), pp. 57–69. [116]

[21] M. O. Ball and J. S. Provan, *Disjoint products and efficient computation of reliability*, Oper. Res. **36**, no. 5 (1988), pp. 703–715. [49]

[22] D. Barnette, P. Kleinschmidt, and C. W. Lee, *An upper bound theorem for polytope pairs*, Math. Oper. Res. **11**, no. 3 (1986), pp. 451–464. [65]

[23] D. Bayer and M. Stillman, *Computation of hilbert functions*, J. Symb. Comput. **14**, no. 1 (1992), pp. 31–50. [81]

[24] K. P. Bennett and E. J. Bredensteiner, *A parametric optimization method for machine learning*, INFORMS J. Comput. **9**, no. 3 (1997), pp. 311–318. [11, 14, 140, 142, 143]

[25] K. P. Bennett and O. L. Mangasarian, *Neural network training via linear programming*, in Advances in optimization and parallel computing, P. M. Pardalos, ed., North-Holland, Amsterdam, 1992, pp. 56–67. [11, 14, 140]

[26] C. Berge, *Hypergraphs. Combinatorics of finite sets*, North-Holland Mathematical Library 45, North-Holland, Amsterdam, 1989. [37, 50]

[27] L. J. Billera and C. W. Lee, *The number of faces of polytope pairs and unbounded polyhedra*, Eur. J. Comb. **2**, no. 4 (1981), pp. 307–322. [65]

[28] J. C. BIOCH AND T. IBARAKI, *Complexity of identification and dualization of positive boolean functions*, Inform. and Comput. **123**, no. 1 (1995), pp. 50–63. [46]

[29] A. BJÖRNER, *Topological methods*, in Handbook of Combinatorics, Vol. II, R. L. Graham, M. Grötschel, and L. Lovász, eds., Elsevier, Amsterdam, 1995, ch. 34, pp. 1819–1872. [5, 46, 75, 77, 79, 89]

[30] A. BJÖRNER, M. LAS VERGNAS, B. STURMFELS, N. WHITE, AND G. M. ZIEGLER, *Oriented Matroids*, Encyclopedia of Mathematics and its Applications 46, Cambridge University Press, Cambridge, 2nd ed., 1999. [38, 54, 104]

[31] A. BJÖRNER AND M. L. WACHS, *Shellable nonpure complexes and posets. I*, Trans. Am. Math. Soc. **348**, no. 4 (1996), pp. 1299–1327. [47]

[32] C. L. BLAKE AND C. J. MERZ, *UCI repository of machine learning databases*, 1998. Available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`. [139]

[33] L. BLUM, F. CUCKER, M. SHUB, AND S. SMALE, *Complexity and Real Computation*, Springer-Verlag, New York, 1997. [54]

[34] J. BOKOWSKI AND B. STURMFELS, *Computational Synthetic Geometry*, Lecture Notes in Mathematics 1355, Springer-Verlag, Berlin, 1989. [52, 54]

[35] R. BORNDÖRFER, *Aspects of Set Packing, Partitioning, and Covering*, PhD thesis, TU Berlin, 1998. [25, 116]

[36] E. BOROS, Y. CRAMA, O. EKIN, P. L. HAMMER, T. IBARAKI, AND A. KOGAN, *Boolean normal forms, shellability, and reliability computations*, SIAM J. Discrete Math. **13**, no. 2 (2000), pp. 212–226. [48]

[37] D. BREMNER, K. FUKUDA, AND A. MARZETTA, *Primal-dual methods for vertex and facet enumeration*, Discrete Comput. Geom. **20**, no. 3 (1998), pp. 333–357. [41]

[38] H. BRUGGESSER AND P. MANI, *Shellable decompositions of cells and spheres*, Math. Scand. **29** (1971), pp. 197–205. [49]

[39] A. CAPRARA AND M. FISCHETTI, *Branch-and-cut algorithms*, in Annotated Bibliographies in Combinatorial Optimization, M. Dell'Amico, F. Maffioli, and S. Martello, eds., John Wiley & Sons, Chichester, 1997, ch. 4, pp. 45–63. [109]

[40] S. CERIA, P. NOBILI, AND A. SASSANO, *Set covering problem*, in Annotated Bibliographies in Combinatorial Optimization, M. Dell'Amico, F. Maffioli, and S. Martello, eds., John Wiley & Sons, Chichester, 1997, ch. 23, pp. 415–428. [25, 115]

[41] N. CHAKRAVARTI, *Some results concerning post-infeasibility analysis*, Eur. J. Oper. Res. **73** (1994), pp. 139–143. [1, 8, 12, 41]

[42] T. M. CHAN, *Output-sensitive results on convex hulls, extreme points, and related problems*, Discrete Comput. Geom. **16**, no. 4 (1996), pp. 369–387. [41]

[43] B. CHAZELLE, *An optimal convex hull algorithm in any fixed dimension*, Discrete Comput. Geom. **10**, no. 4 (1993), pp. 377–409. [41]

[44] J. W. CHINNECK, *Computer codes for the analysis of infeasible linear programs*, J. Oper. Res. Soc. **47**, no. 1 (1996), pp. 61–72. [14]

[45] J. W. CHINNECK, *An effective polynomial-time heuristic for the minimum-cardinality IIS set-covering problem*, Ann. Math. Artif. Intell. **17**, no. 1–2 (1996), pp. 127–144. [14]

[46] J. W. CHINNECK, *Feasibility and viability*, in Advances in Sensitivity Analysis and Parametric Programming, T. Gál and H. J. Greenberg, eds., Int. Ser. Oper. Res. Manag. Sci. 6, Kluwer Academic Publishers, Dordrecht, 1997, ch. 14, pp. 1–41. [10]

[47] J. W. CHINNECK, *Finding a useful subset of constraints for analysis in an infeasible linear program*, INFORMS J. Comput. **9**, no. 2 (1997), pp. 164–174. [14]

[48] J. W. CHINNECK, *Fast heuristics for the maximum feasible subsystem problem*, INFORMS Journal on Computing **13**, no. 3 (2001), pp. 210–223. [14, 120, 140, 142, 143]

[49] J. W. CHINNECK AND E. W. DRAVNIEKS, *Locating minimal infeasible constraint sets in linear programs*, ORSA J. Comput. **3**, no. 2 (1991), pp. 157–168. [14, 22]

[50] T. H. CORMEN, C. E. LEISERSON, R. L. RIVEST, AND C. STEIN, *Introduction to Algorithms*, MIT Press, Cambridge, 2nd ed., 2001. [93, 97, 100]

[51] G. CORNUÉJOLS, *Combinatorial optimization. Packing and covering*, CBMS-NSF Regional Conference Series in Applied Mathematics 74, SIAM, Philadelphia, 2001. [25, 55]

[52] G. CORNUÉJOLS AND A. SASSANO, *On the* 0*,* 1 *facets of the set covering polytope*, Math. Program. **43**, no. 1 (1989), pp. 45–55. [27]

[53] M. E. DYER, *The complexity of vertex enumeration methods*, Math. Oper. Res **8** (1983), pp. 381–402. [124]

[54] T. EITER AND G. GOTTLOB, *Identifying the minimal transversals of a hypergraph and related problems*, SIAM J. Comput. **24**, no. 6 (1995), pp. 1278–1304. [45, 46]

[55] K. FAN, *On systems of linear inequalities*, in Linear Inequalities and Related Systems, H. W. Kuhn and A. W. Tucker, eds., Ann. Math. Studies 38, Princeton University Press, 1956, pp. 99–156. [16]

[56] L. FINSCHI, *A Graph Theoretical Approach for Reconstruction and Generation of Oriented Matroids*, PhD thesis, IFOR, ETH Zürich, 2001. [105]

[57] M. L. FREDMAN AND L. KHACHIYAN, *On the complexity of dualization of monotone disjunctive normal forms*, J. Algorithms **21**, no. 3 (1996), pp. 618–628. [46]

[58] K. FUKUDA, T. M. LIEBLING, AND F. MARGOT, *Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron*, Comput. Geom. **8**, no. 1 (1997), pp. 1–12. [92]

[59] K. FUKUDA AND V. ROSTA, *Combinatorial face enumeration in convex polytopes*, Comput. Geom. **4**, no. 4 (1994), pp. 191–198. [92, 93]

[60] K. FUKUDA, S. SAITO, AND A. TAMURA, *Combinatorial face enumeration in arrangements and oriented matroids*, Discrete Appl. Math. **31**, no. 2 (1991), pp. 141–149. [105]

[61] B. GANTER, *Algorithmen zur formalen Begriffsanalyse*, in Beiträge zur Begriffsanalyse, B. Ganter, R. Wille, and K. E. Wolff, eds., B.I. Wissenschaftsverlag, Mannheim, 1987, pp. 241–254. [92]

[62] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979. [5, 80]

[63] E. GAWRILOW AND M. JOSWIG, `polymake`: *a framework for analyzing convex polytopes*, in Polytopes – Combinatorics and Computation, G. Kalai and G. M. Ziegler, eds., DMV Seminar 29, Birkhäuser, Basel, 2000, pp. 43–74. [5, 27, 31, 39, 106, 124]

[64] E. GAWRILOW AND M. JOSWIG, `polymake`: *an approach to modular software design in computational geometry*, in Proceedings of the 17th Annual Symposium on Computational Geometry, ACM, 2001, pp. 222–231. [5, 27, 31, 39, 106, 124]

[65] J. GLEESON AND J. RYAN, *Identifying minimally infeasible subsystems of inequalities*, ORSA J. Comput. **2**, no. 1 (1990), pp. 61–63. [2, 18]

[66] H. J. GREENBERG, *Consistency, redundancy, and implied equalities in linear systems*, Ann. Math. Artif. Intell. **17**, no. 1–2 (1996), pp. 37–83. [19, 40]

[67] H. J. GREENBERG AND F. H. MURPHY, *Approaches to diagnosing infeasible linear programs*, ORSA J. Comput. **3**, no. 3 (1991), pp. 253–261. [10, 14, 22]

[68] R. GREER, *Trees and Hills: Methodology for Maximizing Functions of Systems of Linear Relations*, Annals of Discrete Mathematics 22, North-Holland, Amsterdam, 1984. [11, 13, 45]

[69] M. GRÖTSCHEL, L. LOVÁSZ, AND A. SCHRIJVER, *Geometric Algorithms and Combinatorial Optimization*, Algorithms and Combinatorics 2, Springer-Verlag, Heidelberg, 2nd ed., 1993. [5]

[70] B. GRÜNBAUM, *Convex Polytopes*, Interscience, London, 1967. [38]

[71] V. GURVICH AND L. KHACHIYAN, *On generating the irredundant conjunctive and disjunctive normal forms of monotone boolean functions*, Discrete Appl. Math. **96–97** (1999), pp. 363–373. [41, 45]

[72] D. S. JOHNSON AND F. P. PREPARATA, *The densest hemisphere problem*, Theor. Comput. Sci. **6** (1978), pp. 93–107. [12]

[73] M. JOSWIG, V. KAIBEL, M. E. PFETSCH, AND G. M. ZIEGLER, *Ambiguous incidences of unbounded polyhedra*. Electronic Geometry Models, No. 2000.05.001, available at `http://www.eg-models.de`, 2000. [73]

[74] M. JOSWIG, V. KAIBEL, M. E. PFETSCH, AND G. M. ZIEGLER, *Vertex-facet incidences of unbounded polyhedra*, Advances in Geometry **1**, no. 1 (2001), pp. 23–36. [66, 82]

[75] M. JOSWIG AND G. M. ZIEGLER, *Convex hulls and oracles*. Preprint, 2002. [62, 63]

[76] V. KAIBEL AND M. E. PFETSCH, *Computing the face lattice of a polytope from its vertex-facet incidences*, Comput. Geom. **23**, no. 3 (2002), pp. 281–290. [79, 81, 91]

[77] V. KAIBEL AND M. E. PFETSCH, *Some algorithmic problems in polytope theory*, in Algebra, Geometry, and Software Systems, M. Joswig and N. Takayama, eds., Springer-Verlag, 2003. To appear. [40, 48, 49, 63, 92, 93, 124]

[78] V. KLEE, *Polytope pairs and their relationship to linear programming*, Acta Math. **133** (1974), pp. 1–25. [65]

[79] U. KÜSSNER AND D. TIDHAR, *Combining different translation sources*, in Natural Language Processing - NLP 2000, Second International Conference, Patras, Greece, June 2000, D. N. Christodoulakis, ed., Lecture Notes in Computer Science 1835, Springer-Verlag, 2000, pp. 261–271. [11]

[80] M. LAURENT, *A generalization of antiwebs to independence systems and their canonical facets*, Math. Program., Ser. B **45**, no. 1 (1989), pp. 97–108. [27, 32, 34]

[81] C. W. LEE, *Bounding the numbers of faces of polytope pairs and simple polyhedra*, in Convexity and graph theory, Proc. Conf., Jerusalem 1981, M. Rosenfeld and J. Zaks, eds., Ann. Discrete Math. 20, North-Holland, Amsterdam, 1984, pp. 215–232. [65]

[82] L. LOVÁSZ, *An Algorithmic Theory of Numbers, Graphs and Convexity*, CBMS-NSF Regional Conference Series in Applied Mathematics 50, SIAM, Philadelphia, 1986. [38]

[83] O. L. MANGASARIAN, *Misclassification minimization*, J. Glob. Optim. **5**, no. 4 (1994), pp. 309–323. [11, 14]

[84] O. L. MANGASARIAN, *Minimum-support solutions of polyhedral concave programs*, Optimization **45**, no. 1–4 (1999), pp. 149–162. [14]

[85] P. MCMULLEN, *The maximum numbers of faces of a convex polytope*, Mathematika **17** (1970), pp. 179–184. [41]

[86] B. MISHRA, *Computational real algebraic geometry*, in Handbook of Discrete and Computational Geometry, J. Goodman and J. O'Rouke, eds., CRC Press, Boca Raton, 1997, ch. 29, pp. 537–556. [54]

[87] N. E. MNËV, *The universality theorems on the classification problem of configuration varieties and convex polytopes varieties*, in Topology and Geometry – Rohlin Seminar, O. Y. Viro, ed., Lecture Notes in Mathematics 1346, Springer-Verlag, Berlin, 1988, pp. 527–543. [54]

[88] T. S. MOTZKIN, *Beiträge zur Theorie der Linearen Ungleichungen*, PhD thesis, Basel, 1933. English version: "Contributions to the Theory of Linear Inequalities", translated by D. R. Fulkerson, in "Theodore S. Motzkin: Selected Papers", D. Cantor, B. Gordon and B. Rothschild, Eds., Birkhäuser, Boston (1983). [15, 16]

[89] T. S. MOTZKIN AND I. J. SCHOENBERG, *The relaxation method for linear inequalities*, Can. J. Math. **6** (1954), pp. 393–404. [15]

[90] G. L. NEMHAUSER AND L. A. WOLSEY, *Integer and Combinatorial Optimization*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, New York, 1988. [5, 19, 109, 117]

[91] P. NOBILI AND A. SASSANO, *A separation routine for the set covering polytope*, in Integer programming and combinatorial optimization, 2nd international IPCO Conference, Pittsburgh, E. Balas, G. Cornuéjols, and R. Kannan, eds., 1992, pp. 201–219. [116]

[92] M. PADBERG AND G. RINALDI, *A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems*, SIAM Rev. **33**, no. 1 (1991), pp. 60–100. [109]

[93] C. H. PAPADIMITRIOU, *Computational Complexity*, Addison-Wesley, Amsterdam, 1994. [5]

[94] M. PARKER, *A Set Covering Approach to Infeasibility Analysis of Linear Programming Problems and Related Issues*, PhD thesis, Department of Mathematics, University of Colorado at Denver, 1995. [2, 11, 13, 14, 19, 28, 29, 111, 120]

[95] M. PARKER AND J. RYAN, *Finding the minimum weight IIS cover of an infeasible system of linear inequalities*, Ann. Math. Artif. Intell. **17**, no. 1–2 (1996), pp. 107–126. [1, 10, 13, 19, 23, 110, 120]

[96] M. PFETSCH, *Examples of generalized antiweb facets.* Electronic Geometry Models, No. 2000.09.029, available at `http://www.eg-models.de`, 2000. [35]

[97] K. POLTHIER, *Mathematical visualization and online experiments with JavaView*, in Mathematical Visulization online, M. Emmer, ed., Matematica e Cultura 3, Springer Verlag, 2000. [5]

[98] K. POLTHIER, S. KHADEM-AL-CHARIEH, E. PREUSS, AND U. REITEBUCH, *JavaView – 3D Geometry in Web Pages*. `http://www.javaview.de/`, 2000. [5]

[99] J. RICHTER-GEBERT, *Realization Spaces of Polytopes*, Lecture Notes in Mathematics 1643, Springer-Verlag, Berlin; Heidelberg, 1996. [38, 54, 61, 68]

[100] N. ROBERTSON AND P. D. SEYMOUR, *Graph minors. XIII. The disjoint paths problem*, J. Comb. Theory, Ser. B **63**, no. 1 (1995), pp. 65–110. [63]

[101] F. ROSSI, A. SASSANO, AND S. SMRIGLIO, *Models and algorithms for terrestrial digital broadcasting*, Annals of Operations Research: Mathematics of Industrial Systems IV. (2001). To appear. [10]

[102] G.-C. ROTA, *On the foundations of combinatorial theory. I: Theory of Möbius functions*, Z. Wahrscheinlichkeitstheorie **2** (1964), pp. 340–368. [77, 78]

[103] J. RYAN, *Transversals of IIS-hypergraphs*, in Proc. 22nd Southeast Conf. on Combinatorics, Graph Theory, and Computing, Baton Rouge, Congr. Numerantium 81, 1991, pp. 17–22. [40, 48, 114]

[104] J. RYAN, *IIS-hypergraphs*, SIAM J. Discrete Math. **9**, no. 4 (1996), pp. 643–653. [40, 46, 49, 51]

[105] J. K. SANKARAN, *A note on resolving infeasibility in linear programs by constraint relaxation*, Oper. Res. Letters **13** (1993), pp. 19–20. [13, 45]

[106] A. SASSANO, *On the facial structure of the set covering polytope*, Math. Program. **44**, no. 2 (1989), pp. 181–202. [27]

[107] A. SCHRIJVER, *Theory of Linear and Integer Programming*, John Wiley & Sons, Chichester, 1986. [5, 15, 19, 21, 54, 116, 117]

[108] B. SCHWIKOWSKI AND E. SPECKENMEYER, *On enumerating all minimal solutions of feedback problems*, Discrete Appl. Math. **117**, no. 1–3 (2002), pp. 253–265. [45]

[109] R. SEIDEL, *Convex hull computations*, in Handbook of Discrete and Computational Geometry, J. Goodman and J. O'Rouke, eds., CRC Press, Boca Raton, 1997, ch. 19, pp. 361–375. [41]

[110] R. P. STANLEY, *Combinatorics and Commutative Algebra*, Progress in Mathematics 41, Birkhäuser, Boston, 2nd ed., 1996. [45]

[111] R. P. STANLEY, *Enumerative Combinatorics, Vol. 1*, Cambridge Studies in Advanced Mathematics 49, Cambridge University Press, 2nd ed., 1997. [77, 78]

[112] S. THIENEL, *ABACUS – A Branch-And-CUt System*, PhD thesis, Universität zu Köln, 1995. [109, 118]

[113] J. N. M. VAN LOON, *Irreducibly inconsistent systems of linear inequalities*, Eur. J. Oper. Res. **8** (1981), pp. 282–288. [16]

[114] M. WAGNER, J. MELLER, AND R. ELBER, *Large-scale linear programming techniques for the design of protein folding potentials*, Tech. Report TR-2002-02, Old Dominion University, 2002. [12]

[115] G. M. ZIEGLER, *Lectures on Polytopes*, Springer-Verlag, New York, 1995. Revised edition 1998. [5, 47, 49, 52, 66, 68, 69, 81]

Index