On the complexity of vertex and facet enumeration for convex polytopes

by

David D. Bremner

School of Computer Science
McGill University
Montréal, Canada

July, 1997

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

Every convex polytope is both the intersection of a finite set of halfspaces and the convex hull of a finite vertex set. Transforming from the halfspaces (vertices, respectively) to the vertices (halfspaces, respectively) is called *vertex enumeration* (*facet enumeration*, respectively). It is an open problem whether there is an algorithm for these two problems polynomial in the input and the output size. For each of the known methods, this thesis develops a characterization of what constitutes an easy or difficult input. Example families of polytopes are presented that show that none of the known methods will yield a polynomial algorithm. On the other hand, a family of polytopes difficult for one class of algorithms can (sometimes) be easily solvable for another class of algorithms; the characterizations given here can be used to guide a choice of algorithms. Similarly, although the general problems of vertex and facet enumeration are equivalent by the duality of convex polytopes, for fixed polytope family and algorithm, one of these directions can be much easier than the other. This thesis presents a new class of algorithms that use the easy direction as an oracle to solve the seemingly difficult direction.

Résumé

Tout polytope convexe est l'intersection d'un ensemble fini de demi-espaces. C'est également l'enveloppe convexe d'un ensemble fini de sommets. La transformation de l'ensemble de demi-espaces en l'ensemble de sommets correspondant s'appelle *l'énumération des sommets*. La transformation inverse s'appelle *l'énumération des facettes*. Pour ces deux transformations duales, le problème de l'existence d'un algorithme, de complexité polynomiale relativement à la taille de l'entrée et de la sortie, est toujours ouvert. Cependant, certains algorithmes peuvent s'avérer de complexité polynomiale pour certaines familles d'entrée. Ceci définit une notion de complexité dépendante, de l'algorithme, de la nature de l'entrée ainsi que de la taille de l'entrée et de la sortie. Nous présentons, en premier lieu, une caractérisation de ce qui constitue des données d'entrée faciles ou difficiles, ceci pour chacune des méthodes d'énumération des sommets et des facettes. Cette caractérisation produit une aide au choix de l'algorithme adéquat, s'il existe, et permet également de démontrer qu'aucune des méthodes connues ne peut conduire à un algorithme polynomial. D'autre part, nous présentons un nouvel algorithme générique qui, pour tout algorithme d'énumération des sommets (ou des facettes), calcule la transformation inverse avec une complexité similaire (i.e. polynomiale ou non), dépendante de la famille de polytopes considérée. Cette nouvelle classe d'algorithmes permet de résoudre efficacement le problème de l'énumeration de sommets (ou des facettes) s'il existe un algorithme pour lequel la transformation inverse est aisée.

Preface

I would like to thank my adviser David Avis for introducing me to the beautiful topic of convex polyhedra, for having a keen sense of what is important, and for many hours of reading, writing, and discussions.

I wish to gratefully acknowledge the scientific collaboration instrumental to this thesis. Sections 3.3 and 4.2 are based on joint work with David Avis. Sections 3.1, 3.2, and 4.3 are joint work with David Avis and Raimund Seidel. Chapter 5 is joint work with Komei Fukuda and Ambros Marzetta. I would like to thank Günter Rote for a careful reading of the thesis and several helpful suggestions and corrections. Any remaining failings of fact or esthetics are of course my own.

The experimental results in this thesis would not have been possible without the implementations of various algorithms due to David Avis, Brad Barber, Ken Clarkson, Thomas Christof and Andreas Loebel, Komei Fukuda, and Ambros Marzetta.

Thanks to the gang at McGill, including but not only Jit, Godfried, Luc, Sue, Suneeta, Steve, and Sylvain for various contributions to and distractions from my life as a graduate student. Thanks especially to Sylvain Lazard for giving Molière back his tongue.

I am grateful for the financial support received from NSERC, from FCAR, and from McGill University.

Finally, thanks to Cheryl for support and encouragement of many kinds.

With the exception of Section 3.2.1 and Section 4.1 which are more expository in nature, Chapters 3, 4, and 5 constitute original contributions to knowledge.

Contents

Chapter 1

Introduction

A *convex polyhedron* is the intersection of a finite number of halfspaces $\mathcal{H}$. A bounded convex polyhedron is called a *polytope*. An *extreme point* of a convex polytope P is some point in P that does not lie on an open line segment between two other points in P. The *convex hull* $\text{conv}(X)$ of a set of points X denotes the intersection of all halfspaces that contain X. A classical theorem from convexity is that every polytope P can be expressed as the convex hull of its extreme points (or vertices) $\mathcal{V}$. These descriptions of P will be referred to as the *halfspace* and *vertex* descriptions, respectively. Converting from the halfspace representation to the vertex representation is called *vertex enumeration*. Converting from the vertex representation to the halfspace representation is called *facet enumeration* or *convex hull*. These problems are essentially equivalent by duality of convex polytopes. Although methods for solving this problem were sketched by Fourier [22] in the early 19th century, the computational complexity of vertex/facet enumeration remains open. Indeed, until recently it was not known if the general methods due to Fourier could result in an algorithm polynomial in $|\mathcal{V}|$, $|\mathcal{H}|$, and the dimension d (in the rest of this thesis we call such an algorithm simply *polynomial*). In this thesis we answer the latter question in the negative. Known methods of vertex/facet enumeration can be broadly grouped into those based on the simplex pivot operation (along with some method of dealing with degeneracy) and those based on incremental construction (i.e. the double description method of Motzkin [34]). For each of these general classes of algorithms, we give "difficult" families of polytopes for which the corresponding algorithms are superpolynomial. Several of our families are "universal" i.e. difficult for all of the main known types of algorithms.

For any given instance of vertex/facet enumeration, there is a corresponding dual instance of transforming the output back to the input. In this thesis we argue that for certain hereditary families of polytopes (i.e. those where every subset of the output for a given polytope is in the family) the complexity of these two transformations is polynomially equivalent (note that this is quite distinct from the dual interpretation of a vertex enumeration problem as facet enumeration problem). We then show how to refine our constructive proof into an efficient and practical algorithm for vertex (resp. facet) enumeration of simplicial (resp. simple) polytopes.

The rest of this chapter is organized as follows. In Section 1.1, we discuss theoretical and practical motivations for the study of vertex/facet enumeration. In Section 1.2, we discuss previous work on this topic, and point out some connections to this thesis. In Section 1.3 we provide an overview of the remainder of the thesis.

1.1 Motivation

The problem of converting between the two representations of polytopes is fundamental from a theoretical point of view. Indeed several of the proofs that every polytope has both halfspace and a vertex representation (see e.g. Minkowski [105], Motzkin [31], or Ziegler [112]) proceed by showing the existence of an algorithm to convert between these two representations. From an algorithmic point of view, the existence (or non-existence) of a polynomial vertex/facet enumeration algorithm would have important implications for the tractability of several arbitrary dimensional geometric problems such as Voronoi diagrams [55, 89] and Power Diagrams [51].

Vertex/facet enumeration is also an important practical problem. Much of the interest in convex polytopes has stemmed from their use in mathematical modeling.

Vertex enumeration is typically applied when the situation to be modeled can be characterized (or approximated) by a set of linear constraints, but the objective function (or measure of solution "goodness") is either non-linear or linear but unknown. An illustrative example is the work of Ceder et al. [56] where the authors model the minimum energy state of an alloy lattice by a linear program with an unknown objective function. From the Fundamental Theorem of Linear Programming (see e.g. [85]), the optimal solution to this linear program must occur at a vertex. By enumerating the vertices and checking each candidate solution for physical reasonableness, they are able to find the global solution. Other applications include combinatorial optimization [53, 59, 60, 58, 61, 64], game theory [34, 52], multi-objective linear programming [63], quantum chemistry [54], and robotics [65].

1.2 History

Known algorithms for vertex and facet enumeration can be broadly divided into *insertion* or *incremental* algorithms and *pivoting* algorithms. Insertion algorithms (for vertex enumeration) are based on the observation that to compute the intersection of a set of halfspaces $\mathcal{H} = \{H_1, H_2, \ldots H_{m-1}, H_m\}$, it suffices to inductively compute $P_{m-1} = \bigcap_{i=1}^{m-1} H_i$ and then compute $P_{m-1} \cap H_m$. Pivoting algorithms are based on the simplex method of linear programming (see e.g. [85]) which provides a method to find all of the vertices adjacent to a given vertex of a polyhedron.

1.2.1 Insertion Algorithms

Research in vertex enumeration predates the discipline of computer science by about a century. In his 1824 paper on linear inequalities, Fourier [22] proposes a method for finding extreme solutions of systems of linear inequalities based on successively eliminating variables. Minkowski [105] made the important observation that every extreme point (vertex) of a d-polytope defined by halfspaces $\mathcal{H}$ satisfies a set of d affinely independent constraints from $\mathcal{H}$ with equality. Based on this he noted that all vertices could be found by simply testing all affinely independent d-sets of constraints. Stokes [38] gives an early exposition of point-hyperplane duality, and rediscovers Minkowski's algorithm in the dual setting of facet enumeration. None of these algorithms are practical for computation without modification.

The first practical algorithm for vertex enumeration was developed by Motzkin in his doctoral dissertation [31] and refined in [34]. Motzkin's *double description method* is roughly the following[1]. At each step we maintain both a vertex and halfspace description of a current intermediate polytope (hence the name "double description method"). Initially, choose some set of $d+1$ halfspaces forming a d-simplex. Compute the vertices and edges of their intersection. At each succeeding step, compute the intersection of the current intermediate polytope and one of the remaining halfspaces. The vertices of the new intermediate polytope are those of the old polytope feasible for the new halfspace plus the intersections of edges of the old polytope and the bounding hyperplane for the new halfspace. Before this can be termed an algorithm, several details must be specified. In particular, the order in which the halfspaces are inserted (or *insertion order*) can make the difference between whether the double description method is polynomial or not on a particular family of polytopes. It was later observed that Motzkin's method is dual to that of Fourier (see Section 4.1 for discussion). What makes Motzkin's method computationally useful, while in general the original method proposed by Fourier is not, is that the double description method only generates and keeps the intersections of edges with the bounding hyperplane of the inserted halfspace, while Fourier's method additionally (in the language of the double description method) generates and keeps redundant points that are the intersection of a non-extreme line segment between two vertices of the previous intermediate polytope with the bounding hyperplane of the new halfspace.

The double description method was rediscovered by (among others) Burger [9] and Chernikova [14]. Fourier's elimination method was rediscovered by Dines [18].

[1] In order to work for unbounded intermediate polytopes the method needs to lift the polytope into a homogeneous cone in one higher dimension. For details see Section 4.1.

Available implementations of the double description method include [41, 44, 47, 50].

In the computer science literature, the first work on convex hulls in dimensions higher than three seems to be that of Seidel [36], who rediscovered and refined Motzkin's algorithm in the dual setting of facet enumeration. In the *beneath and beyond method* instead of adding halfspaces one at at time, points are added one at time with the convex hull of the added points maintained at every step. From the upper bound theorem of McMullen [102], it is known that for fixed d, the the convex hull of n points in $\mathbb{R}^d$ has $O(n^{\lfloor d/2 \rfloor})$ facets, and this bound is achieved by the cyclic polytopes. In order to match this bound in even d, Seidel (and most of the computer science literature that follows him) makes the assumption that the input points are in *general position*, i.e. that no $d+1$ of them are contained in a hyperplane. Such an assumption can be removed by either perturbing the input set or by (essentially equivalently) maintaining a triangulation of the intermediate polytopes. While neither perturbation nor triangulation affects the bound of $O(n^{\lfloor d/2 \rfloor})$, we shall see that in terms of polynomiality in the output size, it makes a huge difference, since the perturbed or triangulated output is generally much larger than the the actual output.

In 1983, Dyer [19] used a construction of Klee [99] to show that on-line incremental algorithms (i.e. those that insert the input constraints in the order given) are superpolynomial in the worst case. Chapter 4 of this thesis generalizes and extends this work in several ways.

An incremental algorithm that meets the worst case bound of $O(n^{\lfloor d/2 \rfloor})$ for all fixed d (as opposed to just even d) was provided (in an expected sense) by Clarkson and Shor [16] and (in a deterministic sense) by Chazelle [13]. Both of these algorithms maintain a triangulation of each intermediate polytope.

The work of Seidel [36], Clarkson and Shor [16], and Chazelle [13] on finding a worst case optimal algorithm can be viewed in part as looking for good insertion orders. In the case of [36], the insertion order is lexicographic. In the case of [13], a central part of the algorithm is a sophisticated insertion order that simulates the random ordering used in [16]. In order for an incremental convex hull algorithm to be polynomial each intermediate polytope must have polynomial size. In fact, it is not hard to see that if this condition holds, then a fairly naive implementation of Motzkin's double description method (with this hypothetical insertion order) will be polynomial. Thus the question "Is there a polynomial incremental convex hull algorithm for polytope family Γ" is equivalent to "Is there a polynomial insertion order for polytope family Γ". One of the results of this thesis is that there are families of polytopes (see Section 4.4) for which there is is no polynomial insertion order.

1.2.2 Pivoting Algorithms

The notion of pivoting also seems to have been first explored by Fourier [22], although not formulated explicitly algebraicly until the late 1930's by Kantorovich (see [97]) and independently by Dantzig in 1951 [86] as the *simplex method*. Consider a d-polytope P defined by a set of halfspaces $\mathcal{H}$. An affinely independent set of d elements of $\mathcal{H}$ whose bounding hyperplanes intersect in some vertex of v of P is called a *basis* for v. Minkowski [105] showed that every vertex has a basis. A *pivot* is the replacement of exactly one element of a basis to yield a new basis. In this manner a *basis graph* of a polytope is naturally defined with nodes corresponding to bases and edges corresponding to pivots. The problem of enumerating the vertices of a polytope can be reduced by the theorem of Minkowski to the problem of traversing this graph. In 1953 Charnes [12] gave an algorithm a using depth first search on the basis graph, which included a perturbation procedure to deal with degeneracy. Mañas and Nedoma [29] gave an algorithm using breadth first search on the basis graph. Altherr [1], Dyer and Proll [20], Dyer [19] and Chvátal [85] gave more refined versions using e.g. balanced trees to store the bases found so far rather than unordered lists. Avis and Fukuda [5] described the *reverse search* algorithm based on the depth first traversal of the basis graph that has the significant advantage that it does not need to store any of the bases previously discovered. Pivoting can also be used for the problem of facet enumeration, where a basis is a set of d affinely independent vertices lying in a facet. Dual pivoting or *gift wrapping* algorithms for facet enumeration where given by Chand and Kapur [11], Swart [39], and Seidel [37]. Rote [35] gave a gift wrapping algorithm based the reverse search technique of Avis and Fukuda interpreted in the dual.

Polytopes where every vertex has exactly one basis are called *simple*. Polytopes where every facet has exactly one basis are called *simplicial*. Where every item to be enumerated (vertex or facet) has exactly one basis, pivoting algorithms are polynomial and quite fast in practice. In the general case there can be (superpolynomially) many bases corresponding to a single vertex. The most popular technique for dealing with this problem in practice is to perturb the input to reduce the number of bases of the polytope (see e.g. [74, 46, 75]). An alternative approach, taken in [11, 39, 35], is to recursively compute (in the context of facet enumeration) the facets of each facet. Dyer [19] previously observed that such algorithms perform superpolynomially on some families of polytopes. In Chapter 3 we argue that such algorithms are superpolynomial even if they use an optimization due to Swart [39].

1.3 Overview of Thesis

The rest of the thesis is organized as follows. In Chapter 2 we give some definitions and preliminary results from the theory of convex polytopes. In Chapter 3 we argue the general method of pivoting is unlikely to provide a polynomial algorithm for vertex/facet enumeration. In Chapter 4, we show that incremental algorithms cannot be polynomial in our sense. In Chapter 5 we introduce a new "primal-dual" method of facet/vertex enumeration that is polynomial for a natural and non-trivial class of input. In Chapter 6 we present some conclusions and directions for future work.

Chapter 2

Preliminaries

This chapter presents some definitions and fundamental results from the theory of convex polytopes that will be useful in the sequel. Several of the results belong to the mathematical folklore; for completeness the easy proofs are included here.

2.1 Convex Polytopes

This section introduces some fundamental notions of convex polytopes from a combinatorial point of view. We start by defining polyhedra and polytopes, and consider the notions of the face lattice, duality, and redundancy. For general references on convex polytopes, and terms not defined here, see [92, 84, 112].

This thesis is primarily concerned with the affine properties of the d-dimensional real vector space $\mathbb{R}^d$, $d \geq 1$. We shall occasionally restrict ourselves to the the d-dimensional rational vector space $\mathbb{Q}^d$. A *hyperplane* is a set of points satisfying some linear inequality $ax = b$. Similarly, a *halfspace* is the set of points satisfying some linear inequality $ax < b$ (i.e. an open halfspace) or $ax \leq b$ (i.e. a closed halfspace). A convex polyhedron is the set of solutions to a system of linear inequalities $Ax \leq b$, or equivalently the intersection of a set of closed halfspaces. A bounded convex polyhedron is called a *polytope*.

Given a set of points $X = \{x_1 \dots x_n\}$, a combination $\sum_{i=1}^{n} \lambda_i x_i$ is called *affine* if $\sum \lambda_i = 1$. A combination is called *nonnegative* if each $\lambda_i \geq 0$, and *convex* if it is both affine and nonnegative. The *affine hull* $\operatorname{aff} X$ is the set of all affine combinations of X, or equivalently the smallest affine subspace containing X. The *dimension* $\dim X$ is the dimension of $\operatorname{aff} X$. A set of k points is called *affinely independent* if it has dimension k. A set of k hyperplanes in $\mathbb{R}^d$ is called affinely independent if its intersection has dimension $d - k$. The *relative interior* $\operatorname{relint} X$ is the interior in $\operatorname{aff} X$. The *convex hull* $\operatorname{conv} X$ is the set of all convex combinations of X. Point p is *extreme* for X if p is not a convex combination of $X \setminus \{p\}$.

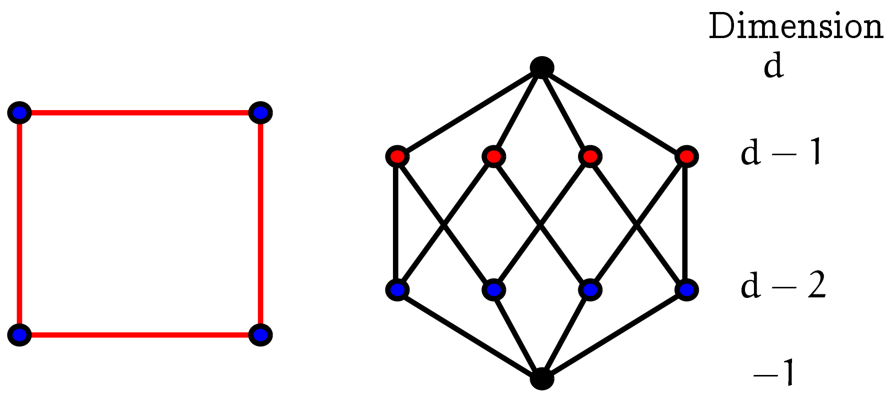
The following can be considered the fundamental theorem of convex polytopes. For an outline of a proof, see Section 4.1. Further details can be found in e.g. Brøndsted [84] or Ziegler [112].

Theorem 2.1 (Minkowski-Weyl) *Every convex polytope is both the intersection of a finite set of halfspaces and the convex hull of its extreme points.*

Given a polytope $P = \bigcap \mathcal{H}$, a halfspace $H \in \mathcal{H}$ is called *redundant* if $P = \bigcap(\mathcal{H} \setminus H)$. Similarly for a polytope P defined as the convex hull of a set X, $x \in X$ is called redundant if $P = \operatorname{conv}(X \setminus x)$ (i.e. x is not an extreme point of $\operatorname{conv} X$). In what follows, unless otherwise noted, we assume that any descriptions of polytopes under discussion contain no redundant elements. This assumption is justified by the fact that redundant elements can be removed by solving a linear number of linear programs, and by the existence of practically and theoretically efficient algorithms for linear programming. $\mathcal{H}(P)$ (respectively $\mathcal{V}(P)$) is the non-redundant halfspace (respectively vertex) description of P. We use m for $|\mathcal{H}(P)|$, n for $|\mathcal{V}(P)|$, and d for the dimension $\dim P$. We use $\mathbb{O}$ ($\mathbb{O}^k$) and $\mathbb{1}$ ($\mathbb{1}^k$) to denote the vector of all zeros (of length k) and all ones (of length k) respectively. Let $\boldsymbol{e}_i$ denote the ith *elementary vector*, i.e. the vector with element i equal to 1 and every other element 0. By convention we define the 0th elementary vector as the origin $\mathbb{O}$.

From a combinatorial point of view, the important information about a polytope is the combinatorial structure of its boundary. A hyperplane H *supports* a polytope P if $H \cap P \neq \emptyset$ and P is contained in one of the closed halfspaces (the *supporting*

Figure 2.1 The face lattice of a square.



halfspace) induced by H. The *faces* of a convex polytope P are $\emptyset$, P (the *improper* faces), and the intersection of a supporting hyperplane of P and P itself (the *proper faces*). Faces of dimension k are called k-faces. By convention we say that the empty face has dimension -1. $f_k(P)$ denotes the number of k-faces of P. Let $\bar{f}(P)$ denote the total number of nonempty faces of P, i.e. $\bar{f}(P) = \sum_{i=0}^{d} f_i(P)$. The names *vertices*, *edges*, *ridges*, and *facets* refer to 0, 1, $(d-2)$, and $(d-1)$-faces respectively. We use $\mathcal{V}(P)$, $\mathcal{E}(P)$, and $\mathcal{F}(P)$ to denote respectively the vertices, edges, and facets of P. The natural graph defined by the vertices and edges of P is called the *skeleton* of P.

The *face lattice* of a polytope is the poset of its faces partially ordered by inclusion. Lattice L_1 is *anti-isomorphic* to lattice L_2 if it is isomorphic with the partial order reversed. Polytopes P and Q are *combinatorially equivalent* (respectively *dual*) if their face lattices are isomorphic (respectively anti-isomorphic). For a face F of d-polytope P, the *codimension* $\operatorname{codim} F$ denotes $d - \dim(F) - 1$. We use $c_k(P)$ to denote the number of faces of P with codimension k. If P' is a polytope dual to P, then we have

$$f_k(P') = c_k(P) \qquad\qquad \bar{f}(P') = \bar{f}(P)$$

Figure 2.1 illustrates the face lattice for a square, which is is easily seen to be self-dual, since the face lattice is symmetric top to bottom.

A d-polytope P is called *simple* if every vertex is the intersection of exactly d-facets. P is called *simplicial* if every facet contains exactly d vertices. From the definition of duality, we see that the simplicial polytopes are dual to the simple polytopes.

We treat sets of points and matrices interchangeably where convenient; the rows of a matrix are the elements of the corresponding set. Where the origin is in the interior of a polytope, each facet defining inequality can be written as $ax \leq 1$ for some a; for facet F we write $\alpha(F)$ to denote such a vector a. For vector h, we adopt the convention that h^+, h^-, and h^0 respectively denote the set of points x such that $hx \leq 1$, $hx > 1$, and $hx = 1$, respectively. Where there is no danger of confusion, we sometimes identify the halfspace h^+ with the corresponding constraint $hx \leq 1$. We use $\mathcal{P}(H)$ to denote the polyhedron $\{\, x \mid Hx \leq \mathbb{1} \,\}$. Similarly we use $\mathcal{H}(P)$ to mean the matrix A where $P = \{\, x \mid Ax \leq \mathbb{1} \,\}$. For a set of points V we use $\mathcal{H}(V)$ to mean $\mathcal{H}(\operatorname{conv} V)$; similarly for a set of halfspaces H, we use $\mathcal{V}(H)$ to mean $\mathcal{V}(\mathcal{P}(H))$. We say that a halfspace h^+ is *valid* (or $hx \leq 1$ is a *valid inequality*) for a set of points X if $X \subseteq h^+$.

For any point set X, the *polar* X^* of X is defined as $\{\, y \mid Xy \leq \mathbb{1} \,\}$. It is known (see e.g. [84]) that if P is a polytope containing the origin in its interior, P^* is a polytope dual to P, containing the origin in its interior.

For a facet F with outward normal h, we write F^+ or h^+ for the corresponding supporting halfspace and F^- or h^- for the other (open) halfspace induced by h.

2.2 Bounds on Face Counts

Many of the lower bounds in this thesis will be based on families of polyhedra with particular bounds on face counts. Here we recall the two most well known such families, and the two fundamental theorems of polytope theory that they realize.

A *cyclic* d-*polytope* $C_d(n)$ on n vertices is the convex hull of n distinct points on the moment curve[1] $c(t) = (t, t^2, \dots t^d)$. It is known (see e.g. [92]) that the convex hull of any n distinct points on the moment curve is combinatorially equivalent. Let $\gamma_i(n, d)$ denote $f_i(C_d(n))$. Let $\gamma(n, d)$ denote $\gamma_{d-1}(n, d)$. Cyclic polytopes are *neighbourly*, i.e. every $k \le \lfloor d/2 \rfloor$ vertices define a face (see e.g. [92]). It follows that

$$\gamma_i(n, d) = \binom{n}{i+1} \qquad 0 \le i < \lfloor d/2 \rfloor$$

The facets of a cyclic polytope are completely characterized by "Gale's evenness condition" [90], as follows. Consider the bit vector corresponding to the set of vertices of $C_d(n)$ lying on some hyperplane h^0. h^0 defines a facet if and only if there are exactly d 1s and every consecutive set of ones is either even in length or contains the first or last vertex. It follows [90, 92, 112] from Gale's evenness condition that

$$\gamma(n, d) = \begin{cases} \dfrac{n}{n - d/2}\dbinom{n - d/2}{n - d} & d \text{ even;} \\ 2\dbinom{n - (d+1)/2}{n - d} & d \text{ odd} \end{cases} \tag{2.1}$$

In fact $\gamma_k(n, d)$ can be computed for the remaining values of k from the "Dehn-Somerville Equations"; see [92],

2.3

9.6 for details and formulae.

Theorem 2.2 (The Upper Bound Theorem [102]) *A convex d polytope with n vertices has at most $\gamma_k(n, d)$ k-faces.*

---

[1] Several other curves will serve equally well; see [92] p. 63 and the references there for more examples.

Let $\bar{\gamma}(n, d)$ denote $\bar{f}(C_d(n))$. Since cyclic polytopes are simplicial, the following (rather crude) bound holds: $\gamma(n, d) \leq \bar{\gamma}(n, d) \leq 2^d \gamma(n, d)$. The following lemma, due to Altshuler and Perles, says that in some sense, the cyclic polytopes are as degenerate as possible.

Lemma 2.1 ([77], p. 102) *For even dimension* d*, every vertex of* $C_d(n)$ *is contained in exactly* $\gamma(n-1, d-1)$ *facets.*

A *truncation polytope* is either a simplex, or the intersection of a truncation polytope with a halfspace that cuts off precisely one vertex. A *stacked polytope* is either a simplex, or the convex hull of a stacked polytope P, along with a point x beyond exactly one facet of P. The following can be proved from polarity.

Proposition 2.1 *The dual of a stacked polytope with* n *vertices is a truncation polytope with* n *facets.*

Let $\beta_k(m, d)$ denote the number of k-faces of a truncation polytope. The following is known (see e.g. [84]):

$$\beta_k(m, d) = \begin{cases} (d-1)m - (d+1)(d-2), & k = 0; \\ \binom{d}{k+1}(m-d) + \binom{d}{k}, & 1 \leq k \leq d-2 \end{cases} \tag{2.2}$$

We use $\beta(m, d)$ to denote $\beta_0(m, d)$.

Theorem 2.3 (The Lower Bound Theorem[78, 79, 82]) *For any simple* d*-polytope* P *with* m *facets,* $f_k(P) \geq \beta_k(m, d)$*. Furthermore, for* $d \geq 4$*, if* $f_k(P) = \beta_k(P)$ *for any* k *then* P *is a truncation polytope.*

Unlike Theorem 2.2, the bound of Theorem 2.3 holds only for simple (or in the dual interpretation simplicial) case. A natural lower bound for the number of vertices of an m facet polytope is provided by "inverting" Theorem 2.2. Define $\lambda(m, d)$ to be the smallest integer l such that $m \leq \gamma(l, d)$. For any m facet d-polytope P,

$$f_0(P) \geq \lambda(m, d) \tag{2.3}$$

Deza [87] has shown that the bound (2.3) is tight for m or d even.

2.4 Products and Sums of Polytopes

Many of the results of this thesis will be based on the *Cartesian products of polytopes* construction. Here we present some well known properties of this construction that will be useful in the sequel. Let P be a k-polytope and let Q be an l-polytope. Let $P \times Q$, called the *product* of P and Q, denote $\{ (p, q) \mid p \in P,\ q \in Q \}$. We regard $\mathbb{R}^k \times \mathbb{R}^l$ as naturally embedded in $\mathbb{R}^{(k+l)}$, with coordinatization $(x_1, \ldots x_k, y_1, \ldots y_l)$ where x_i and y_i are the coordinates of $\mathbb{R}^k$ and $\mathbb{R}^l$ respectively.

Lemma 2.2 *For nonempty subsets S and T of $\mathbb{R}^k$, $\mathbb{R}^l$ respectively*

(a) $\operatorname{aff}(S \times T) = \operatorname{aff} S \times \operatorname{aff} T$

(b) $\operatorname{conv}(S \times T) = \operatorname{conv} S \times \operatorname{conv} T$

(c) $\dim(S \times T) = \dim S + \dim T$

Proof.

(a) Consider an affine combination $(s, t) = \sum_{i \in I, j \in J} \lambda_{ij}(s_i, t_j)$ of $S \times T$. By setting $\sigma_i = \sum_{j \in J} \lambda_{ij}$ and $\tau_j = \sum_{i \in I} \lambda_{ij}$ we obtain affine combinations $s = \sum_{i \in I} \sigma_i s_i$, $t = \sum_{j \in J} \tau_j s_j$. To go the other direction, simply set $\lambda_{ij} = \sigma_i \tau_j$.

(b) Note that both of the mappings between combinations given above preserve non-negativity.

(c) Let $A_1 = x_1 + L_1$ and $A_2 = x_2 + L_2$ be affine subspaces, where L_1 and L_2 are linear subspaces. Let $L'_1 = \{ (x, \mathbb{O}^l) \mid x \in L_1 \}$, and let $L'_2 = \{ (\mathbb{O}^k, x) \mid x \in L_2 \}$.

$$A_1 \times A_2 = (x_1, x_2) + (L_1 \times L_2) = (x1, x2) + (L'_1 + L'_2)$$

A standard result of linear algebra (see e.g. [107]) is that

$$\begin{aligned}\dim(L_1' + L_2') &= \dim(L_1') + \dim(L_2') - \dim(L_1' \cap L_2') \\ &= \dim L_1 + \dim L_2 = \dim A_1 + \dim A_2\end{aligned}$$

□

Lemma 2.3 (The Product Lemma) *Let P be a k-polytope and Q an l-polytope.*

(a) $P \times Q$ *is a* $(k + l)$*-polytope.*

(b) *For* $i \geq j \geq 0$*, the* i*-faces of* $P \times Q$ *are precisely* $F_p \times F_q$ *where* F_p *is a* j*-face of* P *and* F_q *is an* $(i - j)$ *face of* Q*.*

(c) $P \times Q$ *is simple iff* P *and* Q *are simple.*

Proof.

(a) Since P and Q are convex, $P \times Q = \operatorname{conv} P \times \operatorname{conv} Q$. By Lemma 2.2b, $P \times Q = \operatorname{conv}(P \times Q)$. It follows that $P \times Q$ is a polytope. The dimension follows from Lemma 2.2c.

(b) Let R^{k+l} be coordinatized by $(x_1 \ldots x_k, y_1 \ldots y_l)$. Let $ax \leq b$ be a facet defining constraint for facet F of P. The constraint $ax \leq b$ is valid for $P \times Q$. Moreover $\{ (x, y) \in P \times Q \mid ax = b \} = F \times Q$ hence by (2.3a) has dimension $k + l - 1$. It follows that the facet defining inequalities for $P \times Q$ are precisely the the facet defining inequalities for P and Q, interpreted in the higher dimensional space. Any j face must satisfy an affinely independent set of j of these inequalities with equality, and at most k (respectively l) of these can come from P (respectively Q).

(c) Note that a vertex (p, q) only satisfies with equality those facet defining constraints of P and Q that p or q satisfy with equality. □

From Lemma 2.3, we draw some easy conclusions about the face counts of products.

Corollary 2.1 *Let P and Q be k and l-polytopes.*

(a) $f_0(P \times Q) = f_0(P) \cdot f_0(Q)$

(b) $c_0(P \times Q) = c_0(P) + c_0(Q)$

(c) $\bar{f}(P \times Q) = \bar{f}(P) \cdot \bar{f}(Q)$

For polytopes $P \subset \mathbb{R}^k$ and $Q \subset \mathbb{R}^l$, we define the *orthogonal sum* $P \oplus Q$ as

$$P \oplus Q \equiv \operatorname{conv}(\{\,(p, \mathbb{0}^l) \mid p \in P\,\} \cup \{\,(\mathbb{0}^k, q) \mid q \in Q\,\}).$$

Lemma 2.4 *Let* P *and* Q *be* k *and* l *polytopes respectively that contain the origin as an interior point.* $P \oplus Q = (P^* \times Q^*)^*$.

Proof. By Lemma 2.3 and polarity, we know the defining halfspaces for $P^* \times Q^*$ can be written as $\begin{bmatrix} \mathcal{V}(P) & \mathbb{0} \\ \mathbb{0} & \mathcal{V}(Q) \end{bmatrix} x \leq \mathbb{1}$. Taking the polar one more time, we arrive at the definition of $P \oplus Q$. $\square$

We again note some easy but useful consequences for face counts.

Corollary 2.2 *Let* P *and* Q *be* k *and* l *polytopes respectively that contain the origin as an interior point.*

(a) $f_0(P \oplus Q) = f_0(P) + f_0(Q)$

(b) $c_0(P \oplus Q) = c_0(P) \cdot c_0(Q)$

(c) $\bar{f}(P \oplus Q) = \bar{f}(P) \cdot \bar{f}(Q)$

Closely related to the orthogonal sum is the *diagonal sum*. Given k-polytope P and an l-polytope Q, define the *diagonal sum* $P \odot Q$ as

$$P \odot Q \equiv \operatorname{conv}(\{\,(x, -x_k \mathbb{1}^l) \mid x \in P\,\} \cup \{\,(y_1 \mathbb{1}^k, y) \mid y \in Q\,\}).$$

The idea behind the diagonal sum is that P and Q are embedded in the two subspaces

$$-x_k = x_{k+1} = \ldots x_{k+l} \tag{2.4a}$$

$$x_1 = x_2 = \cdots = x_k = x_{k+1} \tag{2.4b}$$

where the subspace defined by (2.4a) intersects the subspace defined by (2.4b) exactly in the point $\mathbb{O}$. It turns out that this is equivalent in a very strong sense to the orthogonal sum construction. Call two polytopes *linearly equivalent* if there is a bijective linear transformation from one to the other.

Lemma 2.5 *Let P and Q be polytopes containing the origin in their interior. $R = P \odot Q$ is linearly equivalent to $R' = P' \oplus Q'$, where P' (respectively Q') is linearly equivalent to P (respectively Q).*

Proof. Let $k = \dim P$, $l = \dim Q$ and $d = k + l$. Let L_1 (L_2 respectively) be the linear subspace defined by (2.4a) ((2.4b) respectively). Let B_1 denote a basis for L_1 (i.e. a set of k linearly independent vectors in L_1). Let B_2 be a basis for L_2. Since $\dim(L_1 \cup L_2) = d$, $B = B_1 \cup B_2$ forms a basis for $\mathbb{R}^d$. The linear transformation B^{-1} transforms B to the identity matrix I. Let $R' = RB^{-1}$ (i.e. the transformation B^{-1} applied to R). Since $L_1 \cap L_2 = \mathbb{O}$, B_1 and B_2 are transformed to disjoint subsets (of rows) of I, hence R' has the desired form $P' \oplus Q'$ (possibly after reordering coordinates). By construction,

$$\mathcal{V}(R) = \mathcal{V}(R')B = \begin{bmatrix} \mathcal{V}(P') & \mathbb{O} \\ \mathbb{O} & \mathcal{V}(Q') \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \end{bmatrix}$$

Let $X = \mathcal{V}(P)$ and $Y = \mathcal{V}(Q)$.

$$\begin{bmatrix} X & -x_{i,k} \\ y_{i,1} & Y \end{bmatrix} = \begin{bmatrix} \mathcal{V}(P')B_1 \\ \mathcal{V}(Q')B_2 \end{bmatrix}$$

It follows that e.g. $P = P'\widehat{B}_1$ where $\widehat{B}_1$ denotes the first k columns of B_1. $\square$

From Lemma 2.5, we can use the diagonal and orthogonal sum interchangeably, depending on which is more convenient. One advantage of the diagonal sum operation is that it preserves the vertex coordinates of the two polytopes, set of the two polytopes. We will use this create a sum of polytopes with coordinates ± 1 that also has coordinates ± 1.

2.5 Complexity

We consider the input (respectively, output) size to be the number of real (or rational) numbers needed to represent the input (respectively output). For convenience, we use $\mathrm{size}(P) \equiv \dim(P)(|\mathcal{V}(P)| + |\mathcal{H}(P)|)$ to denote the summed input and output size.

We assume each single arithmetic operation takes a constant amount of time. This assumption is merely to simplify the discussion, and everything here could be reanalyzed in the binary complexity model without qualitative changes.

All logarithms are binary, unless otherwise specified.

A *family* of polytopes is used here to mean an infinite set of polytopes. Usually, but not necessarily, families arise in some natural way from a problem such as the traveling salesman problem [61], or a construction such as those described below. Given a family of polytopes Γ, a function $g : \Gamma \to \mathbb{R}$ is called *polynomial* for Γ if there exists some univariate polynomial $p(x)$ such that for every $P \in \Gamma$, $g(P) \leq p(\mathrm{size}\, P)$. A function $g : \Gamma \to \mathbb{R}$ is called *weakly polynomial* for Γ if there exists positive function $f(x)$ and polynomial $p(x)$ such that $\forall P \in \Gamma$, $g(P) \leq f(\dim P) \cdot p(\mathrm{size}\, P)$; this corresponds to the notion of considering d to be a constant. If g is not (weakly) polynomial for Γ we say that g is (*strongly*) *superpolynomial* for Γ.

Bibliographic Notes

The numbers $\langle f_0(P), f_1(P), \ldots f_{d-1}(P) \rangle$ are often called the f-*vector* of the polytope P. Characterizing the f-vectors of (various families) of polytopes is a central theme in polytope theory. The Upper Bound Theorem (Theorem 2.2) was first conjectured by Motzkin [106] in 1957 and proved by McMullen [102] in 1970. The f-vectors of simplicial (and by duality simple) polytopes are completely characterized by "McMullen's f-vector conditions". These algebraic conditions were conjectured by McMullen [103, 104] in 1971. The necessity of these conditions was proved by Stanley [110] in 1980. In 1981, the sufficiency of the conditions was shown by Billera and Lee [82]. The Lower Bound Theorem (Theorem 2.3), although first proved by Barnette [78, 79], is also a corollary of McMullen's conditions. Deza and Fukuda [88] also use McMullen's conditions to give a tight lower bound for the number of vertices of a polytope with m i-faces, $0 \leq i \leq \lfloor d/2 \rfloor$.

Some results are also known for families of polytopes other than simple/simplicial. Blind and Blind [83] show that any d-polytope without a triangular 2-face has an f-vector bounded below by that of the d-cube. Kortenkamp et al. [101] give an upper bound of $d! - (d-1)! + 2(d-1)$ facets for an n-vertex d-polytope whose vertices are 0/1-vectors. They give an example family of 0/1-polytopes with more than $(2.76)^d$ facets for sufficiently large d.

For notational convenience, the same algebraic construction of orthogonal sum is used for two different combinatorial purposes here. If P and Q are both full dimensional, then $P \oplus Q$ corresponds to what is known in the literature as a "free sum". Otherwise, it corresponds to a "join". The "diagonal sum" construction appears in a paper by Kortenkamp et al. [101] under the name "direct sum". The free sum operation can be defined in sufficient generality to include both the orthogonal and diagonal sum, by defining it as any operation that embeds the two polytopes in complementary subspaces (two subspaces that intersect in a point and whose union is the entire space) before taking their convex hull. For more details, see [94, 101].

Chapter 3

Pivoting Algorithms and Triangulation

A *basis* for (a vertex of) a d-polytope P is a set of d affinely independent facet-inducing hyperplanes whose intersection is feasible. A *pivot* operation moves from one basis to another by exchanging exactly one hyperplane. It turns out that the *basis graph*, whose nodes are bases and whose (directed) edges are pivots, is connected. This, along with the fact that every vertex has a basis means that the vertices of a polytope can be enumerated by exploring the basis graph; such an algorithm is called a *pivoting algorithm*. The number of bases can be much larger than the number of vertices: a vertex where m_0 facets meet can have as many as $\binom{m_0}{d}$ bases.

The earliest (and most popular in practice) technique to reduce the number of bases visited by a pivoting algorithm is to perturb the input hyperplanes so that the resulting polytope is simple. In general the number of vertices of the perturbed polytope is significantly larger than that of the input polytope. In Section 3.3 we give example families for which every perturbation produces a superpolynomial blowup.

The main step of pivoting algorithms is finding the neighbouring vertices of the current vertex (as opposed to the neighbouring bases of the current basis). Finding the neighbouring vertices to the current vertex v is equivalent to enumerating the edges adjacent to v. By considering a hyperplane that cuts off only v, we can recursively reduce this problem to a vertex enumeration problem in one less dimension. Such algorithms (see e.g. [11, 39, 35, 40]) are equivalent to computing the entire face lattice of the polytope. Families whose face lattice is superpolynomial in m, n, and d are well known. In Section 3.1, we give examples of such families with an infinite number of members in a given dimension. We also consider the variant of this method proposed by Swart [39] which computes a subset of the face lattice. Although this defeats certain simple lower bounds, it turns out also to be superpolynomial in the worst case.

In the facet enumeration setting, a basis is a set of d vertices defining a facet. A dual-pivoting or "gift-wrapping" (see e.g. [39]) algorithm moves between adjacent bases (those that share $d-1$ vertices) in much the same way as a pivoting algorithm for vertex enumeration. In order to reduce the number of bases visited, the input vertices are either perturbed, or the boundary of the polytope is *triangulated*, i.e. decomposed into $(d-1)$-simplices. Triangulation will be the bridge between bounds for face lattice producing algorithms and bounds for algorithms based on perturbation. The results on the face lattice of Section 3.1 will imply lower bounds on the size of triangulations in Section 3.2, and these triangulation bounds will in turn imply bounds on the size of the perturbations in Section 3.3. It should be noted that although triangulation does not provide a polynomial upper bound, the bound of $O(n^{\lfloor d/2 \rfloor})$ provided by the upper bound theorem is much better than the $\Omega(n^d)$ possible total bases.

The lower bounds in this chapter apply to pivoting based algorithms using either perturbation (e.g. [42, 12, 46]) or lattice enumeration [35, 39, 40]. They further apply to incremental algorithms that use triangulation (see e.g. [16, 48, 13]). Because the worst case complexity of the face lattice is asymptotically the same as the worst case output size (vertices or facets), several authors (see e.g. [15, 10]) define the "convex hull problem" as computing the face lattice of the polytope. The bounds of this chapter show that this can be prohibitively expensive for applications that only want the vertices or facets, especially since the face lattice is relatively easy to compute given the vertices and facets (see [26]). Several of these algorithms compute the face lattice of a perturbed polytope, and are thus potentially subject to two superpolynomial blowups. Finally, the lower bounds in this chapter apply to algorithms that apply pivoting to the dual polytope as a method of verifying the output, such as those developed in Chapter 5.

3.1 Computing the Face Lattice

This section presents examples of families whose face lattice is superpolynomial in their size. These families show that no convex hull algorithm that computes the face lattice is polynomial. Let T_d denote the d-dimensional simplex. Since every subset of vertices of a simplex defines a face (see e.g. [92]),

$$\bar{f}(T_d) = 2^{d+1} - 1\,. \tag{3.1}$$

Although computing the face lattice might be a useful technique if it were bad only for simplices, given (3.1) it is not hard to see that the face lattice is superpolynomial for any polytope that contains (or whose dual contains) a "high dimensional" simplicial face and whose input and output sizes are polynomial in the dimension. If P is a truncation d-polytope, $d \geq 3$, then

$$\bar{f}(P) \geq 2^{\Theta(d)}\,\text{size}(P)$$

This can be seen as follows. Let $m = c_0(P)$.

$$\begin{aligned} f_0(P) &= (m-d)(d-1)+2 && \text{By (2.2)} \\ &\leq 2(m-d)d && d \geq 2 \end{aligned}$$

Since for a simple polytope $f_0(p) \geq c_0(P)$, it follows $s \equiv \text{size}(P) \leq 4d^2(m-d)$. Summing (2.2), we get

$$\begin{aligned} \bar{f}(P) &= (m-d+1)(2^d-2)+3 \\ &> (m-d)(2^d-2) > s2^{d-2\log d-3} \qquad\qquad d \geq 3 \end{aligned}$$

Swart [39] observed that instead of computing the vertices of a simplicial k-face recursively, one can can compute them directly. Such an algorithm computes the *abbreviated face set* $\widehat{\mathfrak{F}}(P)$, defined recursively as follows:

$$\widehat{\mathfrak{F}}(P) = \{P\} \bigcup_{F \in \mathcal{F}(P)} \begin{cases} \{F\} & \text{if } F \text{ is a simplex,} \\ \widehat{\mathfrak{F}}(F) & \text{otherwise} \end{cases}$$

Let $\hat{f}(P)$ denote $|\widehat{\mathfrak{F}}(P)|$. Obviously $\hat{f}(P) \leq \bar{f}(P)$. For simple or simplicial polytopes, both $\hat{f}(P)$ and $\bar{f}(P)$ are linear in m and n, although exponential in d. Consider the simplicial case. Every facet is a $(d-1)$-simplex, hence by (3.1) has $2^d - 1$ non-empty faces. It follows that if P is simplicial (respectively simple), $\bar{f}(P) < 2^d m$ (respectively $\bar{f}(P) < 2^d n$). In [39], Swart conjectures that $\hat{f}(P)$ is linear (or "similar" to linear, i.e. bounded by a small polynomial) in $n+m$ for general polytopes. We will see below that for certain (non-simple and non-simplicial) families of polytopes, $\hat{f}(P)$ is not bounded by any polynomial in m and n independent of the dimension.

Computing the abbreviated face set instead of the complete face lattice is only advantageous if the polytope has one or more "large dimensional" simplicial faces. The following consequence of the Product Lemma suggests how to construct polytopes with large face lattices, but with no "large dimensional" simplicial faces.

Lemma 3.1 *Let* P_1 *and* P_2 *be polytopes.* $P_1 \times P_2$ *is a simplex iff* $\dim P_1 = 0$ *or* $\dim P_2 = 0$.

Proof. The "if" direction is clear from the Product Lemma. Let n_1 and n_2 be the number of vertices of P_1 and P_2 respectively. Let $d_1 = \dim P_1$ and let $d_2 = \dim P_2$.

Suppose $P_1 \times P_2$ is a simplex.

$$
\begin{aligned}
d_1 + d_2 + 1 &= n_1 \cdot n_2 && \text{By Lemma 2.3} \\
n_j &\geq d_j + 1 && j \in \{1, 2\} \\
d_1 + d_2 + 1 &\geq (d_1 + 1)(d_2 + 1) = d_1 d_2 + d_1 + d_2 + 1 \\
\therefore 0 &\geq d_1 d_2 && \square
\end{aligned}
$$

Corollary 3.1 *Let P and Q be polytopes containing the origin as an interior point. $P \oplus Q$ is a simplex iff $\dim P = 0$ or $\dim Q = 0$.*

Proof. The observation here is that $P \oplus Q$ is a simplex if and only if $(P \oplus Q)^*$, i.e. $P^* \times Q^*$ is. By Lemma 3.1 this holds iff $\dim P^* = 0$ or $\dim Q^* = 0$. $\square$

In order to make use of Lemma 3.1 and Corollary 3.1, we need families of polytopes whose face lattice is extremely large in terms of the number of vertices. A natural candidate family is the cyclic polytopes. On the other hand, cyclic polytopes, at least those realized as the convex hull of a set of points on the moment curve, are not well behaved numerically, since the bit complexity of the coordinates grows linearly with the dimension. Since the other dth order curves (see Section 2.2) seem to have similar numerical behaviour, it will prove useful to have (non-cyclic) families that behave in some ways like the cyclic polytopes, but have tighter bounds on the values of coordinates. We define two such families using the sum constructions of Section 2.4. For a polytope P, let $\bigoplus_k P$ denote the k-fold sum of P with itself, i.e.

$$\bigoplus_k P = \underbrace{P \oplus P \oplus \ldots \oplus P}_{k \text{ times}}$$

Define $\bigodot_k P$ analogously (recall that $\odot$ is the diagonal sum operator linearly equivalent to $\oplus$; see Lemma 2.5). Let π_n be a convex polygon with n vertices. For odd

n choose points as follows:

$$
\begin{aligned}
x(i) &= (n-1)/2 - i \\
y(i) &= 2x(i)^2 - 3 \qquad\qquad i = 0 \ldots n-1
\end{aligned}
$$

If n is even, construct π_{n-1} and then add a vertex with coordinates $(0, n^2/2 - 2n)$. Let H_d denote the d-dimensional hypercube with vertices in $\{-1, +1\}^d$. Define $\Pi_{2d}(n) \equiv \bigoplus_d \pi_n$ and $U_{3d} \equiv \bigodot_d H_3$. Recall that $f_0(P)$ and $c_0(P)$ denote respectively the number of vertices and facets of P. $\bar{f}(\pi_n) = 2n+1$. From Lemma 2.3 and Lemma 2.4, we know the following:

$$
f_0(\Pi_{2d}(n)) = dn \qquad c_0(\Pi_{2d}(n)) = n^d \qquad \bar{f}(\Pi_{2d}(n)) = (2n+1)^d
$$

H_3 has 1 3-face, 6 facets, 12 edges and 8 vertices. It follows that $\bar{f}(H_3) = 27$.

$$
f_0(U_{3d}) = 8d \qquad c_0(U_{3d}) = 6^d \qquad \bar{f}(U_{3d}) = 3^{3d}
$$

In both cases we have the desired property that the face lattice size is much larger than the vertex representation.

We now introduce three families of polytopes that are somewhat of a recurring theme in this thesis. Although there are perhaps easier ways of achieving our immediate goal of superpolynomially sized face lattices, these three families will turn out to be useful in Chapter 4 as well. Loosely speaking, we will take the $\sqrt{d}$-fold product of polytopes in dimension $\sqrt{d}$. In order to have families with members in every (sufficiently large) kth dimension for constant k, we interpolate in the following manner. For any $d \geq 2$ define $a = \lceil \sqrt{d} \rceil$, $b = \lfloor d/a \rfloor$, and $c = d \bmod a$.

Define

$$\dddot{C}_{2d}(n) \equiv C_{2a}^{b}(n) \times C_{2c}(n)$$
$$\dddot{\Pi}_{2d}(n) \equiv \Pi_{2a}^{b}(n) \times \Pi_{2c}(n)$$
$$\dddot{U}_{3d} \equiv U_{3a}^{b} \times U_{3c}\,.$$

The last term is omitted if $c = 0$, since multiplying by a 0-polytope merely changes the ambient space. Define $\operatorname{sgn}(x)$ as 0 if $x = 0$, $x/\operatorname{abs}(x)$ otherwise. Let $b' = \lceil d/a \rceil = b + \operatorname{sgn}(c)$. We will make use of the fact that $b' \leq a$ and $c < a$.

$$\begin{aligned} f_0(\dddot{C}_{2d}(n)) &= n^{b'} && \text{By Corollary 2.1a} \\ c_0(\dddot{C}_{2d}(n)) &= b\gamma(n, 2a) + \operatorname{sgn}(c) \cdot \gamma(n, 2c) && \text{By Corollary 2.1b} \\ &\in \Theta(n^{a}) && \text{d fixed} \end{aligned}$$

Thus we have

$$\text{size } \dddot{C}_{2d}(n) \in \Theta(n^{a}) \qquad \text{d fixed} \tag{3.2}$$

Similarly, for $\dddot{\Pi}_{2d}(n)$ we have

$$\begin{aligned} f_0(\dddot{\Pi}_{2d}(n)) &= (an)^{b}(cn)^{\operatorname{sgn}(c)} \\ &\leq (an)^{b'} \\ c_0(\dddot{\Pi}_{2d}(n)) &= bn^{a} + \operatorname{sgn}(c)n^{c} \end{aligned}$$
$$\text{size } \dddot{\Pi}_{2d}(n) \in \Theta(n^{a}) \qquad \text{d fixed} \tag{3.3}$$

It should be noted that (3.3) hides a constant of approximately $d^{\sqrt{d}}$ and hence is only of interest when $n \gg d$. Similar arithmetic to the above reveals

$$
\begin{aligned}
f_0(\dddot{U}_{3d}) &\le (8a)^{b'} && \le (8a)^a \\
c_0(\dddot{U}_{3d}) &\le b'6^a && \le (8a)^a \\
\text{size}\,\dddot{U}_{3d} &\le 6d(8a)^a && \le 2^{3a+3}a^{a+2} \\
&= 2^{a(3+\log a+(2\log a+3)/a)} && \\
&\le 3^{a\log a} && d \text{ sufficiently large}
\end{aligned}
\tag{3.4}
$$

Using the fact that the size of the face lattice multiplies under both the sum and product operations, we have

$$
\begin{aligned}
\bar{f}(\dddot{C}_{2d}(n)) &\in \Theta(n^d) && d \text{ fixed} \\
\bar{f}(\dddot{\Pi}_{2d}(n)) &\in \Theta(n^d) && d \text{ fixed} \\
\bar{f}(\dddot{U}_{3d}) &= 3^{3d} &&
\end{aligned}
\tag{3.5}
$$

By Lemma 3.1, $\dddot{C}_{2d}(n)$ has no simplicial face with dimension greater than $\lceil\sqrt{d}\rceil$. By Corollary 3.1, neither $\Pi_{2d}(n)$ nor U_{3d} has a simplicial face with dimension higher than 1. By Lemma 3.1 it follows that neither do products of either of these families. Since every 1-face is simplicial, it follows that

$$
\hat{f}(\dddot{\Pi}_{2d}(n)) \ge \bar{f}(\dddot{\Pi}_{2d}(n))/2 \qquad \hat{f}(\dddot{U}_{3d}) \ge \bar{f}(\dddot{U}_{3d})/2
$$

We summarize the results of this section as follows.

$$
\begin{aligned}
\hat{f}(\dddot{C}_{2d}(n)) &\in \Theta(s_c^{d/\lceil\sqrt{d}\rceil}) && \text{where} && s_c \equiv \text{size}\,\dddot{C}_{2d}(n) && d \text{ fixed} \\
\hat{f}(\dddot{\Pi}_{2d}(n)) &\in \Theta(s_p^{d/\lceil\sqrt{d}\rceil}) && \text{where} && s_p \equiv \text{size}\,\dddot{\Pi}_{2d}(n) && d \text{ fixed} \\
\hat{f}(\dddot{U}_{2d}) &\in \Omega(s_u^{3d/(\lceil\sqrt{d}\rceil\log\lceil\sqrt{d}\rceil)}) && \text{where} && s_u \equiv \text{size}\,\dddot{U}_{3d} &&
\end{aligned}
$$

3.2 Triangulation

In this section we work in the context of facet enumeration; as usual this is merely a matter of convenience, and everything here can be interpreted in the dual vertex enumeration context. It will be useful in the next two sections to work with objects slightly more general than polytopes. A set of polytopes Γ is called a *polytopal complex* if

1. The empty polytope is in Γ,

2. If $P \in \Gamma$, then the faces of P are also in Γ, and

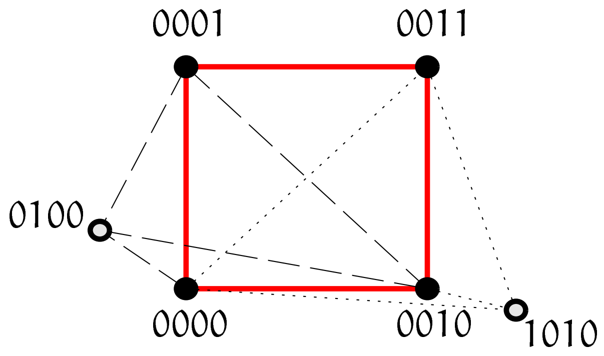
3. for any pair $\{P_i, P_j\} \subset \Gamma$, $P_i \cap P_j$ is a (possibly empty) face of both P_i and P_j.

Since the only complexes we are interested in here are polytopal, we usually abbreviate polytopal complex to *complex*. The polytopes that make up a complex are called its *faces*. By the maximal faces of complex we mean those not contained in any other face. The *boundary complex* of a polytope P is the complex consisting of all of the k-faces of P, $k < d$. Note that the face lattice of a polytope can be viewed as a complex, but a complex is not necessarily a lattice. A complex Δ is called a *decomposition* of a polytope P if $P = \bigcup_{Q \in \Delta} Q$. Complex Γ' is called a refinement of complex Γ if it consists of decompositions of the faces of Γ. A complex Γ is called simplicial if each $P \in \Gamma$ is simplex. A simplicial refinement of a complex is called a *triangulation*. Triangulation is used not only in pivoting based algorithms for facet enumeration, but in certain incremental algorithms. In the former case, the idea is to reduce the number of bases (facet defining sets of vertices) visited by the algorithm. In the latter case, the idea is to improve the performance of the algorithm in low dimensions or to meet the worst case bound of $O(n^{\lfloor d/2 \rfloor})$. In the case of pivoting algorithms, although triangulation is no worse than enumerating all bases of a polytope, it cannot guarantee a polynomial algorithm. In the case of incremental algorithms, not only does triangulation not guarantee a polynomial algorithm, but it can make an algorithm superpolynomial for certain families of polytopes, even given a good insertion order.

Triangulations without extra vertices (i.e. those whose 0-faces are the vertices of the polytope) are of special interest, both from a theoretical point of view and from the point of view of facet enumeration algorithms. It will turn out that the bounds in this section hold for the more general case of allowing additional vertices; we will see in Section 3.3 that they hold even for more general sets of simplices than triangulations.

There are several problems closely related to triangulating a polytope, namely triangulating its facets (individually) and triangulating its boundary complex. Given a triangulation Δ of the boundary complex of P, we can extend it to a triangulation of P (see Section 3.2.1). On the other hand, a triangulation of the facets is not necessarily a triangulation of the boundary: consider for example a ridge of a 4-cube (itself a 2-cube) where the triangulations of the adjacent facets choose different diagonals (see Figure 3.1). A lower bound on the number of $(d-1)$-simplices necessary to triangulate each facet will provide a lower bound for the number of $(d-1)$-simplices neccesary to triangulate the boundary and the number of d-simplices neccesary to triangulate the polytope.

Figure 3.1 The ridge $x_1 = x2 = 0$ of a 4-cube triangulated in two different ways by triangulations of facets $x_1 = 0$ and $x_2 = 0$



Several known bounds on the size of triangulations are based on volume. By providing a lower bound for the volume of the polytope P to be triangulated, and an upper bound for the volume of a simplex contained in P, a lower bound for the number of maximal simplices necessary to triangulate P is obtained. Section 3.2.1 comprises an exposition of two such bounds.

Similarly to the bound on the face lattice size in terms of the number of facets of a simplicial polytope, we can obtain a bound in terms of the number of maximal simplices in a triangulation. Suppose we have a triangulation Δ of d-polytope P with t d-simplices. For $k < d$, every k-face of P must be triangulated by Δ, i.e. must contain at least one k-face of some $T \in \Delta$. Since k-faces of P do not intersect except in j-faces, $j < k$, it follows that

$$f_k(P) \leq t\binom{d}{k+1} \qquad k < d$$

$$\bar{f}(P) \leq t\sum_{k=0}^{d-1}\binom{d}{k+1} + 1 \leq t2^d \tag{3.6}$$

Note that (3.6) does not depend on whether Δ contains extra vertices or not. Here we are interested in the interpretation of (3.6) as a lower bound on the size of a triangulation. Let $\mathrm{tmin}(P)$ denote the minimum number of d-simplices necessary to triangulate a d-polytope P.

$$\mathrm{tmin}(P) \geq \bar{f}(P)2^{-d} \tag{3.7}$$

Let $\mathrm{tmin}_F(P)$ denote the minimum number of $(d-1)$-simplices necessary to triangulate the facets of P.

$$\begin{aligned} \mathrm{tmin}_F(P) &= \sum_{F\in\mathcal{F}(P)} \mathrm{tmin}(F) \\ \mathrm{tmin}_F(P) &\geq 2^{-d+1}\sum_{F\in\mathcal{F}(P)} (\bar{f}(F)) && \text{By (3.7)} \\ &> \bar{f}(P)\,2^{-d+1} && (3.8) \end{aligned}$$

From (3.8), and the bounds (3.5) on face lattices from Section 3.1, we have the following

$$\begin{aligned} \mathrm{tmin}_F(\dddot{C}_{2d}(n)) &\in \Omega(n^d) && \text{d fixed} \\ \mathrm{tmin}_F(\dddot{\Pi}_{2d}(n)) &\in \Omega(n^d) && \text{d fixed} \\ \mathrm{tmin}_F(\dddot{U}_{3d}) &\geq 3^{3d}/2^{3d} \\ &> 3^d \end{aligned}$$

Use the size calculations of (3.2), (3.3), and (3.4), we have almost the same results as for the face lattice sizes.

$$\begin{aligned} \mathrm{tmin}_F(\dddot{C}_{2d}(n)) &\in \Omega(s_c^{d/\lceil\sqrt{d}\rceil}) && \text{where} && s_c \equiv \text{size } \dddot{C}_{2d}(n) && \text{d fixed} \\ \mathrm{tmin}_F(\dddot{\Pi}_{2d}(n)) &\in \Omega(s_p^{d/\lceil\sqrt{d}\rceil}) && \text{where} && s_p \equiv \text{size } \dddot{\Pi}_{2d}(n) && \text{d fixed} \\ \mathrm{tmin}_F(\dddot{U}_{3d}) &\in \Omega(s_h^{d/(\lceil\sqrt{d}\rceil \log\lceil\sqrt{d}\rceil)}) && \text{where} && s_h \equiv \text{size } \dddot{U}_{3d} \end{aligned}$$

3.2.1 Triangulation and Volume

This section gives an exposition of two lower bounds for triangulation sizes based on volume considerations. We start by presenting some relevant background on volume in $\mathbb{R}^d$. By the d-dimensional volume $\mathrm{Vol}(P)$ we mean the standard Lebesgue measure. If P is not full dimensional, we use $\mathrm{Vol}(P)$ to mean the volume of P in $\mathrm{aff}\, P$. Let T_d denote a d-simplex. We take the following as given[1] (see e.g. [81, 94]).

$$\mathrm{Vol}\, T_d = \frac{1}{d!} \,\mathrm{abs} \begin{vmatrix} \mathcal{V}(T_d) & \mathbb{1} \end{vmatrix} \tag{3.9}$$

For a k-polytope $P \subset \mathbb{R}^d$ and point $v \notin \mathrm{aff}\, P$ (the *apex*), define the $(k+1)$ dimensional *pyramid* $\mathrm{Pyr}(P, v)$ as $\mathrm{conv}(P \cup \{v\})$. Let $h(p, P)$ denote the denote the perpendicular distance from p to $\mathrm{aff}\, P$. From properties of determinants (see e.g.

[1] For our purposes this could serve as a definition of volume, since the volume of any polytope is defined here in terms of a triangulation.

(24.16) in [81]), we have the following:

$$\operatorname{Vol}\operatorname{Pyr}(T_{d-1}, v) = \frac{h(v, T_{d-1}) \operatorname{Vol} T_{d-1}}{d}. \tag{3.10}$$

Suppose we have a polytope P containing the origin as an interior point. Let F be a facet of P. Let $\alpha(F)$ be the d-vector such that $\alpha(F)x \leq 1$ supports P in F.

$$h(\mathbb{O}, F) = 1/\|\alpha(F)\| \tag{3.11}$$

If we consider some arbitrary triangulation of F, from (3.10) and (3.11), we have:

$$\operatorname{Vol}\operatorname{Pyr}(F, \mathbb{O}) = \frac{\operatorname{Vol} F}{d\|\alpha(F)\|}. \tag{3.12}$$

By considering the decomposition of P into pyramids with the origin as apex it follows that

$$\operatorname{Vol} P = \sum_{F \in \mathcal{F}(P)} \frac{\operatorname{Vol}(F)}{d\|\alpha(F)\|}. \tag{3.13}$$

Lemma 3.2 *Let* P *and* Q *be polytopes.* $\operatorname{Vol}(P \times Q) = \operatorname{Vol}(P) \cdot \operatorname{Vol}(Q)$

Proof. Let $k = \dim P$ and $l = \dim Q$. We proceed by induction. The lemma holds if $k+l = 0$. Now suppose $k+l = d > 0$, and the lemma holds for $k+l = d-1$. Without loss of generality, suppose that both P and Q (and hence $P \times Q$) contain the origin as an interior point. As in the preceding discussion, consider the decomposition of $P \times Q$ into pyramids with the origin as apex. We know by Lemma 2.3 that every facet of $P \times Q$ is the product of a facet of one polytope with the other polytope. Let $F = F_p \times Q$ for some $F_p \in \mathcal{F}(P)$.

$$\begin{aligned}
\operatorname{Vol}\operatorname{Pyr}(F, \mathbb{O}) &= \frac{\operatorname{Vol} F}{d\|\alpha(F)\|} && \text{By (3.12)} \\
&= \frac{\operatorname{Vol}(F_p) \cdot \operatorname{Vol}(Q)}{d\|\alpha(F)\|} && \text{By the inductive hypothesis}
\end{aligned}$$

$$= \frac{\mathrm{Vol}(Q)\,k}{k+l}\,\frac{\mathrm{Vol}(F_p)}{k\|\alpha(F_p)\|} \qquad\qquad \text{since } \|\alpha(F)\| = \|\alpha(F_p)\|$$

By summing over all such facets F, and considering the symmetric case of facets of the form $P \times F_q$, we arrive at

$$\mathrm{Vol}(P \times Q) = \left(\frac{\mathrm{Vol}(Q)\,k}{k+l}\sum_{F_p \in \mathcal{F}(P)} \frac{\mathrm{Vol}(F_p)}{k\|\alpha(F_p)\|}\right) + \left(\frac{\mathrm{Vol}(P)\,l}{k+l}\sum_{F_q \in \mathcal{F}(Q)} \frac{\mathrm{Vol}(F_q)}{l\|\alpha(F_q)\|}\right)$$

$$= \mathrm{Vol}(P) \cdot \mathrm{Vol}(Q) \qquad\qquad \text{By (3.13)} \qquad \square$$

Perhaps the most well known family of difficult to triangulate polytopes are the hypercubes H_d. Haiman [70] gives the following bound:

$$\mathrm{tmin}\, H_d \geq \frac{2^d d!}{(d+1)^{(d+1)/2}}\,. \tag{3.14}$$

The essence of Haiman's argument is to give an upper bound on the size of the largest simplex contained in H_d. Let $\mathrm{ball}(c, r)$ denote the d-*ball* of radius r centered at c, i.e. $\{x \mid \|x - c\| \leq r\}$. If we take the standard $\{+1, -1\}$ coordinatization of H_d, every vertex lies on the boundary of $\mathrm{ball}(\mathbb{O}, \sqrt{d})$. Fejes Tóth [111] proved that the maximum volume d-simplices contained in a d-ball are the regular ones (where the distance between every pair of vertices is the same). According to [111] the regular d-simplex with circumradius $\sqrt{d}$ has volume $V_{max} = (d+1)^{(d+1)/2}/d!$. As a consequence of Lemma 3.2, the $\{+1, -1\}$-hypercube has volume 2^d. It follows that any triangulation of H_d must contain at least

$$\frac{2^d}{V_{max}} = \frac{2^d\, d!}{(d+1)^{(d+1)/2}}$$

d-simplices. Summing over all facets, we have the following.

Theorem 3.1 *Let* H_d *be a* d*-cube.*

$$\begin{aligned} \operatorname{tmin}_F H_d &\geq s_d^{(1/4)\log\log s_d - 1/2} \\ &\geq s_d^{(1/4)\log d - 1/2} \end{aligned}$$

Proof. The hypercube H_d has 2^d vertices and $2d$ facets. Thus we have

$$\operatorname{size} H_d = d(2^d + 2d) = s_d \leq 2^{2d} = u_d\,.$$

By (3.14), the number of $(d-1)$-simplices required to triangulate the $2d$ facets of H_d (each of which is a $(d-1)$-cube) is at least

$$\frac{2d\,2^{d-1}\,(d-1)!}{d^{d/2}} = \frac{2^d\,d!}{d^{d/2}} > \frac{2^d\,d^d}{d^{d/2}e^d} = \frac{d^{d/2}}{2^{(\log e - 1)d}} > (d/2)^{d/2}$$

But $(d/2) = (1/4)\log u_d \geq (1/4)\log s_d$ and thus triangulating the boundary of H_d requires at least

$$(d/2)^{d/2} \geq [(1/4)\log s_d]^{(1/4)\log s_d} = s_d^{(1/4)\log\log s_d - 1/2}$$

$(d-1)$-simplices, as claimed. The second inequality follows from the fact that $\log s_d > d$. ☐

By the Product Lemma, $\bar{f}(H_d) = 3^d$. Since $\operatorname{size} H_d > 2^d$, it follows that $\bar{f}(H_d) < \operatorname{size}(H_d)^{\log_2 3}$. Thus we see that the the triangulation complexity of a polytope is not always bounded above by the face lattice size.

The lower bound of Theorem 3.1 is only slightly superpolynomial in $\operatorname{size}(P)$. We now show how to obtain a much stronger bound.

Theorem 3.2 [67, 70, 69] Let T_d denote a d-simplex. Every triangulation of $T_k \times T_l$ without extra vertices requires exactly $\binom{k+l}{l}$ $(k+l)$-simplices. Every triangulation of $T_k \times T_l$ requires at least $\binom{k+l}{l}$ $(k+l)$-simplices.

The proof of Theorem 3.2 uses the tools of *unimodular matrices*. A matrix is called *totally unimodular* if every square submatrix has determinant -1, 0, or $+1$. We will use (but not prove) the following theorem, due to Ghouila-Houri [91] (for a more accessible proof, see [109])

Theorem 3.3 [91, 109] Let A be a matrix with entries $\{-1, 0, +1\}$. A is totally unimodular iff every collection of columns of A can be split into two sets such that the sum of the columns in one set minus the sum of the columns in the other set is a vector with entries $\{-1, 0, +1\}$.

The following lemma will allow us to extend the bound (based on unimodular matrices) for triangulations of $T_k \times T_l$ without extra vertices to the general case.

Lemma 3.3 *Let* P *be a* d*-polytope, and* T *a* d*-simplex such that* $T \subseteq P$. *There exists a* d*-simplex* $T' \subseteq P$ *such that* $\operatorname{Vol} T' \geq \operatorname{Vol} T$ *and* $\mathcal{V}(T') \subset \mathcal{V}(P)$.

Proof. Let v be a vertex of T, i.e. $T = \operatorname{Pyr}(Q, v)$ where Q is $d-1$ simplex. By (3.10), $\phi(x) = \operatorname{Vol} \operatorname{Pyr}(Q, x)$ is a linear function of $h(x, Q)$. By the Fundamental Theorem of Linear Programming (see e.g. [85]), there exists some $x^* \in \mathcal{V}(P)$ such that $\phi(x^*) \geq \phi(v)$. □

Proof. (of Theorem 3.2) Let T_d denote convex hull of $\{\boldsymbol{e}_0 \ldots \boldsymbol{e}_d\}$. Let P be the product $T_k \times T_l$. Let $A \in \{0, 1\}^{m \times (k+l+1)}$ be defined by $a_i = (v_i, 1)$ where v_i is the ith vertex of P. We claim that A is totally unimodular. Consider some collection $\mathcal{C}$ of columns of A. Let γ_1 be the sum of columns from $\mathcal{C}$ with index at most k (i.e. columns from T_k). Let γ_2 be the sum of columns from $\mathcal{C}$ with index between $k+1$ and $k+l$ (i.e. columns from T_l). If the last column of A is in $\mathcal{C}$, let $\gamma = \mathbb{1} - \gamma_1 - \gamma_2$; otherwise let $\gamma = \gamma_1 - \gamma_2$. In either case, $\gamma \in \{0, \pm 1\}^m$. By Theorem 3.3, it follows that A is totally unimodular. In particular it follows by (3.9) that every full dimensional simplex with vertices in $\mathcal{V}(P)$ has volume $1/(k+l)!$ (note that since the determinant of an integer matrix is an integer, this is the smallest possible volume for a $(k+l)$-simplex with integer vertices). By Lemma 3.2, P has volume $1/(k!l!)$. It follows that every triangulation of P without extra vertices has at exactly $\binom{k+l}{l}$

$(k+l)$-simplices. By Lemma 3.3, every triangulation of P has at least this many $(k+l)$-simplices.

To generalize to products of arbitrary simplices, it suffices to note that transforming from an arbitrary simplex to $\mathrm{conv}\{e_0 \dots e_d\}$ amounts to a translation followed by a change of basis, i.e. an affine transformation. Taken together, these two transformations are also an affine transformation in $\mathbb{R}^{k+l}$. Since affine transformations preserve triangulations, Theorem 3.2 follows. □

We can restate Theorem 3.2 in terms of the size function. Let $\dddot{T}_{2d}$ denote $T_d \times T_d$.

Theorem 3.4 *Let* $s = \mathrm{size}(\dddot{T}_{2d})$. *For* $d > 3$,

$$\mathrm{tmin}_F(\dddot{T}_{2d}) \geq 2^{\sqrt[3]{s}} \geq \frac{1}{2} s^{d/(3\log d)}$$

Proof. From Lemma 2.3, every facet of $\dddot{T}_{2d}$ is combinatorially equivalent to $T_{d-1} \times T_d$. By Corollary 2.1a, $f_0(\dddot{T}_{2d}) = (d+1)^2$ and by Corollary 2.1b, $c_0(\dddot{T}_{2d}) = 2(d+1)$; hence

$$\begin{aligned} \mathrm{size}(\dddot{T}_{2d}) &= 2d((d+1)^2 + 2(d+1)) = (2d^2+2d)(d+3) & \\ &< (2d^2+3d-2)(d+3) & d \geq 3 \\ &= (2d-1)(d+2)(d+3) < (2d-1)^3 & d > 3 \end{aligned}$$

By Theorem 3.2, we know the number of $(d-1)$ simplices to triangulate all facets is at least

$$\begin{aligned} 2(d+1)\binom{2d-1}{d} &> \sum_{i=0}^{2d-1} \binom{2d-1}{i} \\ &\geq 2^{2d-1} \end{aligned}$$

This establishes the first inequality. Note that for $d \geq 4$, $s \leq 8d^3$, hence

$$2^{2d-1} = 2^{2d}2^{-1} = \frac{1}{2}s^{2d/\log s} \geq \frac{1}{2}s^{2d/(3(\log d+1))} \qquad \square$$

In contrast to the d-cubes, the face lattice is also superpolynomial for $\dddot{T}_{2d}$. By (3.1) and the Product Lemma, $\bar{f}(\dddot{T}_{2d}) \geq 2^{2d} \geq \text{size}(\dddot{T}_{2d})^{d/(2\log d)}$ (d sufficiently large).

3.3 Perturbation

In this section we analyze the relationship between triangulation and perturbation. To avoid a dualization step, we again work in the context of facet enumeration. Perturbation is a useful practical technique to reduce the number of feasible bases of a polytope. Conceptually, one moves each input point a small amount, thus breaking ties within large sets of points lying on the same facet. In order to apply this technique to facet enumeration, one must characterize what kinds of perturbation are permissible. This is possible either by giving numerical bounds [74], by defining the perturbation symbolically [76], or in terms of limits [75]. Here we give a geometric definition, and show that the lower bounds of Section 3.2 apply to algorithms using these perturbations.

It will be convenient in this section to identify faces of polytopes and complexes with their vertex sets. For d-polytope P, $v \in \mathcal{V}(P)$, and $v' \in \mathbb{R}^d$, we say that $\mathcal{V}(P) \setminus \{v\} \cup \{v'\}$ is a *local perturbation* of P if the half-open line segment $(v, v']$ does not intersect any hyperplane induced by a $(d-1)$-face of $P \setminus \{v\}$. In this case, we write $p_v^{v'}(P)$ for $\mathcal{V}(P) \setminus \{v\} \cup \{v'\}$.

Lexicographic triangulations [73, 68] are defined in terms of two refinements. A point v is *beneath* (respectively *beyond*) a facet Q of a polytope P if it is in the open halfspace induced by aff Q containing (respectively not containing) $\text{int}(P)$. Let Γ be a complex (for definitions of complexes, refinements, and triangulations, see Section 3.2). The refinement $\text{pull}(v, \Gamma)$ is defined as follows. For each $F \in \Gamma$,

1. If $v \notin F$, then $F \in \text{pull}(v, \Gamma)$.

2. If $v \in F$, then $\mathrm{pull}(v, \Gamma)$ contains $G \cup \{v\}$ for each face G of F such that $v \notin G$.

The second refinement $\mathrm{push}(v, \Gamma)$ is defined as follows. For each $F \in \Gamma$,

1. If $v \notin F$, then $F \in \mathrm{push}(v, \Gamma)$.

2. If $v \in F$ and $\dim(F \setminus \{v\}) < \dim F$, then $F \in \mathrm{push}(v, \Gamma)$.

3. If $v \in F$ and $\dim(F \setminus \{v\}) = \dim(F) = k$, then $\mathrm{push}(v, \Gamma)$ contains $F \setminus \{v\}$, the faces of $F \setminus \{v\}$, and $G \cup \{v\}$ for each $(k-1)$-face G of $F \setminus \{v\}$ such that v is beyond G (in aff F).

For a polytope P, we use $\mathrm{push}(v, P)$ (respectively $\mathrm{pull}(v, P)$) to mean $\mathrm{push}(v, \Gamma)$ (respectively $\mathrm{pull}(v, \Gamma)$) where Γ is the complex of faces of P. A *lexicographic triangulation* is the decomposition of a polytope obtained by choosing an ordering $v_1 \ldots v_n$ of $\mathcal{V}(P)$ and applying either a push or a pull operation to each vertex in this order. To see that this is a triangulation, note that after a vertex v has been pushed or pulled, for every face F of the decomposition containing v, $\dim(F \setminus \{v\}) < \dim F$. After every vertex of a polytope has been this is true for each vertex v of each face F, thus the complex is simplicial.

We now argue that the combinatorial pushing and pulling operations correspond to certain local perturbations. We can view local perturbations as a certain kind of incremental construction. The following lemma, due to Grünbaum [92], characterizes the combinatorial structure of adding a vertex to a full dimensional polytope.

Lemma 3.4 *Let* P_0 *and* P *be two* d*-polytopes such that* $P = \operatorname{conv}(P_0 \cup \{v\})$*.* F *is a face of* P *iff*

(a) F *is a face of* P_0 *such that* $v \notin F$ *and there exists facet* $Q \supset F$ *of* P *such that* v *is beneath* Q*,*

(b) $F = G \cup \{v\}$ *where* G *is face of* P_0 *and* $v \in \operatorname{aff}(G)$*, or*

(c) $F = G \cup \{v\}$ *where* G *is face of* P_0 *and* G *is contained in two facets* Q_+ *and* Q_- *of* P_0 *such that* v *is beneath* Q_+ *and beyond* Q_-*.*

We say that two complexes are *combinatorially equivalent* if they have isomorphic face posets. Let us fix some d-polytope P and some local perturbation $P' = p_v^{v'}(P)$. Define a mapping $\varphi(F)$ as follows

$$\varphi(F) = \bigcup_{x \in F} \begin{cases} \{v'\} & \text{if } x = v \\ \{x\} & \text{if } x \neq v\,. \end{cases}$$

Let Q denote some facet of P containing v. Let $\Gamma(Q)$ denote the complex of faces of P' that are subsets of $\varphi(Q)$. Let P_0 denote $P \setminus \{v\}$ and let Q_0 denote $Q \setminus \{v\}$.

Lemma 3.5 *If* $\dim Q_0 < \dim Q$ *then* $\Gamma(Q)$ *is combinatorially equivalent to both* $\operatorname{push}(v, Q)$ *and* $\operatorname{pull}(v, Q)$*.*

Proof. If $\dim Q_0 < \dim Q$, Q is a pyramid, and hence combinatorially unchanged by locally perturbing v (see e.g. Theorem 7.7 in [84]). Furthermore, if $\dim Q_0 < \dim Q$, by definition both $\operatorname{push}(v, Q)$ and $\operatorname{pull}(v, Q)$ are simply the complex of faces of Q. □

Lemma 3.6 *If* v' *is beyond* Q *then* $\Gamma(Q)$ *is combinatorially equivalent to* $\operatorname{pull}(v, Q)$*.*

Proof. The mapping φ is one to one and preserves inclusion. It remains to show that $F \in \operatorname{pull}(v, Q)$ iff $\varphi(F) \in \Gamma(Q)$. By Lemma 3.5, we need only consider the case that $\dim Q_0 = \dim Q$.

Suppose $\varphi(F) \in \Gamma(Q)$. Further suppose $\nu' \notin \varphi(F)$. It follows that $F = \varphi(F)$ is a face of Q_0. Since F is preserved when adding ν', by Lemma 3.4 there must be some facet $Q_- \supset F$ of P_0 that ν' is beneath. By the definition of local perturbation ν must also be beneath Q_-, so F is a face of Q and hence of $\text{pull}(\nu, Q)$. If $\varphi(F) = G \cup \{\nu'\}$, then by Lemma 3.4 G is a face of Q_0. By the argument immediately preceding, G is also a face of Q; it follows that $G \cup \{\nu\}$ is a face of $\text{pull}(\nu, Q)$ by definition.

Suppose $F \in \text{pull}(\nu, Q)$. Further suppose that $\nu \notin F$. It follows that F must also be a face of Q (from the definition of $\text{pull}(\nu, Q)$) and Q_0. By Lemma 3.4 (working in $\text{aff}\, Q$) there must be some ridge R of Q such that ν is beneath R (in $\text{aff}(Q)$), and hence beneath the corresponding facet $Q_+ \supset R$. From the definition of a local perturbation, ν' is also beneath Q_+, hence by Lemma 3.4a $\varphi(F) = F$ is a face of P'. Suppose $F = \{\nu\} \cup G$. It follows by definition of $\text{pull}(\nu, Q)$ that G is a face of Q. Since $\nu \notin G$, as above G must be contained in some facet Q_+ of P_0 such that ν and ν' are beneath Q_+. But G is also contained in Q and ν' is beyond Q, so by Lemma 3.4c $\varphi(F) = (\{\nu'\} \cup G)$ is a face of $\Gamma(Q)$. □

Lemma 3.7 *If ν' is beneath Q then $\Gamma(Q)$ is combinatorially equivalent to* $\text{push}(\nu, Q)$.

Proof. As in the proof of Lemma 3.6, we need only show $F \in \text{push}(\nu, Q)$ iff $\varphi(F) \in \Gamma(Q)$ and we may assume that $\dim Q_0 = \dim Q$.

Suppose $\varphi(F) \in \Gamma(Q)$. If $\nu' \notin \varphi(F)$, then $F = \varphi(F)$ is a face of Q_0, hence a face of $\text{push}(\nu, Q)$ by definition. If $\nu' \in \varphi(F)$, then by Lemma 3.4 $\varphi(F) = \{\nu'\} \cup G$ for some face G of P_0. Since $\nu' \notin \text{aff}\, G$, by Lemma 3.4 $G \subseteq Q_+ \cap Q_-$, where $\{Q_+, Q_-\} \subseteq \mathcal{F}(P_0)$ and ν' is beneath (resp. beyond) Q_+ (resp. Q_-). By definition of a local perturbation ν is also beyond Q_-, and hence beyond the corresponding ridge of Q_0. It follows that $G \cup \{\nu\}$ is a face of $\text{push}(\nu, Q)$.

Suppose $F \in \text{push}(\nu, Q)$. If $\nu \notin F$, F is a face of P_0. Since $F \subseteq Q$ and ν' is beneath Q, by Lemma 3.4a, $\varphi(F)$ is also a face of P'. Suppose that $F = \{\nu\} \cup G$. By Lemma 3.4, G must be a face of P_0. From the definition of $\text{push}(\nu, Q)$, there are two cases.

1. If $\dim G < \dim F$, then v cannot be in aff G, hence by Lemma 3.4, there must be some facet $Q_- \supset G$ of P_0 such that v is beyond Q_-.

2. If $\dim G = \dim F$, then by definition of push(v, Q), G must be contained in some ridge of Q_0 such that v is beyond the corresponding facet Q_-.

In either case, since v' is also beyond Q_- (and beneath Q), by Lemma 3.4c $\varphi(F) = G \cup \{v'\}$ is a face of P'. □

If v' is either beneath every facet containing v or beyond every facet of P containing v, then we call $p_v^{v'}(P)$ a *lexicographic perturbation*. In this case we see from Lemma 3.6 and Lemma 3.7 that the boundary complex of P' is combinatorially equivalent to the refinement of the boundary complex of P induced by push(v, P) or pull(v, P). Thus by lexicographically perturbing each vertex of a polytope P sequentially, we obtain a simplicial polytope whose boundary complex is combinatorially equivalent to a lexicographic triangulation of the boundary complex of P. Lexicographic perturbations date back at least to the "right hand side" perturbations proposed by Charnes [12]. Interpreted in our (polar) context of perturbing vertices, this perturbation is defined as follows. Given a polytope $P = \operatorname{conv} V$ containing the origin as an interior point, $P(\epsilon)$ is defined as conv V' where $v_i' = v_i/(1+\epsilon^i)$ for some $1 \gg \epsilon > 0$. For ϵ sufficiently small, this is equivalent to pushing the vertices of P in the order $v_1, v_2, \dots v_n$. Other lexicographic perturbations/triangulations can be obtained by setting v_i' to $v_i/(1-\epsilon^i)$ for certain vertices. Dantzig, Orden and Wolf [17] showed how to implement a lexicographic (pushing) perturbation without computing a value for ϵ (see Section 5.4.1 for discussion).

For a (not necessarily lexicographic) local perturbation $p_v^{v'}(P)$, v' may be beyond some facets containing v, but beneath others. In this case the boundary complex of the perturbed polytope is not necessarily combinatorially equivalent to a refinement of the boundary complex of the original polytope. For example, d vertices in a ridge of P may map to facet of $p_v^{v'}(P)$. Since these vertices have dimension $d-2$ in P, they cannot be a basis of some facet of P. Call a set of d-polytopes $\{P_1 \dots P_k\}$ a *cover* of a d-polytope P if $P = \bigcup_i P_i$. We claim that neither the face lattice based bounds nor the volume based bounds of Section 3.2 depend on properties of a triangulation other than its being a cover. For volume based bounds, allowing simplices to intersect arbitrarily does not affect the bounds on the volume of a simplex contained in P. For lattice based bounds, the bounds follow from the fact that every k-face of a polytope P must contain some k-face F of a cover, and F is contained in a unique k-face of P.

Given any d-polytope, we can always lexicographically perturb it to a simplicial polytope without increasing the number of bases (since the corresponding lexicographic triangulation takes a subset of bases as $(d-1)$-simplices). Thus for the purposes of lower bounds, we may assume that our perturbed polytope is simplicial. Now consider locally perturbing the vertices of some polytope P one by one. At each step, we have an induced cover of each facet (with the additional property that no pair of simplices intersect in their interiors). Locally perturbing a vertex of some facet Q of the perturbed polytope decomposes the corresponding $(d-1)$-polytope in the cover. By this recursive decomposition process we arrive at a cover of each facet of P using no more simplices than the number of facets of the perturbed simplicial polytope. Thus we see that the lower bounds of Section 3.2 apply to algorithms using local perturbation (or equivalent symbolic schemes) as well.

3.4 Experimental Results

In this section we examine the performance of four programs that directly or indirectly use triangulation. **lrs** [42] is an implementation of the *reverse search* algorithm of Avis and Fukuda [5] using exact arithmetic. **lrs** uses lexicographic perturbation. **hull** [45] and **qhull** [43] are insertion based programs that use triangulation. **pd** is a *primal-dual* algorithm that uses lexicographic perturbation on the dual polytope (see Chapter 5 for details). In order to compare the size of the triangulations computed by all three programs from the point of view of lower bounds, the measurements for **pd** presented here for performing the dual transformation to the other two. All experiments in this section are on a Digital AlphaServer 4/233 with

256M of real memory and 512M of virtual memory. The notation “memory limit” means working storage exceeded 128M; this limit was imposed to avoid problems with thrashing.

3.4.1 Lattice Based bounds

For all three families with lattice based lower bounds, we report the triangulation sizes for **qhull** summed over intermediate polytopes.

We consider first the products of cyclic polytopes $\dddot{C}_8(n)$. Figure 3.2 shows a plot of triangulation complexity versus $\mathrm{size}(P)$.

Figure 3.2 Triangulation Complexity, $\dddot{C}_8(n)$.

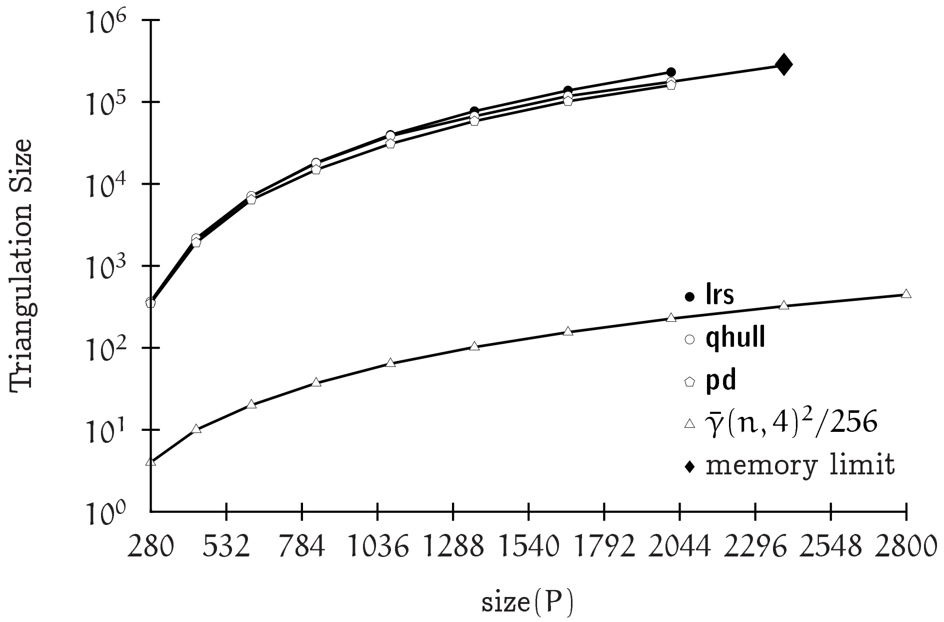


Table 3.1 shows measured triangulation sizes for $\dddot{\Pi}_8(n)$. We can see that even for a polytopes with relatively few vertices and facets, in relatively small fixed dimension (8 in this case), triangulation can be prohibitively expensive. Table 3.2 shows measured triangulation sizes for $\ddot{U}_{3d}$ as d increases. Since our lower bound is less than 3^d, it is far from tight for small d.

Table 3.1 Triangulation size, $\ddot{\Pi}_8(n)$.

n	$f_0(P)$	$c_0(P)$	**lrs**	**pd**	**qhull**
5	100	50	34595	28881	40745
6	144	72	78408	74620	100907
7	196	98	187376	142131	201413
8	256	128	322176	275942	*overflow*
9	324	162	640791	454623	
10	400	200	960200	765906	

Table 3.2 Triangulation size, $\ddot{U}_{3d}$.

d	$f_0(P)$	$c_0(P)$	**lrs**	**pd**	**qhull**
1	8	6	12	12	12
2	16	36	144	144	190
3	128	42	72540	63600	87776
4	256	72	9579672	8818410	*overflow*

3.4.2 Volume Based Bounds

There is ongoing research on the minimum number of simplices necessary to triangulate a d-cube, both in an asymptotic sense and for small d; see e.g. [70, 71] and references therein. Tight lower bounds are known for $\operatorname{tmin} H_d$ for $d \leq 7$, and hence for $\operatorname{tmin}_F H_d$ for $d \leq 8$. In Table 3.3 we compare the size of triangulations computed by various programs with these best possible lower bounds (computed from those given in the paper of Hughes and Anderson [71]). For both of the incremental programs we report only the size of the the triangulated output, rather than summing over all intermediate polytopes. It is interesting to note that the number of simplices for **hull**, and for **lrs** up to dimension 7 is exactly $2d \cdot (d-1)! = 2d!$, which is the largest possible.

Table 3.3 Triangulation size of cubes, versus best possible

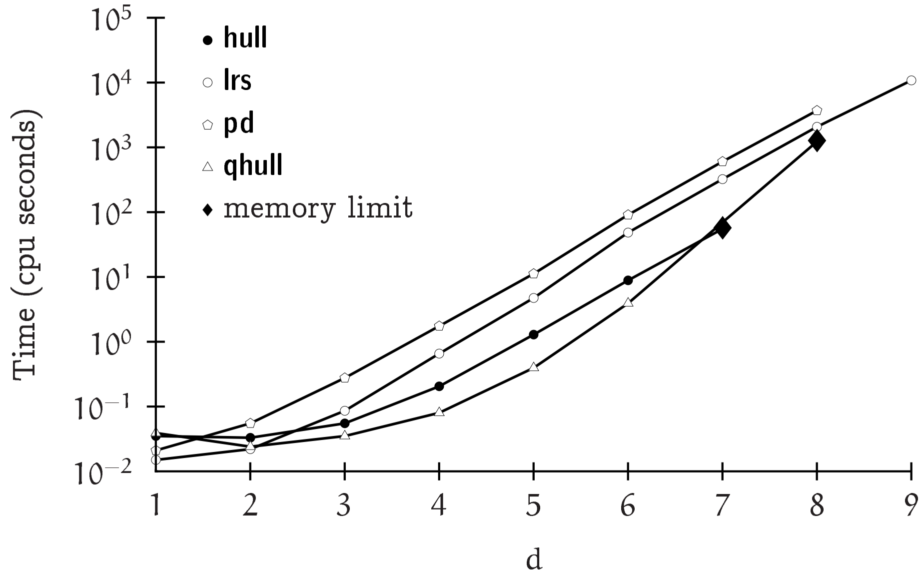
d	best possible	**hull**	**lrs**	**pd**	**qhull**
3	12	12	12	12	12
4	40	48	48	47	44
5	160	240	240	228	210
6	804	1440	1440	1314	1242
7	4312	10080	10080	9038	8472
8	23888	80640	80635	68284	63684

For the products of simplices we know from Section 3.2.1 that every triangulation without extra vertices is exactly the same size. It is therefore encouraging that all four programs tested (**hull**, **lrs**, **pd**, and **qhull**) computed the same size triangulations for runs completed. Figure 3.3 shows the running time in CPU seconds.

Bibliographic Notes

The theorem that every vertex has a basis was first proved by Minkowski [105, 93]. The first published pivoting algorithm known the author [12] includes a discussion of

Figure 3.3 Running time for products of simplices $T_d \times T_d$.



using perturbation to deal with degeneracy. The standard reference for lexicographic perturbation is [17]. A reference probably more useful to implementors is [72].

The observation that truncation polytopes have much larger face lattices than vertex and facet sets occurs in Dyer [19].

Polytopes constructed by taking sums and products of intervals (such as the products of sums of cubes introduced in this chapter) are known in the literature as *Hanner polytopes*. Kalai [96] conjectures that the size of their face lattice, 3^d, provides a lower bound on the size of the face lattice of any *centrally symmetric polytope*, i.e. a polytope P such that $p \in P \Rightarrow -p \in P$.

The proof of Theorem 3.2 given here is mostly based on the one due to Haiman [70]. For general references on unimodular matrices, the reader is referred to the books of Schrijver [109] and Nemhauser and Wolsey [108]. A proof that uses algebraic tools instead of volume is given in [69] and [67].

Bayer [66] defined the class of *weakly neighbourly polytopes*, defined as those where every $k+1$ vertices are contained in a face of dimension at most $2k$, $0 \le k \le d$, and showed that ever triangulation of weakly neighbourly polytope has the same number of full dimensional simplices. It is not difficult to see that products of two simplices are weakly neighbourly, since a vertex (p, q) of $T_d \times T_d$ lies on all of the facet defining hyperplanes that p does, along with all of the facet defining hyperplanes that q does.

Borgwardt [6] considers the expected performance of a gift wrapping algorithm on certain rotation-symmetric distributions of random points. Of course such point sets are in general position, and hence easy for gift wrapping from our point of view.

Chapter 4

Incremental Algorithms

Pivoting algorithms are vulnerable to degeneracy, in the sense that degenerate polytopes can have a number of bases (facet defining simplicies or vertex defining sets of supporting hyperplanes) superpolynomial in the output size. In Chapter 3 we have seen that neither of the known methods for dealing with degeneracy — perturbation and face lattice generation — can yield a polynomial algorithm. The algorithms most widely used for degenerate problems in practice are the incremental algorithms based on the *double description method* of Motzkin et al. [34]. Incremental algorithms are not necessarily affected by degeneracy, but have a fundamental weakness of their own. In an incremental facet enumeration algorithm, after some initialization, we insert points one by one, maintaining the convex hull of the points inserted so far at every step. A necessary condition for such algorithms to be polynomial is that the size of each intermediate polytope be polynomial. It turns out the order the points are inserted can make a huge difference in the size of the intermediate polytopes. This is analogous to the simplex method of linear programming where a family such as the Klee-Minty [100] cubes can be superpolynomial for one pivoting rule but easily solvable using a different pivoting rule.

In the most general sense, insertion orders are procedures to determine at each step of an incremental algorithm, what input element should be processed next. In some cases, such as lexicographic or random ordering, all of the choices can be made before the input is processed. In other cases, such as the maxcutoff rule (where we choose the next element which causes the largest drop in intermediate size), the insertion order is inherently dynamic. In either case, for every input and for every insertion order there are one or more possible permutations of the input generated. We will say that π is a *good* insertion order for a family of polytopes Γ if the size of intermediate polytopes created by π is polynomial for Γ (obviously a much stronger bound is necessary for an insertion order to be "good" in practice). A good insertion order does not by itself guarantee a polynomial algorithm: in particular the use of triangulation can still cause an incremental algorithm to be superpolynomial (see Section 3.2). On the other hand, a naive implementation of the double description method will be polynomial given a good insertion order. Dyer [19] gave a family of polytopes for which inserting the halfspaces in the order given (for vertex enumeration) yields superpolynomially sized intermediate polytopes, most of whose vertices are deleted. In this chapter, we give three extensions and generalizations.

1. Section 4.2 describes families with superpolynomially sized intermediate polytopes *all* of whose vertices are deleted.

2. In Section 4.3 we show that there are families of non-degenerate polytopes for which the largest drop possible in intermediate size (in the sense of the Upper and Lower Bound theorems) occurs for several common insertion orders.

3. In Section 4.4 we show that there are families for which superpolynomial drop in intermediate size occurs for *any* insertion order. This may be contrasted with the situation of the simplex method, where the existence of a polynomial pivoting rule remains an open problem.

We start the chapter by explaining a few of the details of incremental methods, that while not directly necessary to understand the theoretical results in the rest of this chapter, are helpful in understanding some of the experimental results, and possibly of independent interest.

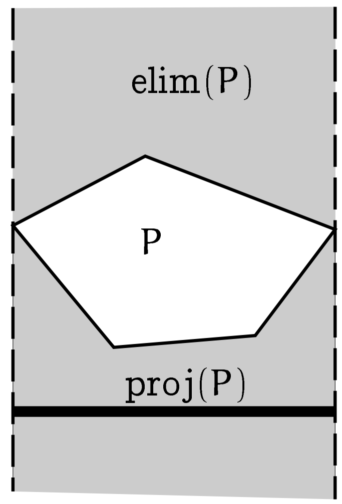
4.1 Initialization, Unboundedness and Elimination

For simplicity, most of thesis takes a rather abstract view of incremental algorithms. There are at least three details that we have been ignoring (and will continue to ignore outside of this section). The first detail is the construction of an initial polytope. The second detail is the possible unboundedness of intermediate polytopes. The third detail is the claimed equivalence of Fourier-Motzkin elimination and incremental construction.

There are two main approaches taken to constructing an initial polytope. We work initially in the context of facet enumeration. The first approach is to find a set of $d + 1$ affinely independent input points and take their convex hull, a d-simplex, as the initial polytope. The other approach is to work inductively on the dimension. Starting with a 0-polytope, with each point we add that lies outside the current affine hull, increase the dimension by one. The first approach has the advantage of simplicity, the second of dealing automatically with input that is not full dimensional.

If we consider the problem of initializing an incremental vertex enumeration algorithm, a subtle difference not covered by the usual tool of duality becomes apparent. While one can always find a set of $d+1$ input halfspaces whose intersection is full dimensional (corresponding to our initial set of affinely independent vertices in the facet enumeration case), it is not always possible to find such a set whose intersection is bounded (consider e.g. a d-cube). Rather than introducing special cases like simplices with a vertex at infinity, the *double description method* [34, 25] homogenizes the input constraints, i.e. for each input constraint $ax \leq b$, one generates a constraint $ax - bx_{d+1} \leq 0$. This gives a polyhedral cone in $\mathbb{R}^{d+1}$ with a unique vertex at the origin (presuming the original polytope was full dimensional). As well as the intersection of a set of inequalities, every polyhedral cone is also the nonnegative hull (set of nonnegative combinations) of its extreme rays, i.e. rays not nonnegative combinations of two other rays in the cone. We can recover the vertices of the original system by setting $x_{d+1} = 1$; in the (for us normal) case where the original system is bounded, this merely requires scaling each extreme ray. We now give a brief sketch of the double description method for finding the extreme rays of a cone. Start by finding some linearly independent set of $d+1$ constraints defining a cone with $d+1$ extreme rays. Suppose we have computed the set of extreme rays for $C_k = \bigcap_{i \le k} h_i^+$ and we wish to compute the extreme rays of $C_{k+1} = C_k \cap h_{k+1}^+$. The extreme rays of C_{k+1} are the extreme rays of C_k feasible for h_{k+1}^+, along with the intersection of 2-faces of P_k with h_{k+1}^0. We can check if two extreme rays are *adjacent*, i.e. lie in a 2-face of C_k, either by computing the rank of the appropriate matrix, or by a simple combinatorial test (see e.g. Proposition 7 in [25]).

Figure 4.1 Elimination and projection



For vector x and $J \subset \mathbb{Z}^+$, let x_J be the vector of elements of x indexed by J. For $X \in \mathbb{R}^{d \times n}$, the kth *orthogonal projection* $\mathrm{proj}_k(X)$ is defined as $\{\, x_{\{1 \ldots d\} \setminus \{k\}} \mid x \in X \,\}$. The kth *elimination* $\mathrm{elim}_k(X)$ is defined as $\{\, x + \lambda \boldsymbol{e}_k \mid x \in X, \lambda \in \mathbb{R} \,\}$. For polyhedron P, note that the facet defining inequalities for $\mathrm{proj}_k(P)$ and $\mathrm{elim}_k(P)$ are the same; it is merely a matter of what space they are interpreted in (see Figure 4.1).

Fourier-Motzkin elimination is a technique first proposed by Fourier [22] to compute the halfspace representation of $\mathrm{elim}_k(P)$ for a polyhedron described by inequalities. We are given a homogeneous system of inequalities $C = \{\, x \mid Ax \ge 0 \,\}$. For rows with nonzero a_{ik}, solve for x_k.

$$
\begin{aligned}
x_k &\geq \frac{(a_i - \mathbf{e}_k a_i)x}{a_{ik}} \equiv l_i \qquad a_{ik} > 0 \\
x_k &\leq \frac{(a_k - \mathbf{e}_k a_j)x}{a_{jk}} \equiv u_j \qquad a_{jk} < 0
\end{aligned} \tag{4.1}
$$

The inequalities $a_i x \geq 0$ where $a_{ik} = 0$, along with inequalities $l_i \leq u_k$ for all possible pairs, form an inequality description of $\text{elim}_k(P)$.

Let P be a d-polytope with vertices $V = \{v_1, v_2, \ldots v_n\}$. The following linear system describes the set of convex combinations of V, (i.e. P).

$$
\begin{aligned}
\sum \lambda_i v_i &= x \\
\lambda_i &\geq 0 \qquad 1 \leq i \leq n \\
\sum \lambda_i &= 1
\end{aligned} \tag{4.2}
$$

We can transform system (4.2) into a homogeneous system of inequalities and apply Fourier-Motzkin elimination to eliminate the λ_i's. If we substitute $\lambda_j = 0$, $k+1 \leq j \leq n$, into (4.2), then we have system of inequalities describing $\text{conv}\{v_1 \ldots v_k\}$. From the definition of the elimination operation, this is still true after $\{\lambda_1 \ldots \lambda_k\}$ are eliminated from (4.2). Since this substitution operation does not introduce any new constraints, it follows that after $\{\lambda_i \ldots \lambda_k\}$ are eliminated, the current set of inequalities must contain each element of $\mathcal{H}(\{v_1 \ldots v_k\})$, lifted into $\mathbb{R}^{d+n-k}$.

Many of the inequalities generated by the elimination procedure described above will be redundant. If we consider the polar interpretation of the inequalities (4.1) as vectors, the combined inequality $l_i \leq u_k$ corresponds taking a nonnegative combination that lies on the polar hyperplane $a_j = 0$. It follows that the generated constraints are non-redundant iff the corresponding vectors (extreme rays) are adjacent. Thus a small variation (inserting hyperplanes, rather than halfspaces) of the double description method can be used to perform elimination.

4.2 Pairs of Piercing Polytopes

In this section we define a set of combinatorial conditions on a pair of polytopes P and Q such that for any pair of polytopes that satisfies these conditions, $\operatorname{size}(P \cap Q)$ is quadratic in $|\mathcal{V}(P)|$ and $\mathcal{H}(P \cap Q) = \mathcal{H}(P) \cup \mathcal{H}(Q)$. We then show the existence of such pairs where $\operatorname{size}(Q)$ is superpolynomial in $\operatorname{size}(P)$. Let $\mathcal{F}(P)$ denote the facets of P and let $\mathcal{E}(P)$ denote the edges of P. Given polytopes P and Q, we say that P *pierces* Q if the following conditions hold:

(4.3a)	$\forall F \in \mathcal{F}(Q)$	$F \cap P \subset \operatorname{relint} F$		
(4.3b)	$\forall F \in \mathcal{F}(Q), \exists e \in \mathcal{E}(P)$	$\operatorname{relint}(e) \cap F \neq \emptyset$	and	$e \not\subset \operatorname{aff} F$
(4.3c)	$\forall e \in \mathcal{E}(P), \exists F \in \mathcal{F}(Q)$	$\operatorname{relint}(e) \cap F \neq \emptyset$	and	$e \not\subset \operatorname{aff} F$

If P pierces Q then we claim the following:

$$\mathcal{V}(P \cap Q) = (\mathcal{V}(P) \cap Q) \cup \bigcup_{e \in \mathcal{E}(P)} (e \cap \partial Q) \tag{4.4}$$

Certainly every $v \in (\mathcal{V}(P) \cap Q)$ is a vertex of $P \cap Q$. For any edge $e \in \mathcal{E}(P)$, by condition (4.3c) we know that $\exists x \in \operatorname{relint}(e) \cap \partial Q$, and x must also be a vertex of $P \cap Q$. It remains to show that these are all of the vertices of $P \cap Q$. Consider some basis B (set of d-affinely independent facets) for a vertex of $P \cap Q$. By (4.3a), B contains at most one facet of Q. But then it must contain at least $d - 1$ facets of P, and (4.4) follows.

It is easy to construct redundant families of inequalities difficult for (naive) insertion algorithms, e.g. by placing a small simplex inside a dual cyclic polytope. Since it is relatively easy to remove redundant constraints using linear programming, these families are not really difficult, hence we would like our example families to be non-redundant. If P pierces Q, then we claim the following:

$$\mathcal{H}(P \cap Q) = \mathcal{H}(P) \cup \mathcal{H}(Q)\,. \tag{4.5}$$

Here it is trivial that $\mathcal{H}(P \cap Q) \subseteq \mathcal{H}(P) \cup \mathcal{H}(Q)$. It remains to show that each $H \in \mathcal{H}(P)$ and each $H \in \mathcal{H}(Q)$ is non-redundant. We start with $\mathcal{H}(Q)$. For convenience, we define some notation. Let $\text{ball}(x, r)$ ($\text{sphere}(x, r)$) denote the d-ball (d-sphere) of radius r, centered at x. Let $P \boxminus H$ denote $\bigcap(\mathcal{H}(P) \setminus H)$. Let F_q be some facet of Q. From condition (4.3b) and condition (4.3a), we know there is some edge e of P such that $x \equiv \text{relint}(e) \cap \text{relint}(F_q) \neq \emptyset$. There must be some $\epsilon > 0$ such that $e \cap \text{ball}(x, \epsilon) \subset \text{relint}(e)$ and $\text{ball}(x, \epsilon) \subset \text{int}(Q \boxminus F_q^+)$. We know one of the vertices w of e must be outside Q. Define $x' = \overline{xw} \cap \text{sphere}(x, \epsilon)$. By construction $x' \in \text{relint}(e) \cap \text{int}(Q \boxminus F_q^+) \cap F_q^-$. It follows that $x' \in (P \cap Q) \boxminus F_q^+ \cap F_q^-$, hence F_q^+ is not redundant.

We now argue that no $H \in \mathcal{H}(P)$ is redundant either. Let $F_p \in \mathcal{F}(P)$. Let $e \in \mathcal{E}(F_p)$. By condition (4.3c) and condition (4.3a), we know there is some facet F_q of Q such that $x \equiv \text{relint}(e) \cap \text{relint}(F_q) \neq \emptyset$. Let ϵ be defined as above. Let x' be some point in $[\text{int}\,\text{ball}(x, \epsilon)] \cap F_q \cap \text{relint}(F_p)$. By considering some point along the outward normal of F_p from x' close to x', we see that F_p^+ is not redundant either.

4.2.1 Examples of Piercing Pairs

We now show the existence of families of piercing pairs of polytopes. Our example family will achieve a superpolynomial drop in intermediate size by piercing a cube with a stacked polytope. We start by defining a family of stacked d-polytopes with $2d + 1$ vertices. Let $\overline{T}_d$ be the polytope defined by the constraints

$$x_i \geq 0 \qquad 1 \leq i \leq d \tag{4.6a}$$

$$\mathbb{1}x \leq 1 \tag{4.6b}$$

$$x_i \leq 3/5 \qquad 1 \leq i \leq d\,. \tag{4.6c}$$

Lemma 4.1 *$\overline{T}_d$ is a truncation polytope with $2d+1$ facets, and d^2+1 vertices consisting of the origin $\mathbb{O}$, along with $(3/5)\boldsymbol{e}_i + (2/5)\boldsymbol{e}_j$, for $i > 0$, $j \neq i$.*

Proof. The constraints (4.6a) and (4.6b) define a simplex T_d with vertices $\boldsymbol{e}_0 \ldots \boldsymbol{e}_d$. Consider the new vertices introduced by intersecting T_d and one of the constraints (4.6c), $h_i^+ : x_i \leq 3/5$. The only vertex of T_d infeasible for h_i^+ is $\boldsymbol{e}_i$, thus the new vertices are convex combinations of $\boldsymbol{e}_i$ and some $\boldsymbol{e}_j$. Since the new vertices lie on the hyperplane $x_i = 3/5$, it follows that they have coordinates $(3/5)\boldsymbol{e}_i + (2/5)\boldsymbol{e}_j$. Since the new vertices strictly satisfy every other inequality (4.6c), any ordering of the constraints (4.6c) produces a truncation order for T_d. □

Let $\widehat{T}_d$ denote the polyhedron defined by the following constraints:

$$\begin{aligned} \mathbb{1}x &\geq -d && \\ (3d\boldsymbol{e}_i - \mathbb{1})x &\leq d && 1 \leq i \leq d \\ (3d\boldsymbol{e}_i + 2d\boldsymbol{e}_j - \mathbb{1})x &\leq d && \{i,j\} \subset \{1\ldots d\} \end{aligned}$$

Lemma 4.2 *$\widehat{T}_d$ is a stacked polytope dual to $\overline{T}_d$, with vertices $\mathbb{1}/4$ and $\{-d\boldsymbol{e}_i, \frac{d}{3d-1}\boldsymbol{e}_i\}$, $1 \leq i \leq d$.*

Proof. Consider the polyhedron defined by the following constraints:

$$\begin{aligned} -1/d \leq x_i &\leq -1/d + 3 && 1 \leq i \leq d \\ \mathbb{1}x &\leq 4 && \end{aligned} \tag{4.7}$$

The reader can verify that this polytope is obtained by scaling $\overline{T}_d$ by 5 and translating by $-\mathbb{1}/d$, hence it has vertices $-\mathbb{1}/d$, $3\boldsymbol{e}_i - \mathbb{1}/d$, and $3\boldsymbol{e}_i + 2\boldsymbol{e}_j - \mathbb{1}/d$. $\widehat{T}_d$ is the polar of the polytope defined by constraints (4.7). □

We conclude the section by arguing that there exist polytopes with few facets and many vertices (namely certain cubes) pierced by the stacked polytopes $\widehat{T}_d$.

Theorem 4.1 *For* $d \geq 3$, H_d *is pierced by* $\widehat{T}_d$, *where* H_d *is the hypercube defined by*

$$(U_i) \qquad x_i \leq 1/3 \qquad 1 \leq i \leq d$$

$$(L_i) \qquad x_i \geq -(d-1) \qquad 1 \leq i \leq d\,.$$

Proof. We consider the intersection each bounding hyperplane of H_d with $\widehat{T}_d$, and show that such an intersection is in the relative interior of the corresponding facet of H_d. It suffices to consider the intersection of the bounding hyperplane with the edges of $\widehat{T}_d$.

Consider a bounding hyperplane defined by U_i. There is exactly one vertex of $\widehat{T}_d$ infeasible for U_i, namely $s_i = \frac{d}{3d-1}\boldsymbol{e}_i$. From the fact that $\widehat{T}_d$ is a stacked polytope, it follows that s_i is adjacent to (i.e. shares an edge with) vertices $\mathbb{1}/4$ and $-d\boldsymbol{e}_j$, $j \neq i$. The first case is easy, since both $\mathbb{1}/4$ and s_i strictly satisfy every constraint of H_d other than than U_i; thus any point on that edge must lie interior to the $(d-1)$-cube defined by U_k, L_k, $k \neq i$. Let x be the intersection of the edge between s_i and $-d\boldsymbol{e}_j$ and the hyperplane $x_i = 1/3$. From the definition of a convex combination, we have

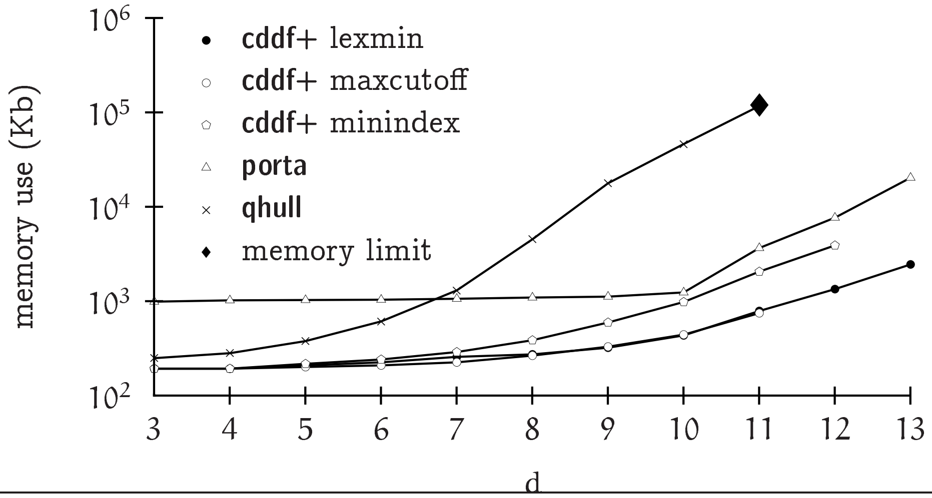
$$\begin{aligned} x = \lambda s_i + (1-\lambda)(-d\boldsymbol{e}_j) \qquad &= \frac{3d-1}{3d}s_i + \frac{-d\boldsymbol{e}_j}{3d} \qquad \text{since } \lambda s_i = 1/3\boldsymbol{e}_i \\ = (\boldsymbol{e}_i - \boldsymbol{e}_j)/3\,. & \end{aligned}$$

x is also in the relative interior of the appropriate facet.

The case of bounding hyperplanes defined by L_i is analogous. The vertex infeasible for L_i is $-d\boldsymbol{e}_i$. It is adjacent to $\mathbb{1}/4$ (this case is again easy), $-d\boldsymbol{e}_j$, and $d/(3d-1)\boldsymbol{e}_j$, $j \neq i$. The intersections between these edges and the defining hyperplane for L_i have the form $(1-1/d)s_i + (1/d)s_j$. In the second case, we have x_j coordinate -1 which is greater than $-(d-1)$ for $d > 2$, and in the third case we have x_j coordinate $1/(3d-1)$, which is less than $1/3$ for $d > 1$.

We have seen that conditions (4.3a) and (4.3b) are satisfied. To see that (4.3c) is also satisfied, we note that every edge of $\widehat{T}_d$ contains a vertex outside H_d, and that no vertex of $\widehat{T}_d$ is on the boundary of H_d. □

Figure 4.2 Pierced cubes, memory use in kilobytes.



The pierced cubes (i.e. $H_d \cap \widehat{T}_d$) are difficult for insertion algorithms in much the same way as the earlier examples of Dyer [19]: if an insertion algorithm inserts the constraints in the order given, then an adversary can force the construction of the d-cube as an intermediate polytope, which has a superpolynomial number of vertices in the final output size. They differ from the earlier examples in that all of the vertices of the large intermediate polytope are infeasible for the final output.

4.2.2 Experimental Results

The lower bounds of this chapter are lower bounds on the space used by particular classes of algorithms. Such bounds of course imply corresponding bounds on time complexity, but from a practical point of view are even stronger, since waiting twice as long is often more practical than doubling the available memory. In this section we consider the space usage of three incremental facet enumeration programs. **cdd+** [47] is an implementation of the double description method, using either floating point (**cddf+**) or exact arithmetic (**cddr+**). **porta** [44] is an implementation of Fourier-Motzkin elimination using exact arithmetic. **qhull** is an incremental algorithm using floating point and triangulation.

In Figure 4.2 we compare the memory usage of **cddf+**, **porta**, and **qhull** for the

Figure 4.3 Pierced cubes, summed intermediate size.

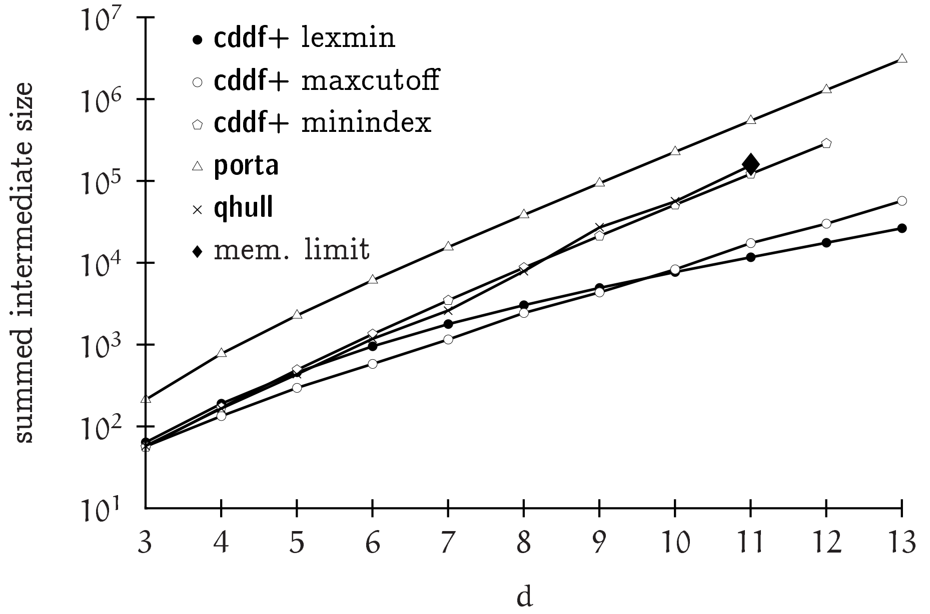
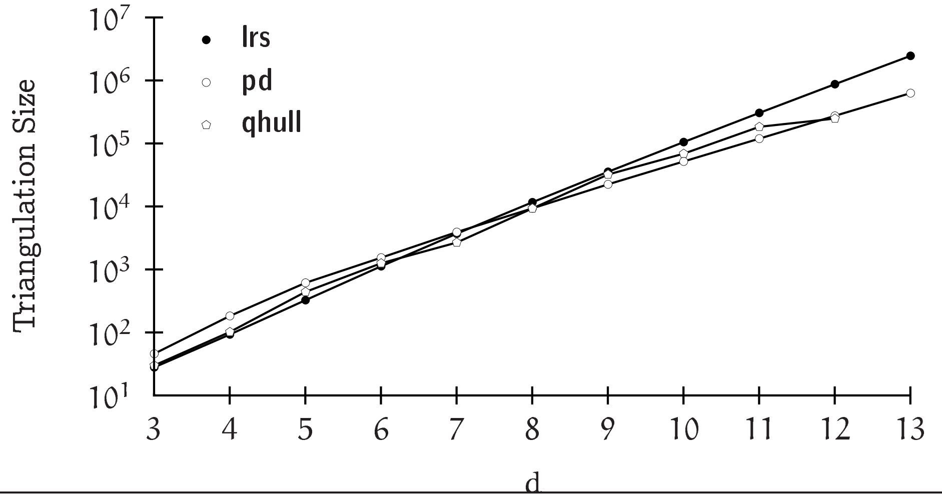


Figure 4.4 Pierced Cubes, triangulation size.



pierced cube examples. These measurements are on a Digital AlphaServer 4/233 with 256M of real memory and 512M of virtual memory. To prevent problems with thrashing, we have imposed an arbitrary limit of 128 megabytes of working space. The notation *memory limit* indicates the last run of series able to complete due to this limit. Since size(P) is polynomial in the dimension, the super-linear curves in Figure 4.2 are suggestive of memory usage superpolynomial in size(P) (since c^d would be a line on a logarithmic plot).

For incremental algorithms, a machine independent measure of the work done is the sum of sizes of the intermediate polytopes created. In Figure 4.3 we compare this statistic for **cddf+**, **porta**, and **qhull**. The number reported by **porta** is the sum of intermediate sizes over all iterations; thus an inequality is counted once for each iteration that it exists, rather than just once when created. **qhull** and **cddf+** report the total number of hyperplanes created over all iterations. Note that **qhull** stores the triangulation of each intermediate facet, thus even when the intermediate polytopes are smaller (in terms of vertices) than those produced by **cdd+**, **qhull** uses significantly more memory. The number of intermediate rays created by **cddr+** is identical to the number created by **cddr+**, suggesting that the pierced cube examples are reasonably numerically stable.

Although the pierced cube examples were not designed to be difficult for triangulation/perturbation based algorithms, experimentally it seems that they are. Figure 4.4 shows triangulation size for **lrs**, **pd**, and **qhull**. Note that in the case of **pd**, this is the size of the triangulation of the dual polytope (see Chapter 5).

4.3 Dwarfed Polytopes

In this section, rather than intersecting a large intermediate polytope with another polytope, we intersect with a single halfspace. In order to prevent redundancy, we must preserve a small number of the vertices of our intermediate polytope. Nonetheless, the drop in intermediate size achieved will be asymptotically larger than in the previous section. A halfspace of $\mathbb{R}^d$ is said to *dwarf* a d-polytope P with m facets if $P \cap H$ is a simple polytope with $m+1$ facets and

$$\beta(m+1, d) = (d-1)(m+1) - (d+1)(d-2) \tag{4.8}$$

vertices; in this case we say that $P \cap H$ is a *dwarfed polytope*. Note that (4.8) is the minimum number of vertices for a simple polytope with $m+1$ facets permitted by the Lower Bound Theorem. It follows that for $d > 3$, dwarfed polytopes are truncation polytopes, and have at at least one good insertion order, namely the truncation order. In this section we will provide example families of dwarfed polytopes where $f_0(P)$ is superpolynomial in $f_0(P \cap H)$, and show that there are several natural insertion orders that inserted the dwarfing halfspace last (or late).

Before proving the main result of this section, we will need two preliminary results. The first is characterization of the vertices and facets of the polytope formed by intersecting a polytope with a single halfspace. Let P be a d-polyhedron and H a halfspace in $\mathbb{R}^d$ with bounding hyperplane h such that $h \cap P \neq \emptyset$ but h contains no vertex of P. We argued in Section 4.2 that by dimension considerations,

$$\mathcal{V}(P \cap H) = (\mathcal{V}(P) \cap H) \cup \bigcup_{e \in \mathcal{E}(P)} e \cap h\,. \tag{4.9}$$

The following is a consequence of the definition of a facet.

$$\mathcal{F}(P \cap H) = \{P \cap h\} \cup \bigcup_{F \in \mathcal{F}(P)} F \cap H \tag{4.10}$$

Let P be a polytope, and H an open halfspace. We call a vertex v of P *surviving* if $v \in H$. We call an edge surviving if both of its vertices are surviving.

Lemma 4.3 *Let* P *be a polytope and* H *an open halfspace. The graph of surviving vertices and edges of* P *w.r.t.* H *is connected.*

Proof. Define a linear program with the constraints $\mathcal{H}(P) \cup \{H\}$ and an objective function of the inward normal of H. Every vertex of $P \cap H$ is either a surviving vertex or contained in the bounding hyperplane of H. From the correctness of the simplex method with Bland's pivot rule (see e.g. [85]), there is a path in the skeleton of P from any surviving vertex to a unique optimum face F of $P \cap H$. Since the simplex method monotonically increases the value of the objective function (i.e. the distance from the bounding hyperplane of H), this path does not intersect the bounding hyperplane of H, hence is entirely contained in surviving edges. By Balinski's Theorem (see e.g. [80]), the skeleton of F is connected and the lemma follows. $\square$

We are now ready to prove the central result of this section.

Lemma 4.4 *Let P be a simple d-polytope with m facets. Let H be halfspace with no vertex of P on its boundary. If the surviving vertices and edges form a tree with t nodes, $P \cap H$ has $(d-1)t+2$ vertices and $d+t$ facets.*

Proof. We start by showing that $P' = P \cap H$ has $(d-1)t+2$ vertices. Call an edge of P a *cut edge* if it is adjacent to exactly one surviving vertex, i.e. if it intersects the bounding hyperplane of H. Since there are t surviving vertices, it suffices by (4.9) to show that there are $(d-2)t+2$ cut edges. P is simple, so the total number of adjacencies between edges of P and surviving vertices is dt. The surviving edges form a tree, so $2(t-1)$ of the adjacencies are consumed by surviving edges. This leaves $dt-2(t-1)$ adjacencies to cut edges, each of which generates a new vertex of $P \cap H$.

We now argue that P' has $d+t$ facets. By (4.10) it suffices to show that $d+t-1$ facets of P contain surviving vertices. Let T be the tree of surviving vertices and edges. Consider T as a "pivot tree" where each edge corresponds to a pivot, interchanging exactly one incident facet. Root T at some arbitrary node. Label each edge of T with the entering facet, i.e. the facet incident to the child but not to the parent. For any node v in the tree, every facet incident to that node must either be incident to the root, or have entered the set of incident facets at some step on the path from the root to v. It follows that there are at most $d+(t-1)$ facets incident to surviving vertices. By Lemma 4.3, the surviving vertices and edges in a given facet form a connected subgraph (i.e. a subtree) of T (here F is considered as a polytope in its own right; notice that Lemma 4.3 does not depend on the polytope being full dimensional). It follows that for a facet F, every path from the root that enters F must enter at the same vertex, hence each edge label in the tree is unique. It follows that that there are exactly $d + t - 1$ facets incident to surviving vertices. □

The following is then a straightforward consequence of the previous lemma.

Theorem 4.2 (Dwarfing Theorem) *If P is a simple d-polytope with m facets and H is a halfspace with no vertex of P on its boundary so that the surviving vertices and edges form a tree with $m + 1 - d$ nodes, then H is a dwarfing halfspace for P.*

The following illustrates an easy application of the Dwarfing Theorem.

Theorem 4.3 (Dwarfed Cubes) *Let H_d be the d-cube specified by the $2d$ constraints $0 \leq x_i \leq 3/5$ for $1 \leq i \leq d$. H_d is dwarfed by the halfspace $h^+ \equiv \{\, x \mid \mathbb{1}x \leq 1 \,\}$.*

Proof. The surviving vertices are the origin, and those vertices adjacent to the origin. The theorem then follows from the Dwarfing Theorem. The theorem can also be proved directly, since $H_d \cap h^+$ is the the polytope $\overline{T}_d$ from Lemma 4.1. □

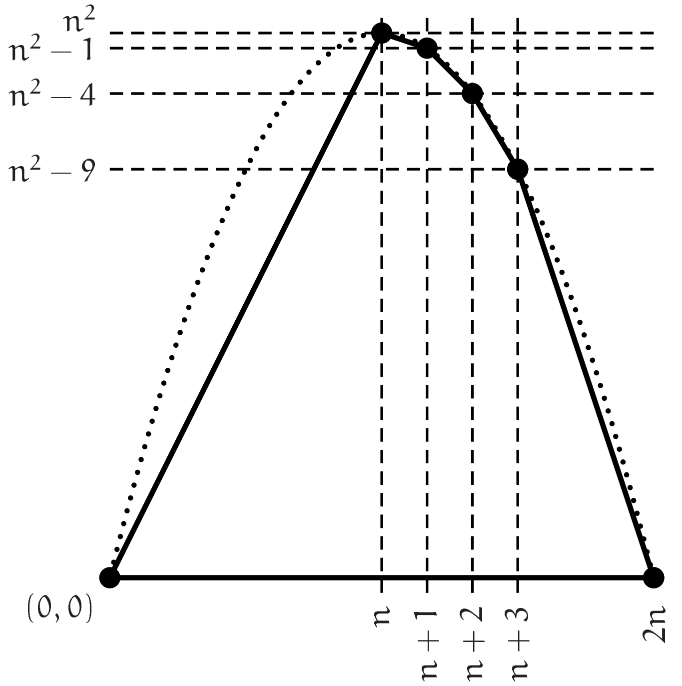
In retrospect, the pierced cube examples of Section 4.2.1 are the intersection of dual dwarfed cubes (i.e. "dwarfed cross polytopes") and cubes. Next we show that for simple polytopes, the highest drop in size permitted by the Upper and Lower Bound theorems can indeed occur.

Theorem 4.4 (Dwarfed Dual Cyclic Polytopes) *For every m-facet d-polytope dual to a cyclic polytope there is a dwarfing halfspace.*

Proof. Let P be a polytope dual to a cyclic polytope $C_d(m)$. Suppose P has a 2-face F that is a polygon with $m+2-d$ vertices. Let ν be one of those vertices and let R be the remaining $m + 1 - d$ vertices. Let ℓ be a line in $\operatorname{aff} F$ that separates ν from R. By perturbing $h_F = \operatorname{aff} F$ slightly we can find a hyperplane h containing ℓ that separates R from $\mathcal{V}(P) \setminus R$. Let H be the closed halfspace bounded by h that contains R. Then by construction there are $m+1-d$ surviving vertices strung together into a path by $m-d$ surviving edges (the boundary of the polygon F without v and its two incident edges). Now apply the Dwarfing Theorem.

It remains to show that polytope P has such a 2-face F with $m-d+2$ vertices. It turns out that P has many such faces (actually $\gamma(d-3, m)$ of them). Let Q be the cyclic polytope that is the dual of P with vertices $p_1, p_2, \ldots, p_m$ in their natural order along the curve used to generate Q. Let $U = \{p_1, \ldots, p_{d-2}\}$. Then, according to Gale's evenness condition (see Section 2.2) for each of the $m+1-d$ indices i with $d-2 < i < m$, the set $U \cup \{p_i, p_{i+1}\}$ spans a facet of Q; moreover, $U \cup \{p_{d-1}, p_m\}$ spans a facet of Q, and this yields all facets containing U. Thus U spans a $(d-3)$-face F^* of Q that is contained in $m+2-d$ facets. Taking the dual we thus get a 2-face F of P that contains $m+2-d$ vertices. $\square$

Figure 4.5 The polygon W_6.



Since cyclic polytopes are difficult for many implementations for numerical reasons independent of intermediate size, we consider numerically better behaved (but non-cyclic) family with about the same dwarfing behaviour. Let W_n be convex polygon with vertices as follows:

$$\begin{aligned} x(i) &\in \{0, n, n+1, n+2, \ldots, 2n-3, 2n\} \\ y(i) &= n^2 - (x(i) - n)^2 \end{aligned} \tag{4.11}$$

Figure 4.5 illustrates W_6. The defining halfplanes for W_n are as follows:

$$\begin{aligned} -nx + y &\leq 0 \\ (2i+1)x + y &\leq (n+i)(n+i+1) && 0 \leq i \leq n-4 \\ (2n-3)x + y &\leq 2n(2n-3) \\ y &\geq 0 \end{aligned}$$

Let $\dddot{W}_{2\delta}(n)$ denote the δ-fold product W_n^δ. From the Product Lemma, we know $\dddot{W}_{2\delta}(n)$ is a simple polytope with dimension 2δ, δn facets, and n^δ vertices, where the vertices $\dddot{W}_{2\delta}(n)$ are the products of vertices of W_n and the facets are defined by the facet (edge) defining inequalities of W_n, lifted into $\mathbb{R}^{2\delta}$.

Theorem 4.5 (Dwarfed Products of Polygons) *For $\delta \geq 2$, $\dddot{W}_{2\delta}(n)$ is dwarfed by the constraint $x_1 + x_2 + \ldots + x_\delta \leq 2n - 1$ where $\mathbb{R}^{2\delta}$ is coordinatized $(x_1, y_1, x_2, y_2 \ldots)$.*

Proof. By simple arithmetic, we can see that no vertex of $\dddot{W}_{2\delta}(n)$ with more than one non-zero x_i coordinate, or with $x_i = 2n$ for some i, will survive. Consider some surviving vertex v. By possibly reordering coordinates, we may assume that $v = (\mathbb{O}^{2\delta-2}, x(i), y(i))$ where $x(i)$ and $y(i)$ are defined by (4.11). By the Product Lemma, the edges of $\dddot{W}_{2\delta}(n)$ are the product of edges of W_n and vertices of $\dddot{W}_{2(\delta-1)}(n)$. Hence v is adjacent exactly to surviving vertex (vertices) $v' = (\mathbb{O}^{2\delta-2}, x(j), y(j))$ where j is the index before or after i. It follows that the surviving edges form δ paths emanating from the origin with $n-2$ edges each, one along each "x-coordinate" direction. Thus the surviving vertices and edges form a tree with $\delta(n-2)+1 = m - d + 1$ nodes. Now apply the Dwarfing Theorem. $\square$

It should be clear that for any algorithm that inserts halfspaces online, an adversary merely has to present the dwarfing halfspace last in order to force superpolynomial performance. We now consider how to use the families defined in this section to argue that more sophisticated insertion orders are superpolynomial. The next theorem shows that for many polytopes, ordering the halfspaces at random is nearly as bad as the worst case. Let $E\{X\}$ denote the expectation of random variable X.

Theorem 4.6 *Let* $P = \bigcap H$ *be a* d*-polytope and let* H' *be a set of* k *halfspaces of* $\mathbb{R}^d$ *such that* $H \cap H' = \emptyset$. *Let* Q *be the polyhedron formed by intersecting the halfspaces appearing before any element of* H' *in a random ordering of* $H \cup H'$.

$$E\{c_j(Q)\} \geq c_j(P)\frac{(j+1)!}{(j+k+1)!} .$$

Proof. Every face of codimension j must be the intersection of at least one j-*basis* of $j+1$ facet defining halfspaces. If a j-basis B of some face of P occurs before any halfspace of H', then it will also define a face of Q. The probability that any such basis B from H occurs before any halfspace from H' is the probability that in a permutation of $B \cup H'$, B occurs at the beginning, i.e. $|B|!/|B \cup H'|!$. □

Note that for dwarfed polytopes, we have

$$E\{\mathrm{size}(Q)\} \geq \mathrm{size}(P)/(d+1) .$$

Another well known insertion order is to insert the halfspaces in lexicographic or reverse lexicographic order. If the constraints are not reduced to some canonical form before ordering then it is trivial for an adversary to enforce whatever ordering is desired by scaling the constraints. Two natural canonical forms for coefficient vectors are *unit form* where the most significant (i.e. highest coordinate index) nonzero entry is 1 (considering the right hand side as a $(d+1)$st entry) and *reduced integer form* where the coefficients are relatively prime integers. The following lemma shows that these two are essentially equivalent for the purposes of lower bounds.

Lemma 4.5 *For any inequality system $Ax \le b$, $A \in \mathbb{Q}^{m\times d}$, $b \in \mathbb{Q}^m$, it is possible to translate and scale the the coordinate system so that the unit form and the reduced integer form of the transformed inequality system are identical.*

Proof. Assume without loss of generality that $P \equiv \{\, x \mid Ax \le b \,\}$ is full dimensional. Let z be some interior point of P. Let P' denote the translation of P by $-z$. Since $\mathbb{0} \in \operatorname{int} P'$, we can write constraint i describing P' in the form $\sum_j (\alpha_{ij}/\beta_{ij})x_j \le 1$ where the α_{ij}'s are integers and the β_{ij}'s are positive integers. For each j, let δ_j denote the least common multiple of the β_{ij}'s. Let $u_{ij} = (\alpha_{ij}/\beta_{ij}) \cdot \delta_j$ and $y_j = x_j/\delta_j$. Then $Uy \le \mathbb{1}$ is affinely equivalent to $Ax \le b$, and furthermore is in both reduced integer and unit form. □

It will be slightly more convenient to work with unit form. We call the lexicographic order after reduction to unit form *unit lexicographic order*.

Theorem 4.7 *Let $P = \{\, x \mid Ax \le \mathbb{1} \}$, be a d-polytope containing the origin in its interior. There exists a rigid motion that places any fixed row of the transformed system first or last in unit lexicographic order, as desired.*

Proof. Consider hyperplane h_i^0. If $h_{i1} \neq 0$, h_i^0 intersects the x_1 axis at the point e_1/h_{i1}. Since the origin is in the interior of P, both the positive and negative x_1 axes must intersect some bounding hyperplane of P. The first hyperplane intersected by the positive x_1 axis will be the lex maximum one (i.e. largest positive a_{i1}) and the first intersected by the negative x_i axis will the lex minimum one (ignoring ties).

Now suppose we wish to make some hyperplane h_i^0 lex minimum or maximum. Let F_i be the corresponding facet. Choose some x_j such that the x_j axis is not parallel to h_i^0. Reorder coordinates so that x_j is x_1. Let ℓ be line parallel to the x_j axis intersecting the relative interior F_i. Let u be some point in $\operatorname{int}(P) \cap \ell$. Translate so that u is the origin and ℓ the x_1 axis. From the discussion proceeding, we see that (the transformed) h_i^0 is now either lex minimum or lex maximum. We can switch between these two options by swapping the positive and negative x_1 axes.

Note that since ℓ intersects the relative interior of F_i, lexicographic order is entirely determined by the first coordinate, i.e. there are no ties. □

4.3.1 Experimental Results

Figure 4.6 Summed intermediate size, dwarfed cubes

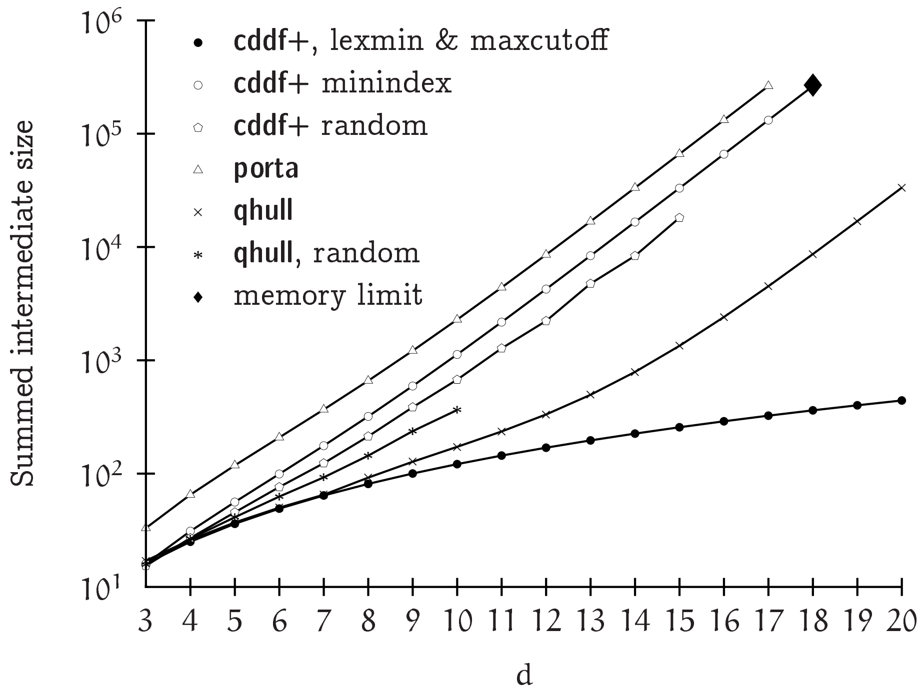
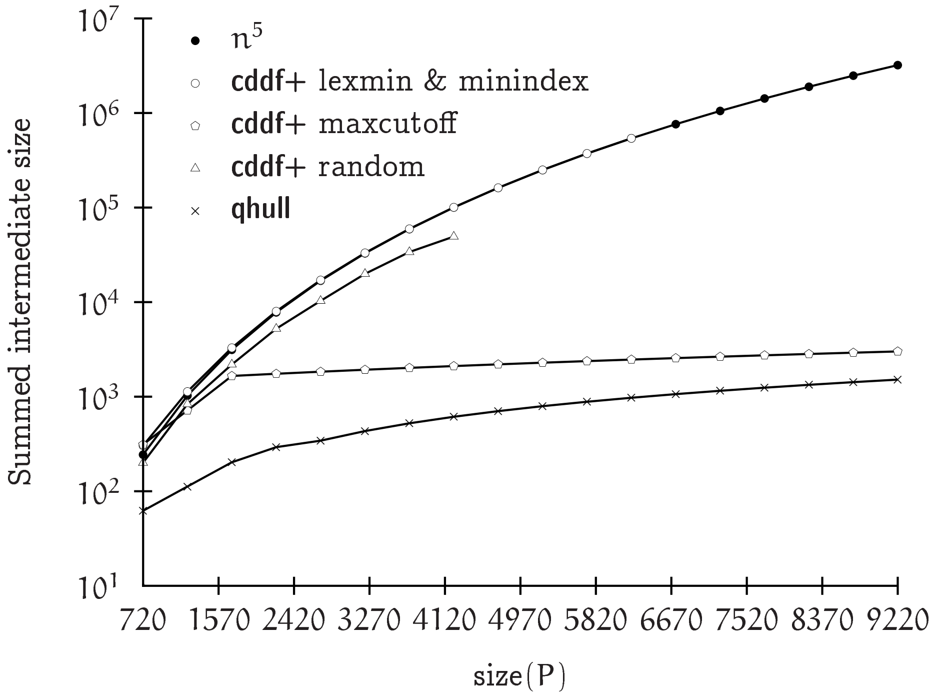


Figure 4.6 plots summed intermediate size for **cddf+**, **porta**, and **qhull** on dwarfed cubes. On completed runs, **cddr+** produces the same intermediate sizes as **cddf+**. Random insertion (the numbers here are averaged over 100 runs) behaves badly for dwarfed cubes as predicted by Theorem 4.6. This means that in a probabilistic sense, most insertion orders are bad. On the other hand, common heuristics such as lexicographic ordering and maxcutoff seem to work quite well. The “farthest outside” heuristic of **qhull** also works significantly better than random ordering. It turns out that maxcutoff degenerates into lexmin on these examples by a quirk of initialization. When using a dynamic insertion order such as maxcutoff, the initial simplex must be chosen by some other method (since initially the maxcutoff rule

Figure 4.7 Summed intermediate size, 10 dimensional dwarfed products of polygons



does not make any sense). In the case of **cdd+**, the initial simplex is chosen lexicographically. After the initial simplex is chosen, then each remaining hyperplane cuts off the same number (1) of vertices, and ties are broken lexicographically. Since at every step we have truncation polytope, lexmin and maxcutoff (given lexicographic insertion orders) are polynomial insertion orders for dwarfed cubes.

We consider the growth of intermediate size for dwarfed products of polygons with the dimension held fixed and the number of input halfspaces varying. Figure 4.7 shows a plot of summed intermediate size versus summed input and output size. The results for dwarfed cubes and dwarfed products of polytopes support the intuition that maxcutoff ordering behaves well for dwarfed polytopes, since it will tend to insert the dwarfing halfspace early.

4.4 Families Hard for All Insertion Orders

In the previous section, we have seen that there exist families of polytopes where the insertion of one particular "dwarfing" halfspace last causes a huge drop in intermediate size. Here we consider the natural extension to the case where every input halfspace causes a huge drop in intermediate size when inserted last. This will imply a superpolynomial lower bound for *any* insertion order. It will convenient in this section to consider the dual situation for facet enumeration where every vertex causes a huge drop in the number of intermediate facets when inserted last.

4.4.1 The Main Result

At each step of an incremental facet enumeration algorithm, we maintain (at minimum) the vertex description V_i and the halfspace description H_i of the current intermediate polytope (the "double description" of Motzkin et al. [34]). We are interested here in the drop in the size of the halfspace description caused by inserting the last vertex. This is the same as the increase in the number of facets caused by removing one vertex of a polytope and recomputing the convex hull of the remaining vertices. In the following definitions, let P be a d-polytope and let v be a vertex of P. Let $P \ominus v$ denote $\operatorname{conv}(\mathcal{V}(P) \setminus \{v\})$. Let $\mathcal{F}(P)$ denote the facets of P, $\mathcal{F}_v(P)$ the facets of P containing v, and let $\widetilde{\mathcal{F}}_v(P)$ be defined as follows:

$$\widetilde{\mathcal{F}}_v(P) \equiv \begin{cases} \{F \in \mathcal{F}(P \ominus v) \mid v \in F^-\} & \text{if } \dim(P \ominus v) = d; \\ \{P \ominus v\} & \text{otherwise.} \end{cases}$$

We define $\operatorname{loss}(v, P) \equiv |\widetilde{\mathcal{F}}_v(P)|$ as the number of halfspaces deleted by inserting v last[1]. Similarly, we define $\operatorname{gain}(v, P) \equiv |\mathcal{F}_v(P)|$ as the number of halfspaces created by inserting v last. The net drop in intermediate size is then $\operatorname{drop}(v, P) = \operatorname{loss}(v, P) - \operatorname{gain}(v, P)$. Finally we define $\operatorname{gain}(P) \equiv \max_v \operatorname{gain}(v, P)$, $\operatorname{loss}(P) \equiv \min_v \operatorname{loss}(v, P)$,

[1] we make the notation simplifying assumption that in the case where $\dim(P \ominus v) = \dim(P) - 1$, the affine hull of $P \ominus v$ is stored as two halfspaces

and $\mathrm{drop}(P) \equiv \min_v \mathrm{drop}(v, P)$. If there is a vertex whose removal decreases the number of facets (as in for example a stacked polytope) then $\mathrm{drop}(P)$ is negative.

The lower bounds in this section follow from using the product and sum of polytopes constructions defined in Section 2.4. The central geometric observation is that while the number of facets of the final polytope sums under the product of polytopes operation, the number of intermediate facets multiplies. We call a polytope P *robust* if $\dim(P \ominus v) = \dim P$ for every vertex v of P.

Theorem 4.8 *Let P and Q be polytopes with dimension at least 2. $P \times Q$ is robust and* $\mathrm{drop}(P \times Q) = \mathrm{loss}(P) \cdot \mathrm{loss}(Q)$.

We prove Theorem 4.8 via several lemmas. Since $\mathrm{gain}(v, P)$ is non-negative, the following holds: $\mathrm{loss}(P) - \mathrm{gain}(P) \leq \mathrm{drop}(P) \leq \mathrm{loss}(P)$. Let P and Q be polytopes with dimension at least 2. Theorem 4.8 follows from the following two facts.

$$\mathrm{gain}(P \times Q) = 0 \tag{4.12}$$

$$\mathrm{loss}(P \times Q) = \mathrm{loss}(P) \cdot \mathrm{loss}(Q) \tag{4.13}$$

We start with (4.12), which is equivalent to saying that every facet of $P \times Q$ is robust. Since each facet of $P \times Q$ is the product of a facet of P (respectively P) and Q (respectively a facet of Q), this follows from the next lemma.

Lemma 4.6 *Let P and Q be polytopes with $\dim P \geq 1$ and $\dim Q \geq 1$. $P \times Q$ is robust.*

Proof. Let $v = (p, q)$ be a vertex of $P \times Q$. Let P' denote $(P \times Q) \ominus v$:

$$P' = \mathrm{conv}(\{\,(P \ominus p) \times Q\,\} \cup \{\,P \times (Q \ominus q)\,\}).$$

If $\dim(Q \ominus q) = \dim Q$, then the lemma follows by the Product Lemma. Otherwise, $q \notin \mathrm{aff}(Q \ominus q)$. Let p' be some vertex of P other than p. By Lemma 2.2a, if $(x, y) \in \mathrm{aff}(X \times Y)$, then $x \in \mathrm{aff}\, X$ and $y \in \mathrm{aff}\, Y$. It follows that $(p', q) \notin \mathrm{aff}(P \times (Q \ominus q))$.

But $(p', q) \in P'$, so

$$\dim P' > \dim [P \times (Q \ominus q)] \geq \dim P + \dim Q - 1 \qquad \square$$

We now turn our attention to (4.13).

Lemma 4.7 *Let* P *and* Q *be polytopes containing the origin as a vertex.*

$$\widetilde{\mathcal{F}}_{\mathbb{O}}(P \times Q) = \{ F_p \oplus F_q \mid F_p \in \widetilde{\mathcal{F}}_{\mathbb{O}}(P), F_q \in \widetilde{\mathcal{F}}_{\mathbb{O}}(Q) \}$$

Proof. Let P' denote $(P \times Q) \ominus \mathbb{O}$. Suppose we have facets $F_p \in \widetilde{\mathcal{F}}_{\mathbb{O}}(P)$, $F_q \in \widetilde{\mathcal{F}}_{\mathbb{O}}(Q)$. We can write linear constraints $ax \geq 1$ and $by \geq 1$ that support P' in F_p and F_q respectively. The constraint $ax + by \geq 1$ supports P'. The vertices of $P \times Q$ that lie on $ax + by = 1$ are precisely the vertices of $F_p \oplus F_q$. It is known (see e.g. the proof of Lemma 2.2c) that if $\operatorname{aff} P_1 \cap \operatorname{aff} P_2 = \emptyset$ and P_1 is not a translate of P_2 then $\dim(P_1 \oplus P_2) = \dim P_1 + \dim P_2 + 1$.

Now suppose we have some $F \in \widetilde{\mathcal{F}}_{\mathbb{O}}(P \times Q)$. Let h^0 denote aff F. Every vertex of $P \times Q$ in h^0 must have at least one adjacent edge e in F^-, since otherwise $P \times Q \subset F^+$. The other vertex of e must be $\mathbb{O}$, since otherwise h^0 is not a supporting hyperplane for $(P \times Q) \ominus \mathbb{O}$. By the Product Lemma, the vertices defining h^0 must be of the form $(p, \mathbb{O})$ or $(\mathbb{O}, q)$ for $p \in \mathcal{V}(P)$, $q \in \mathcal{V}(Q)$. It follows that any basis (set of d affinely independent vertices) B defining h^0 must have the form $B = \begin{bmatrix} B_p & \mathbb{O} \\ \mathbb{O} & B_q \end{bmatrix}$ where B_p is a basis of some $F_p \in \widetilde{\mathcal{F}}_{\mathbb{O}}(P)$ and B_q is a basis of some $F_q \in \widetilde{\mathcal{F}}_{\mathbb{O}}(Q)$. $\square$

Since by change of coordinates we can assume without loss of generality that an arbitrary vertex of $P \times Q$ lies on the origin, (4.13) and hence Theorem 4.8 follows. From the proof of Theorem 4.8 we also get (for $\dim P \geq 2$ and $\dim Q \geq 2$) a complete characterization of the facets of $(P \times Q) \ominus v$ since

$$\mathcal{F}((P \times Q) \ominus v) = \{ F \ominus v \mid F \in \mathcal{F}(P \times Q) \} \cup \widetilde{\mathcal{F}}_v(P \times Q) .$$

4.4.2 Consequences

We now present some consequences of Theorem 4.8 for particular families of polytopes. To construct families of polytopes hard for incremental convex hull algorithms, it suffices to take products of families with large loss functions. In this subsection we present three such families. Recall that $C_d(n)$ denotes the cyclic d-polytope with n vertices.

Lemma 4.8 *For even* d, $n \geq d+2$, *for any vertex* v *of* $C_d(n)$,

$$\begin{aligned} \mathrm{loss}(v, C_d(n)) &= \binom{n-d/2-2}{n-d-1} = \left[\frac{d(n-d)}{(2n-d-2)\,n}\right]\gamma(n,d) \\ &\in \Theta(n^{d/2-1}) \qquad\qquad d \textit{ fixed.} \end{aligned}$$

Proof. Let the dimension $d = 2k$ for some positive integer k. Let v be a vertex of $P = C_d(n)$. By Lemma 2.1, each vertex of an even dimensional cyclic polytope is contained in $\gamma(n-1, d-1)$ facets. Since P is simplicial, $\mathrm{gain}(v, P) = \gamma(n-1, d-1)$. Since $P \ominus v$ is full dimensional, $f_{d-1}(P \ominus v) - f_{d-1}(P) = \mathrm{loss}(v, P) - \mathrm{gain}(v, P)$. It follows that

$$\begin{aligned} \mathrm{loss}(v, P) &= f_{d-1}(P \ominus v) - f_{d-1}(P) + \mathrm{gain}(v, P) \\ &= \gamma(n-1, d) - \gamma(n, d) + \gamma(n-1, d-1)\,. \end{aligned}$$

Substituting in the appropriate values of $\gamma(n, d)$ from (2.1) for odd and even d,

$$\mathrm{loss}(v, P) = \left[\frac{n-1}{k} - \frac{(n-k-1)\,n}{(n-2\,k)\,k} + \frac{2\,n-2\,k-2}{n-2\,k}\right]\frac{(n-k-2)\,!}{(n-2\,k-1)\,!\;(k-1)\,!}$$

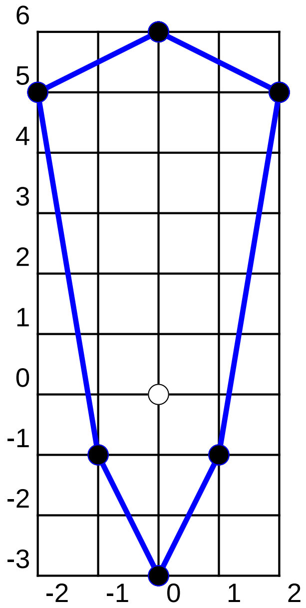
The term in brackets simplifies to 1. □

As usual, we now give two alternative families to products of cyclic polytopes. A polytope P is called *centered* if for every $v \in \mathcal{V}(P)$, $P \ominus v$ contains the origin as an interior point. As in Section 3.1, define π_n as follows. For odd n, choose points:

$$\begin{aligned} x(i) &= (n-1)/2 - i \\ y(i) &= 2x(i)^2 - 3 \qquad\qquad i = 0 \ldots n-1 \end{aligned}$$

If n is even, construct π_{n-1} and then add a vertex with coordinates $(0, n^2/2 - 2n)$. Figure 4.8 illustrates P_6. The reader can verify that P_5 is centered. Since $\mathcal{V}(P_5) \subset \mathcal{V}(P_n)$ for $n > 5$, P_n is centered for $n \geq 5$.

Figure 4.8 π_6



The following lemma gives a general method for constructing families of polytopes with large loss functions.

Lemma 4.9 *Let* P *be a centered polytope. Let* Q *be a polytope with* m *facets containing the origin in its interior. Let* $v = (p, \mathbb{O})$ *be a vertex of* $P \oplus Q$.

$$\text{loss}(v, P \oplus Q) = \text{loss}(p, P) \cdot m\,.$$

Proof. Note that $(P \oplus Q) \ominus (p, \mathbb{O}) = (P \ominus p) \oplus Q$. Since P is centered, by Lemma 2.4,

$ax + by \leq 1$ defines a facet of $(P \ominus p) \oplus Q$ iff $ax \leq 1$ defines a facet of $P \ominus p$ and $by \leq 1$ defines a facet of Q. Vertex $(p, \mathbb{O})$ is infeasible for $ax + by \leq 1$ iff p is infeasible for $ax \leq 1$. $\square$

Again following the development of Section 3.1, let $\Pi_{2k}(n)$ denote $\bigoplus_k \pi_n$.

Corollary 4.1 *For even* d, $n \geq 5$, $\operatorname{loss} \Pi_d(n) = n^{d/2-1}$.

From Lemma 4.8 and Corollary 4.1, we have two families of polytopes with large loss functions. Intuitively, our construction takes the $\sqrt{d}$-fold product of $2\sqrt{d}$-polytopes from these families. In order to have example polytopes in each sufficiently large even dimension, a slightly more complex construction is necessary. Following Section 3.1, for any even dimension $2d \geq 4$ define $a = \lceil \sqrt{d} \rceil$, $b = \lfloor d/a \rfloor$, $c = d \bmod a$, and $K_{2d}(n) = Q_{2a}^b(n) \times Q_{2c}(n)$ where $Q_k(n)$ is either a cyclic polytope $C_k(n)$ or a sum of polygons $\Pi_k(n)$ (and we again omit the term $Q_{2c}(n)$ if c=0).

Theorem 4.9 *For* $d \geq 2$ *held fixed,*

(a) $s \equiv \operatorname{size} K_{2d}(n) \in O(n^{\lceil \sqrt{d} \rceil})$, *and*

(b) $\operatorname{drop} K_{2d}(n) \in \Omega(s^{d/\lceil \sqrt{d} \rceil - 1})$.

Proof. Let $b' = \lceil d/a \rceil = b + \operatorname{sgn}(c)$. We will make use of the fact that $b' \leq a$. By (3.2) and (3.3), $\operatorname{size} K_{2d}(n) \in \Theta(n^a)$. By Lemma 4.8 and Corollary 4.1, $\operatorname{loss} Q_{2k}(n) \in \Theta(n^{k-1})$. By Theorem 4.8, $\operatorname{drop} K_{2d}(n) \in \Omega(n^{\phi(d)})$ where

$$\begin{aligned}
\phi(d) &= (a-1)b + \operatorname{sgn}(c) \cdot (c-1) \\
&= (ab + \operatorname{sgn}(c) \cdot (c-1) + \operatorname{sgn}(c)) - (b + \operatorname{sgn}(c))) \\
&= d - b' \\
&= a(d/a - b'/a) \\
&\geq a(d/a - 1)
\end{aligned}$$

$\square$

From this theorem we can see that incremental convex hull algorithms are strongly superpolynomial in the worst case, irrespective of the insertion order used. Since these same families were shown in Chapter 3 to be hard for perturbation (triangulation) and for face lattice producing algorithms, it follows that no method consisting of running several of the well known methods in parallel will be polynomial either.

An important class of polytopes for facet enumeration is the 0/1-*polytopes*, whose vertices are a subset of $\{0, 1\}^d$. Because of their importance in combinatorial optimization, a facet enumeration algorithm polynomial for 0/1-polytopes would be significant result; unfortunately incremental algorithms fail here also. We consider the equivalent case of polytopes whose vertices are a subset of $\{+1, -1\}^d$. Let H_d denote the hypercube with vertices $\{+1, -1\}^d$. The reader can verify that H_d is centered for $d \geq 3$. Recall that U_{3k} denotes $\bigodot_k H_3$. Taking the definition of a, b, and c from Theorem 4.9, let $\dddot{U}_{3d}$ denote $U_{3a}^b \times U_{3c}$.

Theorem 4.10 *For* d *sufficiently large,*

$$\operatorname{drop} \dddot{U}_{3d} \geq s^{d/(\lceil\sqrt{d}\rceil \log\lceil\sqrt{d}\rceil)} \qquad \textit{where} \qquad s \equiv \operatorname{size} \dddot{U}_{3d}.$$

Proof. By (3.4), for d sufficiently large, $\operatorname{size} \dddot{U}_{3d} \leq 3^{a \log a}$. By Lemma 2.5, Lemma 4.9 also holds with the $\oplus$ operation replaced by $\odot$. Since $\operatorname{loss} H_3 = 1$, it follows that $\operatorname{loss} U_{3k} = 6^{k-1}$. Let $b' = \lceil d/a \rceil$. By Theorem 4.8,

$$\begin{aligned}
\operatorname{drop} \dddot{U}_{3d} &\geq 6^{(a-1)b+\operatorname{sgn}(c)\cdot(c-1)} \\
&= 6^{d-b'} && \text{see proof of Theorem 4.9} \\
&= \frac{2^d 3^d}{2^{b' \log 6}} \\
&\geq 3^d && \text{if } b' \log 6 < d \\
&\geq s^{d/(a \log a)} && d \text{ suff. large} \qquad \square
\end{aligned}$$

4.4.3 Experimental Results

This subsection presents experimental results for incremental facet enumeration programs on our three “universal” families $\dddot{C}_{2d}(n)$, $\dddot{\Pi}_{2d}(n)$, and $\dddot{U}_{3d}$.

Figure 4.9 Maximum intermediate facets versus size(P) for $\dddot{C}_8(n)$, **cddr+**

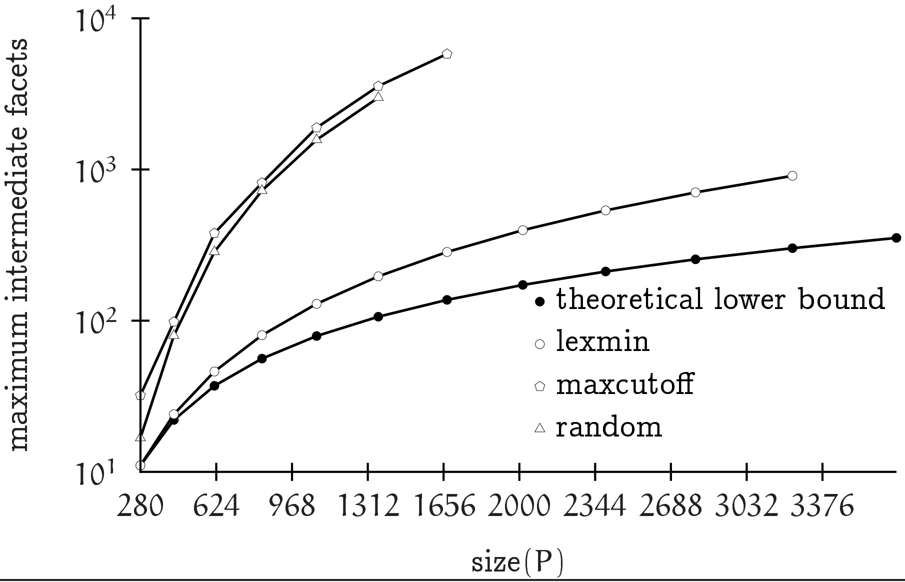
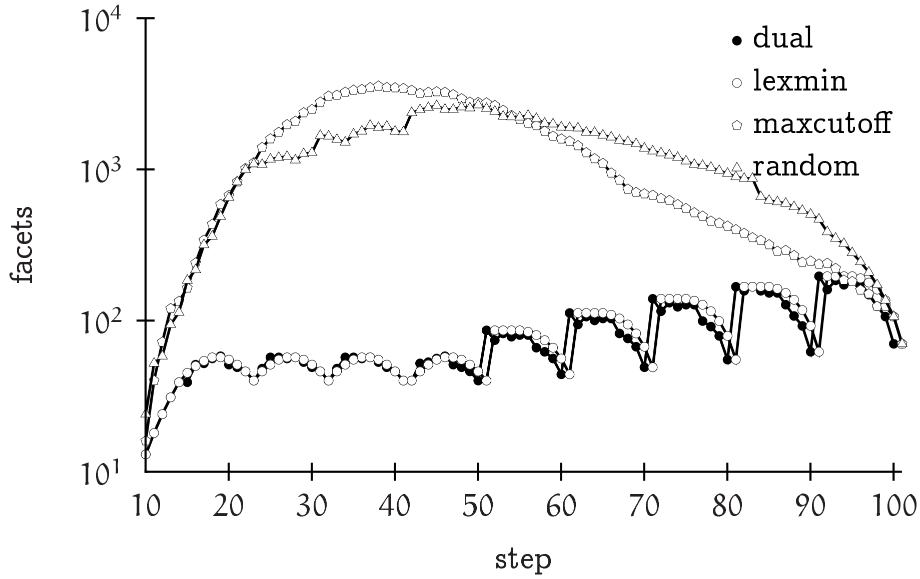


Figure 4.9 plots the maximum number of intermediate facets versus size(P) for the products of cyclic polytopes $\dddot{C}_8(n) = C_4^2(n)$. These measurements are all for **cddr+**; measurements for **porta** are identical to those for **cddr+** with the lexmin insertion order. The numbers for random ordering are the average of 100 trials. Although Theorem 4.9 provides a lower bound for all insertion orders, in practice there is a wide variation between the best insertion orders and the worst ones. Figure 4.10 shows the intermediate size (in facets) at each step for several insertion orders. The trace for random ordering is from a single trial. The order labeled “dual” is the order the vertices are produced by **cddr+** using lexmin ordering for the dual vertex enumeration problem. Both this order and lexmin somehow reflect the product structure of these polytopes since all vertices of the form (v_i, x) for a particular v_i are inserted sequentially (or at least close together). In [4], the authors observe that for random ordering, the lower bound of Theorem 4.9 can be improved

by a factor of n (for d fixed).

Figure 4.10 Trace of intermediate size for $\dddot{C}_8(10)$, **cddr+**.



A natural question for examples such products of cyclic polytopes where one transformation is difficult, is whether there is a good insertion order for the dual transformation. Table 4.1 shows the results of various halfspace insertion orders for $\dddot{C}_8(n)$. There is again a fairly wide variation among insertion orders. The best order is minindex, which again reflects the product structure of the polytopes, since the halfspaces from each factor polytope (i.e. $C_4(n)$) are grouped together in the input files. The results for this order are very close to the best of the orders for facet enumeration, included as the last column of Table 4.1. The difference between the best and worst insertion orders is quite significant in practical terms: while the best order takes several minutes for $\dddot{C}_8(10)$ (which has 100 vertices and 70 facets), the worst order takes over four hours (both on a Digital AlphaServer 4/233).

The products of sums of polygons $\dddot{\Pi}_d(n)$ have the same asymptotic lower bound as products of cyclic polytopes (see Theorem 4.9). Their better numerical behaviour allows the use of floating point, which fails quite quickly for $\dddot{C}_d(n)$, even in 8 dimensions. In Table 4.2, rather than fixing the dimension, we fix the number of vertices as a function of the dimension. Once again there is large variation among

Table 4.1 Maximum intermediate facets, $\dddot{C}_8(n)$, vertex enumeration.

n	**cddr+** lexmin	**cddr+** maxcutoff	**cddr+** minindex	**cddr+** random	porta	**cddr+** lex., facets
5	25	25	25	25	25	11
6	42	46	42	50.210	42	24
7	70	74	70	111.790	70	46
8	112	182	112	245.540	112	80
9	202	260	171	472.360	171	129
10	364	442	250	819.270	250	196

the insertion orders; as the dimension increases the best of the insertion orders is much farther from the theoretical lower bound.

Table 4.2 Maximum intermediate size, $\dddot{\Pi}_{2d}(5)$, facet enumeration

d	$f_0(P)$	$c_0(P)$	lower bound	**cddf+** lexmin	**cddf+** maxcutoff	**cddf+** minindex	**porta**
1	5	5	6	5	5	5	5
2	10	25	30	25	25	25	25
3	50	30	35	45	166	45	45
4	100	50	75	125	1658	125	125
5	150	150	275	525	6367	525	525
6	225	250	875	2125	35379	2125	2125
7	1125	255	880	7755		7755	7755

As before we consider dual problem of vertex enumeration for $\dddot{\Pi}_{2d}(5)$. Table 4.3 suggests that the best insertion orders for vertex enumeration are better than those for facet enumeration.

Table 4.4 shows maximum intermediate size for the products of sums of 3-cubes. If we compare with the experimental results of Section 3.4.1, we can see that for all three "universal" families, even though the asymptotic lower bounds given for pivoting algorithms and incremental algorithms are about the same, the experimental performance of the incremental algorithms seems much closer to the lower bounds,

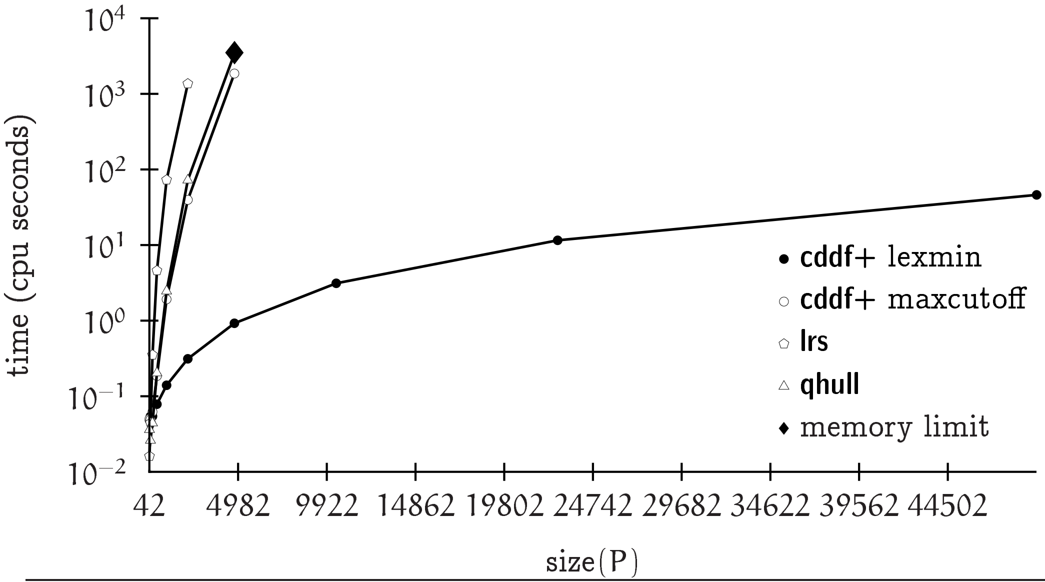
Table 4.3 Maximum intermediate size, $\ddot{\Pi}_{2d}(5)$, vertex enumeration.

d	$f_0(P)$	**cddf+** lexmin	**cddf+** maxcutoff	**porta**
1	5	5	5	5
2	10	14	23	13
3	50	65	71	50
4	100	130	195	130
5	150	260	2836	195
6	225	390	4246	360
7	1125	1950	4248	1125
8	2250	3900	4506	2925
9	3375	5850		5400

Table 4.4 Maximum intermediate size, $\ddot{U}_{3d}$, facet enumeration

d	$f_0(P)$	$c_0(P)$	lower bound	**cddf+** lexmin	**cddf+** maxcutoff	**cddf+** minindex	**porta**
1	8	6	7	7	7	7	7
2	16	36	42	42	42	42	42
3	128	42	48	60	918	60	60
4	256	72	108	180	33671	180	180
5	384	252	468	900		900	900
6	576	432	1728	4320		4320	4320
7	4608	438	1734	6918			8214

Figure 4.11 Running time in cpu seconds, H_d



and hence much larger problems are solvable by incremental methods within a given amount of time. In all three cases, the examples solvable in a reasonable amount of time on current hardware are really too small for the asymptotic bounds to be very meaningful. For example, for $d \leq 10$, $\mathrm{drop}(\dddot{U}_{3d}) < \mathrm{size}(\dddot{U}_{3d})$.

The three families from this section are in some sense universally difficult for facet enumeration algorithms, since in Chapter 3 we saw that they are difficult also for pivoting algorithms. By way of contrast, we consider two other families polytopes shown to difficult for pivoting algorithms, namely cubes and products of simplices. In each case, it turns out that at least empirically, there are good insertion orders for these families. Figure 4.11 and Figure 4.12 compare the running time of **cddf+** using two different insertion orders with **qhull** and the pivoting based **lrs**. It is interesting that the two greedy insertion orders (maxcutoff and qhull) perform quite badly on these examples. In the case of **qhull** this can be partially blamed on triangulation, but as Table 4.5 shows, the untriangulated intermediate size grows quite quickly as well. In [4] the authors show that a family closely related to $\dddot{T}_{2d}$ is difficult for maxcutoff insertion order. Lexicographic order seems to work very well, even for the 30 dimensional product of simplices example, whose facets would

Figure 4.12 Running time in cpu seconds, $\dddot{T}_{2d}$

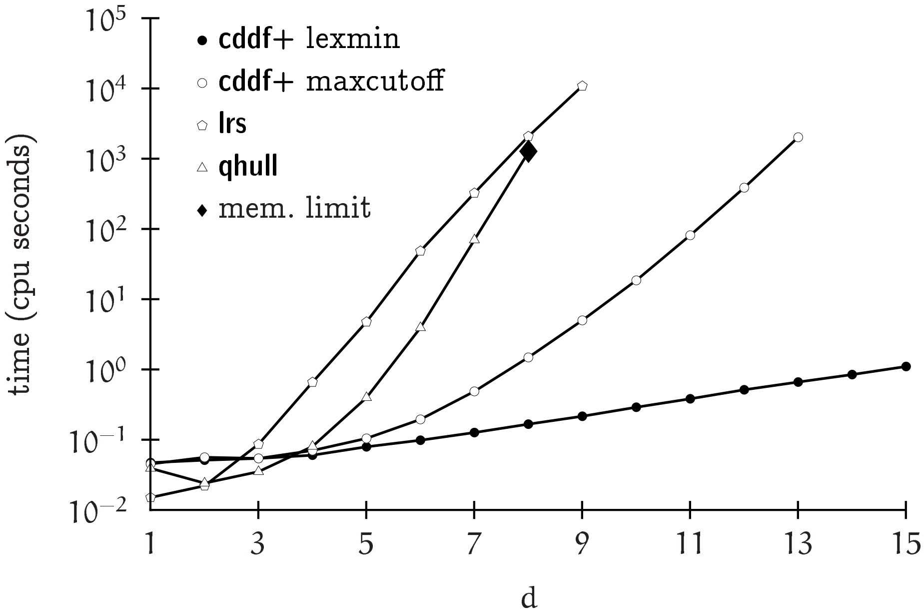


Table 4.5 Summed intermediate size, $\dddot{T}_{2d}$.

d	**cddf+** lexmin	**cddf+** maxcutoff	**qhull**
1	6	6	6
2	14	15	12
3	23	35	25
4	34	79	62
5	47	175	171
6	62	383	609
7	79	831	2240
8	98	1791	8774
9	119	3839	*overflow*
10	142	8191	

require more than 500 million $(d - 1)$-simplices to triangulate.

Bibliographic Notes

The first published lower bounds for incremental algorithms seem to be those of Dyer [19]. In this paper Dyer uses a construction of Klee [99] which build a polytope with many facets, all incident to one "universal" facet[2]. Dyer shows that one can find a halfspace that creates a prism out of the universal facet. This yields a drop in intermediate size almost as extreme as Theorem 4.4.

The description of the double description method in [34] is in terms of cones. Although as we discuss in Section 4.1, this is the cleanest way to deal with unboundedness, it seems to have the unfortunate effect of making the algorithm less accessible to the computational geometry community. For a description and experimental comparison of several variations of the double description method, see [25].

The results of Section 4.2 were first presented in slightly different form in [2] and [3]. The dwarfed polytope examples of Section 4.3 first appeared in [4], which also included several arguments showing that particular families of dwarfed polytopes (rather than dwarfed polytopes in general) are bad for greedy insertion orders that choose the next halfspace that cuts off the maximum or minimum number of intermediate vertices. The results of Section 4.4 appeared in a preliminary form in [7]. Raimund Seidel suggested the use of products of sums of polygons for Theorem 4.9. The use of products of sums of cubes for Theorem 4.10 was suggested to the author by David Avis.

[2] such polytopes are called Kirkman polytopes, after the Reverend Thomas P. Kirkman, who characterized them in 3-dimensions [98]

Chapter 5

Primal–Dual Algorithms

5.1 Introduction

It follows from the duality of convex polytopes that the vertex and facet enumeration problems are polynomially equivalent, that is, the existence of a polynomial algorithm for one problem implies the same for the other problem. Several polynomial algorithms (see e.g. [5, 11, 85, 19, 37, 39]) are known under strong assumptions of nondegeneracy, which restrict input polytopes to be simple in the case of vertex enumeration and simplicial in the case of facet enumeration. On the other hand, we have seen in Chapter 3 and Chapter 4 that none of the known methods are polynomial for general polytopes.

In this chapter, we extend the known polynomially solvable classes by looking at the dual problems. The *dual* problem of a vertex (facet, respectively) enumeration problem is the facet (vertex) enumeration problem for the same polytope where the input and output are simply interchanged. For a particular class of polytopes and a fixed algorithm, one transformation may be much easier than its dual. One might be tempted to explain this possible asymmetry by observing that the standard nondegeneracy assumption is not self-dual. Are the dual problems of nondegenerate vertex (facet) enumeration problems harder? More generally, are the complexities of the primal and the dual problem distinct?

Recall that an algorithm is said to be *polynomial* if the time to solve any instance is bounded above by a polynomial in the size of input and output. A *successively polynomial algorithm* is one whose kth output is generated in time polynomial in k and the input size s, for each k less than or equal to the cardinality of output. Clearly every successively polynomial algorithm is a polynomial algorithm.

In Section 5.2 we will show in a certain sense that the primal and dual problems are of the same complexity. More precisely, we will show the following theorem: if there is a successively polynomial algorithm for the vertex (facet, respectively) enumeration problem for a hereditary class of problems, then there is a successively polynomial algorithm for the facet (vertex) enumeration problem for the same class, where a hereditary class contains all subproblems of any instance in the class. We propose a new class of algorithms that take advantage of this phenomenon. Loosely speaking, *primal–dual* algorithms use a solution to the easy direction as an oracle to help solve the seemingly hard direction.

The non-degenerate polytopes are in some sense the least complicated of all polytopes. The simplicial polytopes have the fewest possible vertices on each facet; the simple polytopes have the fewest possible number of facets containing each vertex. Which of these properties is most desirable depends on the problem at hand. If we are given a halfspace description and want to enumerate its vertices, then we know how to efficiently solve the problem if the halfspaces define a simple polytope. Similarly, if the problem is to enumerate the facets given the vertices, then we know how to solve the problem if the polytope is simplicial. In both of these cases we say the problem input is *primal-nondegenerate*. If, on the other hand, the problem is vertex (respectively facet) enumeration and the input is simplicial (respectively simple) enumeration then we say that the problem is *dual-nondegenerate*. In the language of Chapter 3 we can measure the primal (respectively dual) degeneracy of a polytope in a quantitative sense by its primal (respectively dual) triangulation complexity.

From the general result of Section 5.2 relating the complexity of the primal and dual problems, and known polynomial algorithms for the primal nondegenerate case, we arrive at a polynomial algorithm for vertex enumeration for simplicial polytopes and facet enumeration for simple polytopes. In Section 5.3 we show how to refine this algorithm to yield an algorithm with time complexity competitive with the algorithms known for the primal-nondegenerate case. Section 5.4 considers how to modify the algorithm of Section 5.3 to perform well when the polytope is (moderately) dual-degenerate. Section 5.5 contains some experimental results and Section 5.6 some concluding remarks for the chapter.

The only published investigation of the dual-nondegenerate case the author is aware of is in a paper by Peter Gritzmann and Victor Klee [27]. Their approach, most easily understood in terms of vertex enumeration, consists of intersecting the constraints with each defining hyperplane and, after removing the redundant constraints, finding the vertices lying on that facet by some brute force method. David Avis (private communication) has independently observed that this method can be extended to any polytope whose facets are simple (or nearly simple) polytopes. The method of Gritzmann and Klee requires solving $O(m^2)$ linear programs (where m is the number of input halfspaces) to remove redundant constraints. Our approach does not rely on the polynomial solvability of linear programming if an interior point is known (as is always the case for facet enumeration).

5.2 Primal–Dual Algorithms

In this section we consider the relationship between the complexity of the primal problem and the complexity of the dual problem for vertex/facet enumeration. We will fix the primal problem as facet enumeration in the rest of this chapter, but the results can also be interpreted in terms of vertex enumeration. For convenience we assume in this chapter that the input polytope is full dimensional and contains the origin as an interior point. While it is easy to see this is no loss of generality in the case of facet enumeration, in the case of vertex enumeration one might need to solve a linear program to find an interior point.

We call a family Γ of polytopes *facet-hereditary* if for any $P \in \Gamma$, for any $H' \subset \mathcal{H}(P)$, if $\bigcap H'$ is bounded then $\bigcap H'$ is also in Γ. The main idea of this chapter is summarized by the following theorem.

Theorem 5.1 *If there is a successively polynomial vertex enumeration algorithm for a facet-hereditary family of polytopes then there is a successively polynomial facet enumeration algorithm for the same family.*

Simple polytopes are not necessarily facet-hereditary, but each simple polytope can be perturbed symbolically or lexicographically onto a combinatorially equivalent polytope whose facet defining halfspaces are in "general position", i.e. the arrangement of facet inducing hyperplanes defined by the polytope is simple. The family of polytopes whose facet inducing halfspaces are in general position is obviously facet-hereditary.

Corollary 5.1 *There is a successively polynomial algorithm for facet enumeration of simple polytopes and for vertex enumeration of simplicial polytopes.*

Proof of Theorem 5.1 is constructive, via the following simple algorithm. Algorithm 1 takes as input a set V of points in $\mathbb{R}^d$, and a subset $H_0 \subset \mathcal{H}(V)$ such that $\bigcap H_0$ is bounded. We show below how to compute such a set of halfspaces.

Algorithm 1 PrimalDualFacets(V, H_0)

$H_{cur} \leftarrow H_0$
while $\exists \tilde{v} \in \mathcal{V}(H_{cur}) \setminus V$ do `FindWitness`
  Find $h \in \mathcal{H}(V)$ s.t. $\tilde{v} \in h^-$ `DeleteVertex`
  $H_{cur} \leftarrow H_{cur} \cup \{h\}$
endwhile
return H_{cur}.

At every step of the algorithm we maintain the invariant that $\operatorname{conv} V \subseteq \mathcal{P}(H_{cur})$. When the algorithm terminates, $\mathcal{V}(H_{cur}) \subseteq V$, hence $\mathcal{P}(H_{cur}) \subseteq \operatorname{conv} V$. There are two main steps in this algorithm that we have labeled FindWitness and Delete-Vertex. The vertex $\tilde{v} \in \mathcal{V}(H_{\mathrm{cur}}) \setminus V$ is a *witness* in the sense that for any such vertex, there must be a facet of $\mathcal{H}(V)$ not yet discovered whose defining halfspace cuts off $\tilde{v}$. From the precondition of the theorem there exists a successively polynomial algorithm to enumerate the vertices of H_{cur}. It follows that in time polynomial in $|V|$ we can find $|V|+1$ vertices of $\mathcal{P}(H_{\mathrm{cur}})$, or discover $\mathcal{V}(H_{\mathrm{cur}}) = V$. If we discover $|V|+1$ vertices, one of these vertices must be a witness. In order to find the facet cutting off a witness (the "DeleteVertex" step), we need to solve a separating hyperplane problem for a point and convex set. The separating hyperplane problem can be solved via the following linear program: maximize $\tilde{v}y$ subject to $Vy \leq \mathbb{1}$. If y^* is a basic optimal solution (i.e. a solution corresponding to a vertex of the polar polytope $P^* = \{y \mid Vy \leq \mathbb{1}\}$) of the linear program then $y^*x \leq 1$ is the desired separating halfspace. While there are linear programming algorithms polynomial in the bit size of the input, there are not yet any known that are polynomial in $n = |V|$ and d, which is what we need for our theorem. It turns out that because we have a halfspace description of the convex hull of the union of our two sets, we can solve the separating hyperplane problem via a much simpler algorithm. The rest of this section is organized as follows. In Section 5.2.1 we discuss how to implement the DeleteVertex step without solving a linear program. In Section 5.2.2 we discuss how to preprocess to eliminate the various boundedness and full dimensionality assumptions made above. Taken together, the results of these two sections will establish the following stronger version of Theorem 5.1:

Theorem 5.2 *For any facet hereditary family of polytopes* Γ *if we can generate* k *vertices of an* m*-facet* d*-polytope* $P' \in \Gamma$ *(or certify that* P' *has less than* k *vertices) in time* $O(f(k, m, d))$ *then we can enumerate the* m *facets of an* n*-vertex* d*-polytope* P *in time*

$$O(nd^3 + mnd^2 + m^2d + \sum_{i=d+2}^{m} f(n+1, i-1, d)) .$$

In certain cases (such as the dual-nondegenerate case considered in Section 5.3), we may have a theoretical bound for $f(k, m, d)$ polynomial in k, m, and d. In other cases, such a theoretical bound may be difficult to obtain, but we may have experimental evidence that a certain method (e.g. some heuristic insertion order for an incremental algorithm) is efficient for vertex enumeration for Γ. In either case the techniques described in this section can be used to obtain an efficient method for facet enumeration as well. It is worth noting that there is no restriction of the input points to be in "convex position". Redundant (interior) input points will have no effect other than to slow down pivot operations and tests for membership in the input (i.e. m will be the total number of input points, including redundant points).

5.2.1 Deleting Vertices without Linear Programming

Our main tool here will be the pivot operation of the simplex method of linear programming. Any inequality system

$$Hx \leq \mathbb{1} \tag{5.1}$$

can be represented in the standard "dictionary" form (see e.g. [85]) as follows. We transform each inequality into an equality by adding a slack variable, to arrive at the following system of linear equations or *dictionary*:

$$s = \mathbb{1} - Hx \tag{5.2}$$

More precisely, a dictionary for (5.1) is a system obtained by solving (5.2) for some subset of m slack and original variables (where m is the row size of H). A solution to (5.2) is feasible for (5.1) if and only if $s \geq \mathbb{0}$. In particular since $H\mathbb{0} < \mathbb{1}$, $s = \mathbb{1}$ is a feasible solution to both. The variables are naturally partitioned into two sets. The variables appearing on the left hand side of a dictionary are called *basic*; those on the right hand side are called *cobasic*.

A *pivot* operation moves between dictionaries by making one cobasic variable (the *entering variable*) basic and one basic variable (the *leaving variable*) cobasic. If we have a feasible point for a polytope and a halfspace description, in d pivot operations we can find a vertex of the polytope. If we ensure that each pivot does not decrease a given objective function, then we have the following.

Lemma 5.1 (Raindrop Algorithm) *Given* $H \in \mathbb{R}^{m\times d}$, $\omega \in \mathbb{R}^d$ *and* $v_0 \in \mathcal{P}(H)$, *in time* $O(md^2)$ *we can find* $v \in \mathcal{V}(H)$ *such that* $\omega v \geq \omega v_0$.

Proof. We start by translating our system by $-v_0$ so that our initial point is the origin. As a final row to our dictionary we add the the equation $z = \omega x$ (the *objective row*). Note that by construction, $x = \mathbb{0}$ is a feasible solution. We start a pivot operation by choosing some cobasic variable x_j to become basic. Depending on the sign of the coefficient of x_j in the objective row, we can always increase or decrease x_j without decreasing the value of z. As we change the value of x_j, some of the basic slack variables will decrease as we get closer to the corresponding hyperplane. By considering ratios of coefficients, we can find one of the first hyperplanes reached. By moving that slack variable to the right hand side (making it cobasic), and moving x_j to the left hand side, we obtain a new dictionary in $O(md)$ time (see e.g. [85] for details of the simplex method). We can continue this process as long as there is a cobasic x-variable. After exactly d pivots, all x-variables are basic. It follows that the corresponding basic feasible solution is a vertex (See Figure 5.1). □

The raindrop algorithm seems to be part of the folklore of Linear Programming; a generalized version is discussed in [114].

By duality of convex polytopes we have the following.

Lemma 5.2 (Dual Raindrop Algorithm) *Given* $V \in \mathbb{R}^{n\times d}$, $\omega \in \mathbb{R}^d$ *and* h_0 *such that* $V \subset h_0^+$, *in* $O(nd^2)$ *time we can find* $h \in \mathcal{H}(V)$ *such that* $h\omega \geq h_0\omega$.

Essentially this is the same as the initialization step of a gift-wrapping algorithm (see e.g. [11, 39]), except that we are careful that the point ω is on the same side of our final hyperplane as of the one we started with. Figure 5.2 illustrates the rotation dual to the pivot operation in Lemma 5.1.

Figure 5.1 The raindrop algorithm

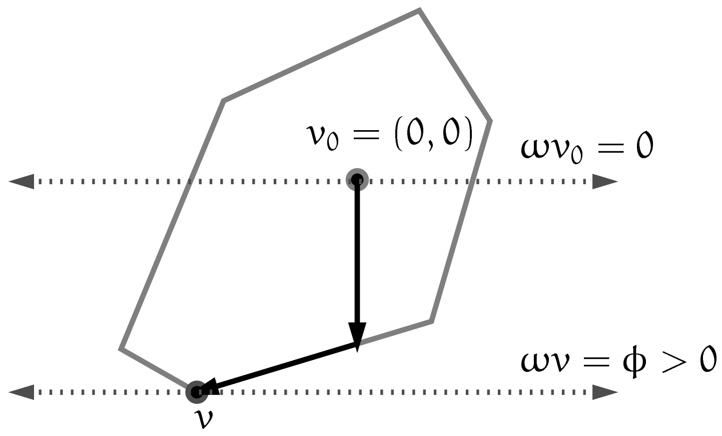
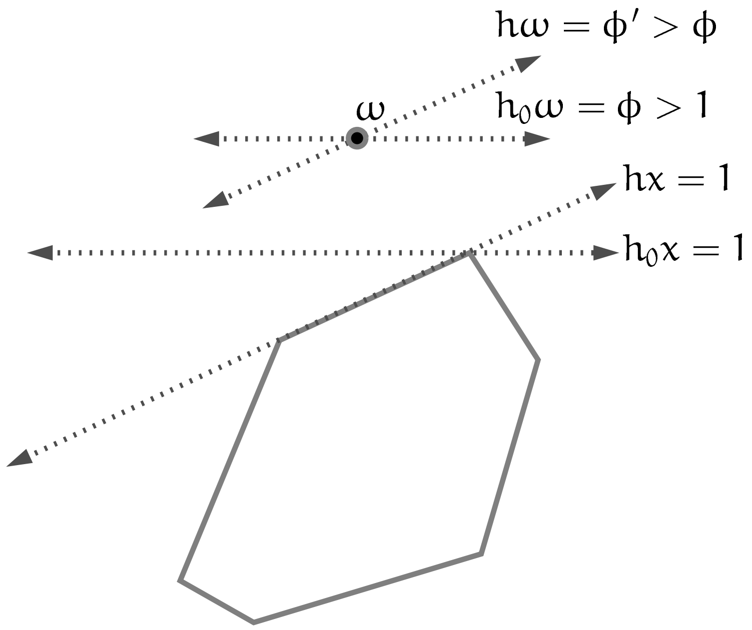
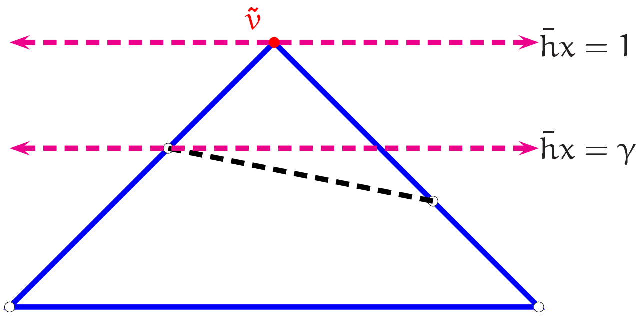


Figure 5.2 Pivoting from a valid inequality to a facet.



We can now show how to implement the DeleteVertex step of Algorithm 1 without linear programming. A *basis* B for a vertex $v \in \mathcal{V}(H)$ is a set of d rows of H such that $Bv = \mathbb{1}$ and $\operatorname{rank} B = d$. We can obviously find a basis in polynomial time; in the pivoting based algorithms in the following sections we will always be given a basis for v.

Figure 5.3 Illustrating the proof of Lemma 5.3.



Lemma 5.3 (DeleteVertex) *Given* $V \in \mathbb{R}^{n \times d}$, $H_0 \subset \mathcal{H}(V)$, $\tilde{v} \in \mathcal{V}(H_0) \setminus V$, *and a basis* B *for* $\tilde{v}$, *we can find* $h \in \mathcal{H}(V)$ *such that* $\tilde{v} \in h^-$ *in time* $O(nd^2)$.

Proof. Let $\bar{h} = \frac{1}{d} \sum_{b \in B} b$. The inequality $\bar{h}x \le 1$ is satisfied with equality by $\tilde{v}$ and with strict inequality by every $v \in V$ (since $\tilde{v}$ is the unique vertex of $\mathcal{P}(H_0)$ lying on $\bar{h}^0$; see Figure 5.3). Let $\gamma = \max_{v \in V} \bar{h}v$. Since $\mathbb{O} \in \operatorname{int} \operatorname{conv} V$, $\gamma > 0$. Let $h_0 = \bar{h}/\gamma$. The constraint $h_0 x \le 1$ is valid for $\operatorname{conv} V$, but $h_0 \tilde{v} > 1$. The lemma then follows from Lemma 5.2. □

If we are not given a basis for the vertex $\tilde{v}$ we wish to cut off, we can use the mean of the outward normals of all facets meeting at $\tilde{v}$ in place of the vector $\bar{h}$. This mean vector can be computed in time $O(|H_0|d)$ time.

Corollary 5.2 *Given* $V \in \mathbb{R}^{n \times d}$, $H_0 \subset \mathcal{H}(V)$, *and* $\tilde{v} \in \mathcal{V}(H_0) \setminus V$, *we can find* $h \in \mathcal{H}(V)$ *such that* $\tilde{v} \in h^-$ *in time* $O(nd^2 + |H_0|d)$.

It will prove useful below to be able to find a facet of $\operatorname{conv} V$ that cuts off a particular extreme ray or direction of unboundedness for our current intermediate polyhedron.

Lemma 5.4 (DeleteRay) *Given* $V \in \mathbb{R}^{n\times d}$ *and* $r \in \mathbb{R}^d \setminus \{\mathbb{O}\}$*, in* $O(nd^2)$ *time we can find* $h \in \mathcal{H}(V)$ *such that* $hr > 0$*.*

Proof. The proof is similar to that of Lemma 5.3. Let $\gamma = \max_{v\in V} rv$. Since $\mathbb{O} \in \operatorname{int}\operatorname{conv} V$, $\gamma > 0$. Let $h_0 = r/\gamma$. The constraint $h_0 x \leq 1$ is valid for $\operatorname{conv} V$, but $h_0 r = (r\cdot r)/\gamma > 0$. By Lemma 5.2, in $O(nd^2)$ time we can compute $h \in \mathcal{H}(V)$ such that $hr \geq h_0 r > 0$. □

5.2.2 Preprocessing

We have assumed throughout that the input polytopes are full dimensional and contain the origin as an interior point. This is polynomially equivalent to assuming that along with a halfspace or vertex description of P, we are given a relative interior point point, i.e. an interior point of P in $\operatorname{aff} P$. Given a relative interior point, then (either representation of) P can be shifted to contain the origin as an interior point and embedded in a space of dimension $\dim P$ in $O(Nd^2)$ time by elementary matrix operations, where N is the number of input halfspaces or points.

Finding a relative interior point in a set of points requires only the computation of the centroid. On the other hand, finding a relative interior point of the intersection of a set of halfspaces H requires solving at least one (and no more than $|H|$) linear programs. Since we are interested here in algorithms polynomial in n, m, and d, and there not yet any such linear programming algorithms, we thus assume that the relative interior point is given.

In order to initialize Algorithm 1, we need to find some subset $H_0 \subset \mathcal{H}(V)$ whose intersection is bounded. We start by showing how to find a subset whose intersection is pointed, i.e. has at least one vertex.

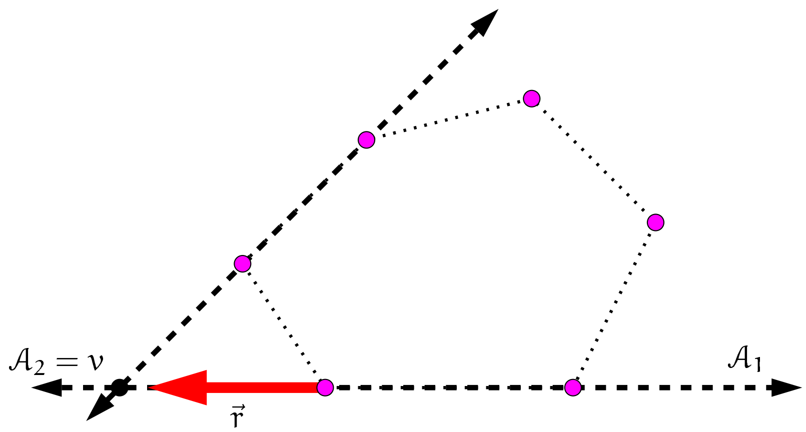
Algorithm 2 FindPointedCone

$H \leftarrow \emptyset$. $\vec{r} \leftarrow x \in \mathbb{R}^d \setminus \{\mathbb{O}\}$. $\mathcal{A} \leftarrow \mathbb{R}^d$.
while $|H| < d$ do
    $h \leftarrow \text{DeleteRay}(\vec{r}, V)$
    $H \leftarrow H \cup \{h\}$
    $\mathcal{A} \leftarrow \mathcal{A} \cap h^0$
    Let a and b distinct points in $\mathcal{A}$.
    $\vec{r} \leftarrow a - b$.
endwhile
return H

Lemma 5.5 *Given* $V \in \mathbb{R}^{n \times d}$, *in* $O(nd^3)$ *time, Algorithm 2 computes subset* $H \subset \mathcal{H}(V)$ *such that* $\bigcap H$ *defines a vertex.*

Proof. We can compute a parametric representation of the affine subspace $\mathcal{A}$ defined by the intersection of all hyperplanes found so far in $O(d^3)$ time by Gaussian Elimination. With each DeleteRay call in Algorithm 2, we find a hyperplane that cuts off some ray in the previous affine subspace (see Figure 5.4). It follows that the dimension of $\mathcal{A}$ decreases with every iteration. □

Figure 5.4 Successive affine subspaces $\mathcal{A}_i$ computed by Algorithm 2



We now show how to augment the set of halfspaces computed by Algorithm 2 so that the intersection of our new set is bounded. To do so, we use a constructive proof of Carathéodory's Theorem. The version we use here is based on that presented by Edmonds [21]. Similar ideas occur in an earlier paper by Klee [113].

Lemma 5.6 (The Carathéodory Oracle) *Given* $H \in \mathbb{R}^{m \times d}$ *such that* $\mathcal{P}(H)$ *is a bounded* d*-polytope and* $v_0 \in \mathcal{P}(H)$*, in time* $O(md^3)$ *we can find* $V \subset \mathcal{V}(H)$ *such that* $v_0 \in \operatorname{conv} V$ *and* $|V| \le d + 1$.

Proof. (sketch) Let $P = \mathcal{P}(H)$. Apply Lemma 5.1 to find $v \in \mathcal{V}(H)$. If $v = v_0$, return v. Otherwise, find the point z at which the ray $\overrightarrow{vv_0}$ exits P. Intersect all constraints with the minimal face containing z and recurse with z as the given point in the face. The recursively computed set, along with v, will contain v_0 in its convex hull. □

By duality of convex polytopes, we have the following:

Lemma 5.7 (The Dual Carathéodory Oracle) *Given a* d*-polytope* $P = \operatorname{conv} V$ *and* h_0 *such that* $V \subset h_0^+$*, we can find in time* $O(|V|d^3)$*, some* $H \subset \mathcal{H}(V)$ *such that* $h_0 \in \operatorname{conv} H$ *and* $|H| \le d + 1$.

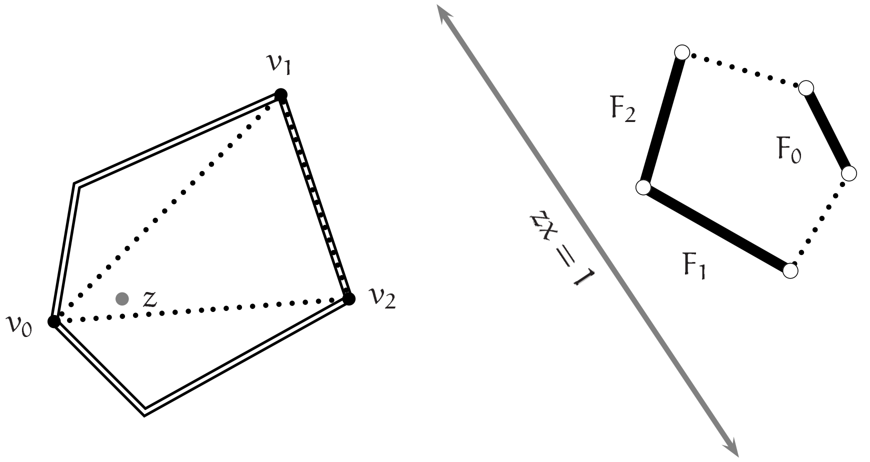
Figure 5.5a illustrates the application of the Carathéodory oracle to find a subset of vertices of a polygon containing an interior point v_0 in their convex hull. In Figure 5.5b the equivalent dual interpretation of finding a set of facets that imply a valid inequality is shown. In order to understand the application of Lemma 5.7, we note the following:

Proposition 5.1 *Let* $P = \{\, x \mid Ax \le \mathbb{1} \,\}$ *and* $Q = \{\, x \mid A'x \le \mathbb{1} \,\}$ *be polyhedra such that each row* a' *of* A' *is a convex combination of rows of* A. $P \subseteq Q$.

Using Lemma 5.5 and Lemma 5.7, we can now find a subset of $\mathcal{H}(V)$ whose intersection is bounded.

Lemma 5.8 *Given* $V \in \mathbb{R}^{n \times d}$*, in time* $O(nd^3)$ *we can compute a subset* $H \subseteq \mathcal{H}(V)$ *such that* $\bigcap H$ *is bounded and* $|H| \le 2d$.

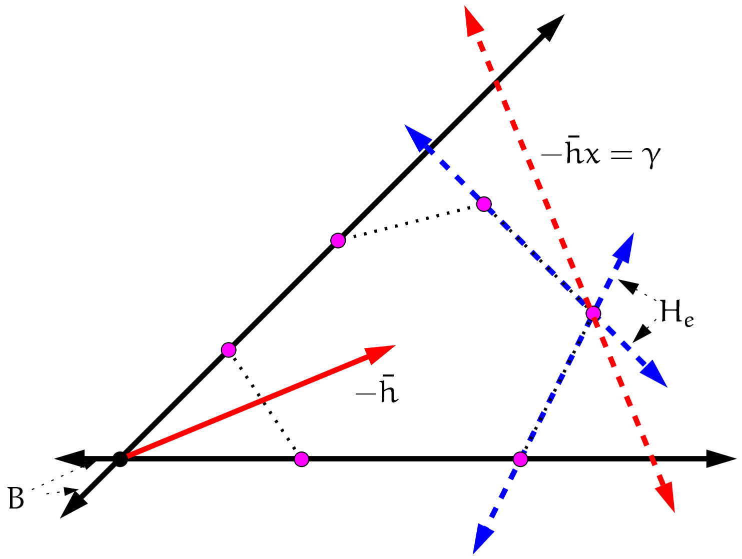
Figure 5.5 The primal and dual Carathéodory oracles.



(a) Using the Carathéodory Oracle to find a set of points whose convex hull contains v_0.

(b) The dual problem of finding a set of facets that imply a given valid constraint $h_0 x \leq 1$.

Figure 5.6 Illustrating the proof of Lemma 5.8



Proof. We start by computing set B of d facet defining halfspaces whose intersection defines a vertex, using Algorithm 2. The proof is then similar to that of Lemma 5.3 and Lemma 5.4. Compute the mean vector $\bar{h}$ of the normal vectors in B (see Figure 5.6). Let $\gamma = \max_{v \in V} -\bar{h}v$. Let $h_0 = -\bar{h}/\gamma$. Note that h_0^+ is valid for V, but any ray feasible for $\bigcap B$ will be cut off by this constraint; hence $\mathcal{P}(B) \cap h_0^+$ is bounded. Now by applying Lemma 5.7 we can find a set of halfspaces $H_e \subset \mathcal{H}(V)$ such that $h_0 \in \operatorname{conv} H_e$. Since h_0^0 contains at least one vertex of V, $|H_e| \leq d$. By Proposition 5.1, $\mathcal{P}(B \cup H_e)$ is bounded. □

5.3 The Dual-Nondegenerate Case

In this section we describe how the results of the previous section lead to a polynomial algorithm for facet enumeration of simple polytopes. We then give a refinement of this algorithm that yields an algorithm whose time complexity is competitive with the known algorithms for the primal nondegenerate case.

From the discussion above, we know that to achieve a polynomial algorithm for facet enumeration on a particular family of polytopes we need only have a polynomial algorithm for vertex enumeration for each subset of facet defining halfspaces of a polytope in the family. Dual-nondegeneracy (i.e. simplicity) is not quite enough in itself to guarantee this, but it is not difficult to see that the halfspaces defining any simple polytope can be perturbed so that they are in general position without affecting the combinatorial structure of the polytope. In this case each dual sub-problem is solvable by any number of pivoting methods (see e.g. [5, 85, 19]). Equivalently (and more cleanly) we can use lexicographic ratio testing (see Section 5.4.1) in the pivoting method. A basis is a subset of $\mathcal{H}(P)$ whose bounding hyperplanes define a vertex of P. Although a pivoting algorithm may visit many bases (or perturbed vertices) equivalent to the same vertex, notice that any vertex of the input is simple hence will have exactly one basis. It follows that we can again guarantee to find a witness or find all vertices of $\mathcal{P}(H_{cur})$ in at most $n + 1$ bases (where $n = |V|$, as before) output by the pivoting algorithm. In the case where each vertex is not too degenerate, say at most $d+\delta$ facets meet at every vertex for some small constant δ, we may have to wait for as many as $n \cdot \binom{d+\delta}{\delta} + 1$ bases. Of course this grows rather quickly as a function of δ, but is polynomial for δ constant. In the rest of this section we assume for ease of exposition that the polytope under consideration is simple.

It is not completely satisfactory to perform a vertex enumeration from scratch for each verification (FindWitness) step since each succeeding input to the vertex enumeration algorithm consists of adding exactly one halfspace to the previous input. We now show how to avoid this duplication of effort. We are given some subset $H_{cur} \subset \mathcal{H}(V)$ such that $\mathcal{P}(H_{cur})$ is bounded and a starting vertex $v \in \mathcal{V}(H_{cur})$ (we can use the raindrop algorithm to find a starting vertex in $O(|H_{cur}|d^2)$ time).

Algorithm 3 $\mathrm{dfs}(v, H_{cur})$

```
for j ∈ 1...d do
    v′ ← ComputeNeighbour(v, j, H_cur)
    if new(v′) then
        dfs(v′, H_cur)
    endif
endfor
```

Algorithm 3 is a standard pivoting algorithm for vertex enumeration using depth first search. The procedure $\mathrm{ComputeNeighbour}(v, j, H_{cur})$ finds the j-th neighbour of v in $\mathcal{P}(H_{cur})$. This requires $O(md)$ time to accomplish using a standard simplex pivot. To check if a vertex is new (i.e. previously undiscovered by the depth first search) we can simply store the discovered vertices in some standard data structure such as a balanced tree, and query this structure in $O(d \log n)$ time.

We could use Algorithm 3 as a subroutine to find witnesses for Algorithm 1, but we can also modify Algorithm 3 so that it finds new facets as a side effect. We are given a subset $H_0 \subset \mathcal{H}(V)$ as before and a starting vertex $v \in \mathcal{V}(H_0)$ with the additional restriction that v is a vertex of the input. In order to find a vertex of $\mathcal{P}(H_0)$ that is also a vertex of the input, we find an arbitrary vertex of $\mathcal{P}(H_0)$ using Lemma 5.1. If this vertex is not a vertex of the input then we apply DeleteVertex to find a new halfspace which cuts it off, and repeat. In what follows, we assume the halfspaces defining the current intermediate polytope are stored in some global dictionary; we sometimes denote this set of halfspaces as H_{cur}. We modify Algorithm 3 by replacing the call to ComputeNeighbour with a call to the procedure ComputeNeighbour2. In addition to the neighbouring vertex v', ComputeNeighbour2 computes the (at most one) halfspace defining v' not already known. Suppose we have found v (i.e. v is a vertex of the current intermediate polytope). Since P is simple we must have also found all of the halfspaces defining v. It follows that we have a halfspace description of each edge leaving v. Since we have a halfspace description of the edges, we can pivot from v to some neighbouring vertex v' of the current intermediate polytope. If $v' \in V$ then we know v' must be adjacent to v in conv V; otherwise we can cut v' off using our DeleteVertex routine. If P is simple, then no perturbation is necessary, since we will cut off degenerate vertices rather than trying to pivot away from them. Thus ComputeNeighbour2 can be implemented as in Algorithm 4.

Algorithm 4 ComputeNeighbour2(v, j, H_{cur})

```
repeat
    ṽ ← ComputeNeighbour(v, j, H_cur)
    If ṽ ∉ V then
        h ← DeleteVertex(ṽ, H_cur, V)
        AddToDictionary(h, H_cur)
    end if
until ṽ ∈ V
return ṽ
```

Lemma 5.9 *With* $O(mnd)$ *preprocessing,* ComputeNeighbour2 *takes time* $O(md + k(md + nd^2))$*, where* k *is the number of new halfspaces discovered.*

Proof. As mentioned above, ComputeNeighbour takes $O(md)$ time. The procedure AddToDictionary merges the newly discovered halfspace into the global dictionary. Since P is simple, we know the new halfspace will be strictly satisfied by the current vertex v; it follows that we can merge it into the dictionary by making the slack variable basic. This amounts to a basis transformation of the bounding hyperplane, which can be done in $O(d^2)$ time.

Since the search problem is completely static (i.e. there are no insertions or deletions), it is relatively easy to achieve a query time of $O(d + \log n)$, with a preprocessing cost of $O(n(d + \log n))$ using e.g. kd-trees [30]. We claim that the inequality $n \leq 2^{md}$ follows from the Upper Bound Theorem. For $0 \leq d \leq 3$ this is easily verified. For $d \geq 4$,

$$
\begin{aligned}
n &\leq 2\binom{m - \lfloor d/2 \rfloor}{\lfloor d/2 \rfloor} && \text{Upper Bound Theorem} \\
&\leq \frac{2m^{\lfloor d/2 \rfloor}}{\lfloor d/2 \rfloor !} \\
&\leq m^{d/2} = 2^{(d \log m)/2} && d \geq 4
\end{aligned}
$$

It follows that $d + \log n \leq 2md$, hence the query time is $O(md)$, and the preprocessing time is $O(nmd)$. Since each pivot in ComputeNeighbour2 that does not discover a vertex of V discovers a facet of $\operatorname{conv} V$, we can charge the time for those pivots to the facets discovered. $\square$

A depth first search based primal–dual algorithm is given in Algorithm 5. Note that we do not need an additional data structure or query step to determine if a v' is newly discovered. We simply mark each vertex as discovered when we search in ComputeNeighbour2. Furthermore, for P simple, $m \leq n$. Thus we have the following:

Theorem 5.3 *Given* $V \in \mathbb{R}^{n \times d}$, *if* $\operatorname{conv} V$ *is simple, we can compute* $H = \mathcal{H}(V)$ *in time* $O(n|H|d^2)$.

Algorithm 5 pddfs(v, H_0)

$H_{cur} \leftarrow H_0$
For $j \in 1 \dots d$ do
  $v' \leftarrow$ ComputeNeighbour2(v, j, H_{cur}) `add new halfspaces to` H_{cur}
  if new(v') then
    $H_{cur} \leftarrow H_{cur} \cup$ pddfs(v')
  endif
endfor
return H_{cur}

5.4 The Dual-Degenerate Case

We would like an algorithm that is useful for moderately dual-degenerate polytopes. In a standard pivoting algorithm for vertex enumeration based on depth or breadth first search, previously discovered bases must be stored. Since the number of bases is not necessarily polynomially bounded in the dual-degenerate case[1] we turn to *reverse search* [5] which allows us to enumerate the vertices of a non-simple polytope without storing the bases visited. The rest of this section is organized as follows. Section 5.4.1 explains how to use reverse search for vertex enumeration of non-simple polytopes via lexicographic pivoting. Section 5.4.2 shows how to construct a primal dual facet enumeration algorithm analogous to Algorithm 5 but with the recursion or stack based depth first search replaced by the "memoryless" reverse search.

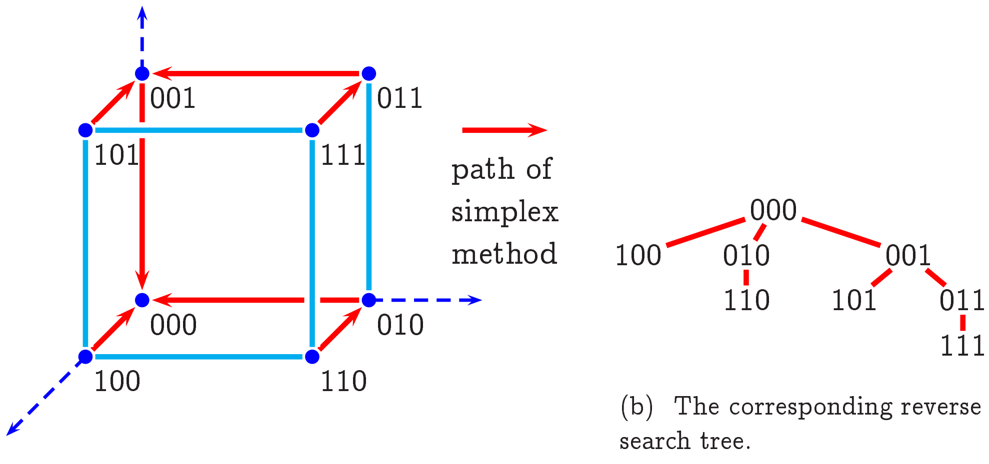
5.4.1 Lexicographic Reverse Search

The essence of reverse search in the simple case is as follows. Choose an objective function (direction of optimization) so that there is a unique optimum vertex. Fix some arbitrary pivot rule. From any vertex of the polytope there is a unique se-

[1]Even if the number of bases is bounded by a small polynomial in the input size, any super-linear space usage may be impractical for large problems.

quence of pivots taken by the simplex method to this vertex (see Figure 5.7a). If we take the union of these paths to the optimum vertex, it forms a tree, directed towards the root. It is easy to see algebraicly that the simplex pivot is reversible; in fact one just exchanges the roles of the leaving and entering variable. Thus we can perform depth-first search on the "simplex tree" by reversing the pivots from the root (see Figure 5.7b). No storage is needed to backtrack, since we merely pivot towards the optimum vertex.

Figure 5.7 Reverse search on a 3-cube



(a) The "simplex tree" induced by the objective $-\mathbb{1}$.

(b) The corresponding reverse search tree.

In this section we discuss a technique for dealing with degeneracy in reverse search. In essence what is required is a method for dealing with degeneracy in the simplex method. Here we use the method of *lexicographic pivoting* (also used in Avis' implementation of reverse search [42]), which can be shown to be equivalent to a standard symbolic perturbation of the constant vector b in the system $Ax \leq b$ (see e.g. [85] for discussion). Since the words "lexicographic" and "lexicographically" are somewhat ubiquitous in the remainder of this chapter, we sometimes abbreviate them to "lex".

In order to present how reverse search works in the non-simple case, we need to discuss in more detail the notions of dictionaries and pivoting used in Section 5.2. We start by representing a polytope as a system of linear equalities where all of the variables are constrained to be nonnegative. Let P be a d-polytope defined by a system of m inequalities. As before, convert each inequality to an equality by adding a slack variable. By solving for the original variables along with some set of $m-d$ slacks and eliminating the original variables from the $m-d$ equations with slack variables on the left hand side, we arrive at the *slack representation* of P. Geometrically, this transformation can be viewed as coordinatizing each point in the polyhedron by its scaled distance from the bounding hyperplanes. By renaming slack variables, we may assume that the slack representation has the form

$$(5.3) \qquad Ax = b \qquad \text{where} \qquad A = \begin{bmatrix} I & A' \end{bmatrix}, \qquad A' \in \mathbb{R}^{(m-d)\times d}.$$

For $J \subset \mathbb{Z}^+$ and vector x, let x_J denote the vector of elements of x indexed by J. Similarly, for matrix A, let A_J denote the subset of columns of A indexed by J. If $\operatorname{rank} A_J = |J| = \operatorname{rank} A$, we call J a *basis* for A, and call A_J a *basis matrix*. Suppose $B \subset \{1 \ldots m\}$ defines a basis of (5.3) (i.e. a basis for A). Let C (the *cobasis*) denote $\{1 \ldots m\} \setminus B$. We can rewrite (5.3) as

$$b = A_B x_B + A_C x_C .$$

Rearranging, we have the familiar form of a dictionary

$$(5.4) \qquad x_B = A_B^{-1} b - A_B^{-1} A_C x_C .$$

The solution $\beta = A_B^{-1} b$ obtained by setting the cobasic variables to zero is called a *basic* solution. If $\beta \geq \mathbb{O}$, then β is a called a *basic feasible solution* or *feasible basis*. Each feasible basis of (5.3) corresponds to a basis of some vertex of the corresponding polyhedron, in the sense of an affinely independent set of d supporting hyperplanes; by setting $x_i = 0$, $i \in C$ we specify d inequalities to be satisfied with equality. If the corresponding vertex is simple, then the resulting values for x_B will be strictly positive, i.e. no other inequality will be satisfied with equality. In the rest of this paper we use basis in the (standard linear programming) sense of a set of linearly independent columns of A and reserve *cobasis* for the corresponding set of supporting hyperplanes incident on the vertex (or equivalently, the set of indices of the corresponding slack variables). A *pivot* operation moves between feasible bases by replacing exactly one variable in the cobasis with one from the basis. To pivot to a new basis, start by choosing some cobasic variable x_k in C to increase. Let $\beta = A_B^{-1}b$ and let $A' = A_B^{-1}A_C$. The standard simplex ratio test looks for the first basic variable forced to zero by increasing x_k, i.e. it looks for

$$\min_{a'_{ik}>0} \frac{\beta_i}{a'_{ik}} .$$

In the general (non-simple) case, there will be ties for this minimum ratio. Define

$$L(B) \equiv \begin{bmatrix} \beta & A_B^{-1} \end{bmatrix}$$

$$L'(B)_{ij} \equiv \begin{cases} \infty & \text{if } a'_{ik} = 0 \\ L(B)_{i,j}/a'_{ik} & \text{otherwise} \end{cases}$$

To choose a variable to leave the basis, we find the lexmin row of $L'(B)$, i.e. we first apply the standard ratio test to β and then break ties by applying the same test to successive columns of $L(B)$. Intuitively, this simulates performing the standard ratio test in a perturbed system $Ax \leq b + \vec{\varepsilon}$ where $\vec{\varepsilon}_i = \epsilon^i$ for some $0 < \epsilon \ll 1$. This perturbation is equivalent to sequentially perturbing the hyperplanes in index order, with each successive hyperplane pushed outward by a smaller amount. That there is a unique choice for the leaving variable (i.e. that the corresponding vertex of the perturbed polytope is simple) follows from the fact that A_B^{-1} is non-singular.

A vector x is called *lexicographically positive* if $x \neq \mathbb{0}$ and the lowest indexed nonzero entry is positive. A basis B is called lexicographically positive if every row of $L(B)$ is lexicographically positive. Let B be a basis set and let C be the corresponding cobasis set. Given an objective vector ω, the *objective row* of a dictionary is defined by

$$\begin{aligned} z = \omega x &= \omega_B x_B + \omega_C x_C \\ &= \omega_B A_B^{-1} b + (\omega_C - \omega_B A_B^{-1} A_C) x_C \,. \qquad \text{by (5.4)} \end{aligned}$$

The simplex method chooses cobasic variable to increase with a positive coefficient in the *cost row* $\omega_C - \omega_B A_B^{-1} A_C$ (i.e. a variable x_j s.t. increasing x_j will increase the objective value z). Geometrically, we know that increasing a slack variable x_k will increase (resp. decrease) the objective function ωx iff the inner product of the objective vector with the outer normal of the corresponding halfspace h_k^+ is negative (resp. positive). Every cobasis C for vertex v defines a polyhedral cone P_C with apex v containing P. A cobasis is optimal for objective vector $\omega^* \in R^d$ if $(v+\omega^*) \in (v-P_C)$ (note that ω^* is the original objective vector before transforming to the slack representation). Reinterpreting in terms of the slack representation, we have the following standard result of linear programming (see e.g. [85]).

Proposition 5.2 *If the cost row has no positive entry, then the current basic feasible solution is optimal.*

If the entering variable is chosen with positive cost row coefficient, and the leaving variable is chosen by the lexicographic ratio test, we call the resulting pivot a *lexicographic pivot*. A vector v is *lexicographically greater* than a vector v' if $v-v'$ is lexicographically positive. The following facts are known about lexicographic pivoting:

Proposition 5.3 [72] *Let S be lexicographically positive basis, T be a basis arrived at from S by a lexicographic pivot, and ω a nonzero objective vector.*

(a) T *is lexicographically positive, and*

(b) $\omega_T L(T)$ *is lexicographically greater than* $\omega_S L(S)$.

A basis is called *lex optimal* if it is lexicographically positive, and there are no positive entries in the corresponding cost row. In order to perform reverse search, we would like a unique lex optimal basis. We claim that if $C = \{m - d + 1 \ldots m\}$, we can fix C as the unique lex optimal basis by choosing as the objective function $-\sum_{i \in C} x_i$. This is equivalent to choosing the mean of the outward normals of the hyperplanes in C as objective direction. If we consider an equivalent perturbed polytope, the intuition is that all of the perturbed vertices corresponding to a single original vertex are contained in the cone defined by the lex maximal cobasis (see Figure 5.8).

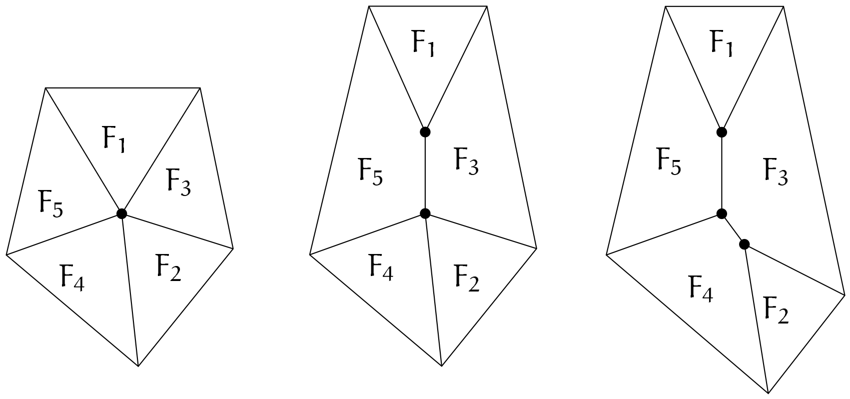
Lemma 5.10 *Let $S = \{1 \ldots m - d\}$ denote the initial basis defined by the slack representation. For objective vector $\omega = \left[\mathbb{O}^{m-d}, -\mathbb{1}^{d}\right]$, a lex positive basis B has a positive entry in the cost row if and only if $B \neq S$.*

Proof. The cost row for S is $-\mathbb{1}^d$. Let B be a lex positive basis distinct from S, and let β denote the basic part of the corresponding basic feasible solution. Let k denote the number of non-identity columns in A_B. If $\omega_B \beta < 0$, then there must be some positive entry in the cost row since β is not optimal. Suppose that $\omega_B \beta = 0$. It follows that $\beta = \left[\beta', \mathbb{O}^k\right]$ since $\omega_B = \left[\mathbb{O}^{m-d-k}, -\mathbb{1}^k\right]$ and $\beta \geq \mathbb{O}$. Let j be the first column of A_B that is not column j of an $(m - d) \times (m - d)$ identity matrix. Let $a = [\mathbb{O}, \hat{a}]$ denote row j of A_B. Since the first $m - d - k$ columns of A_B are identity columns, $\hat{a}$ is a k-vector. Let $\alpha = [\alpha', \hat{\alpha}]$ be column j of A_B^{-1}, where $\hat{\alpha}$ is also a k-vector. Since $\hat{a}\hat{\alpha} = 1$, we know $\hat{\alpha} \neq \mathbb{O}$. By the lex positivity of $L(B)$, along with the fact that $\beta = \left[\beta', \mathbb{O}^k\right]$ and the fact that the first $j - 1$ columns of A_B^{-1} are identity columns, it follows that $\hat{\alpha}$ has no negative entries. It follows that element j of $\omega_B A_B^{-1}$ is negative. Since identity column j is not in A_B, it must be present in A_C, in position $j' < k$. It follows that element j' of $\omega_B A_B^{-1} A_C$ is negative, hence element j' of the cost row is positive. □

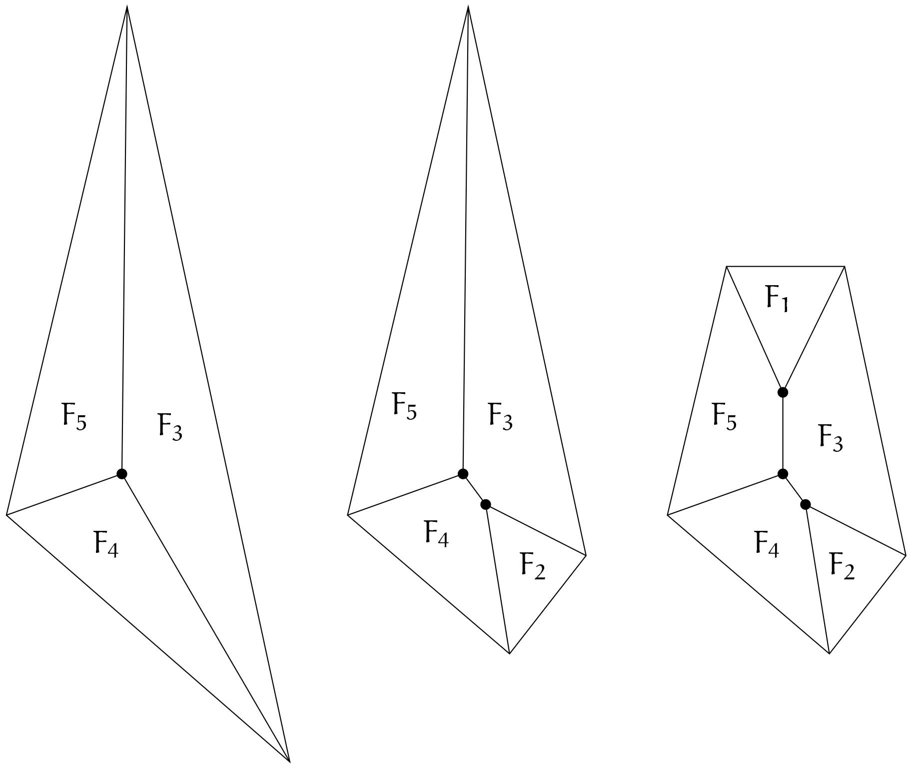
From the preceding two lemmas, we can see that the lexicographically positive bases can be enumerated by reverse search from a unique lex optimal basis. The following tells us that this suffices to enumerate all of the vertices of a polytope.

Lemma 5.11 *Every vertex of a polytope has a lexicographically positive basis.*

Figure 5.8 Lexicographic perturbation and incremental construction



(a) Sequentially perturbing the halfspaces defining a vertex



(b) Intersecting the perturbed halfspaces in reverse order

Proof. Let P be a polytope. Let v be an arbitrary vertex of P. Choose some objective function so that v is the unique optimum. Choose an initial lex positive basis. Run the simplex method with lexicographic pivoting. Since there are only a finite number of bases, and by Proposition 5.3 lexicographic pivoting does not repeat a basis, we must eventually reach some basis of v. Since lexicographic pivoting maintains a lex positive basis at every step, this basis must be lex positive. □

Algorithm 6 gives a schematic version of the lexicographic reverse search algorithm. We rename variables so that $C_{\mathrm{opt}} = \{m - d + 1 \ldots m\}$ is a cobasis of the initial vertex v_0. The routine PivotToOpt does a lexicographic pivot towards this cobasis with the objective function $\omega = \left[\mathbb{0}^{m-d}, -\mathbb{1}^{d}\right]$. If there is more than one cobasic variable with positive cost coefficient, then we choose the one with lowest index. PivotToOpt returns not only the new cobasis, but the index of the column of the basis that entered. The test $\mathrm{IsPivot}(C', C)$ determines whether $(C, k) = \mathrm{PivotToOpt}(C')$ for some k.

Algorithm 6 $\mathrm{ReverseSearch}(H, v_0)$

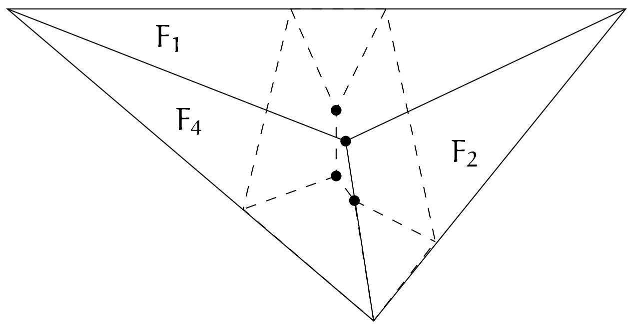
```
C ← C_opt, j ← 1, AddToDictionary(H_0, H_cur)
repeat
    while j ≤ d
        C′ ← ComputeNeighbour(C, j, H_cur)
        if IsPivot(C′, C) then
            C ← C′, j ← 1                                    down edge
        else
            j ← j + 1                                        next sibling
        end if
    end while
    (C, j) ← PivotToOpt(C)                                   up edge
    j ← j + 1.
until j > d and C = C_opt
```

As before, we could use Algorithm 6 to implement a verification (FindWitness) step by performing a vertex enumeration from scratch. In the next section, we discuss how to construct an algorithm analogous to Algorithm 5 that performs only a single vertex enumeration, but which uses reverse search instead of a standard depth first search.

5.4.2 Primal–Dual Reverse Search

In this section we give a modification of Algorithm 6 that computes the facet defining halfspaces as a side effect. Define pdReverseSearch(H_0, v_0) as Algorithm 6 with the call to ComputeNeighbour replaced with a call to ComputeNeighbour2. As in Section 5.3, we suppose that preprocessing steps have given us an initial set of facet defining halfspaces H_0 such that $\mathcal{P}(H_0)$ is bounded and there is some v_0 that is a vertex of the input and of $\mathcal{P}(H_0)$. It will turn out that the numbering of halfspaces is crucial. We number the jth halfspace discovered (including preprocessing) as $m - j$ (of course, we do not know what m is until the algorithm completes, but this does not prevent us from ordering indices). This reverse ordering corresponds to pushing later discovered hyperplanes out farther, thus leaving the skeleton of earlier discovered vertices and edges undisturbed; compare Figure 5.8b where halfspaces are numbered as in pdReverseSearch, with Figure 5.9, where a different insertion order causes intermediate vertices to be cut off.

Figure 5.9 A perturbed vertex cut off by later halfspaces.



The modified algorithm pdReverseSearch can be considered as a simulation of a "restricted" reverse search algorithm for vertex enumeration where we are given access only to a subset of the halfspaces, and where the "input halfspaces" are labeled in a special way. Since the lexicographic reverse search described in the previous section works for each labeling of the halfspaces, to show that the restricted reverse search correctly enumerates the vertices of P, we need only show that it visits the same set of cobases as the unrestricted algorithm would, if given the same labeling and initial cobasis.

Let $Ax = b$, $A \in \mathbb{R}^{(m-d)\times m}$ be the slack representation of a d-polytope. We can write the slack representation in homogeneous form $S = [I\ A' \ -b]$ where $A' \in \mathbb{R}^{(m-d)\times d}$. Suppose at some step of the restricted reverse search the lowest index halfspace visited (including initialization) is $k+1$. The restricted reverse search therefore has access to all of S except for the first $k-1$ rows and the first $k-1$ columns.

Let K denote $\{k \dots m\}$ for some $k \leq m-d$. For any cobasis $C \subset K$, let $\widehat{B}$ denote $K \setminus C$. We define the k-*restricted basis matrix* for C as the last $m-d-k+1$ rows of $A_{\widehat{B}}$. Let R denote the k-restricted basis matrix for C, and let ρ denote $R^{-1}b_K$. By the k-*restricted lexicographic ratio test* we mean the lexicographic ratio test applied to the matrix $[\rho\ R^{-1}]$. By way of contrast we use the *unrestricted* lexicographic ratio test or basis matrix to mean the previously defined lexicographic ratio test or basis matrix.

We observe that the restricted basis matrix is a submatrix of the unrestricted basis matrix for a given cobasis, and that this property is preserved by matrix inversion. Let R denote the k-restricted basis for C. Let U denote the (unrestricted) basis matrix for C. Since $k \leq m-d$, and the first $m-d$ columns of the slack representation form an identity matrix, we know columns of U before k must be identity columns. It follows that

$$U = \begin{bmatrix} I & M \\ \mathbb{O} & R \end{bmatrix}$$

for some matrix M. The reader can verify the following matrix identity.

$$U^{-1} = \begin{bmatrix} I & M \\ \mathbb{O} & R \end{bmatrix}^{-1} = \begin{bmatrix} I & -MR^{-1} \\ \mathbb{O} & R^{-1} \end{bmatrix}. \tag{5.5}$$

The edges of the reverse search tree are pivots. Referring to our interpretation of lex pivoting as a perturbation, in order that both versions of the reverse search generate the same perturbed vertex/edge tree, they must generate the same set of pivots. We argue first that choosing the same hyperplane to leave the cobasis (i.e. edge to leave the perturbed vertex), yields the same hyperplane to enter the cobasis in both cases (i.e. the same perturbed vertex).

Lemma 5.12 *Let* P *be a* d*-polytope and let* $Ax = b$ *be the slack representation of* P*. Let* $C \subset \{k \dots m\}$ *be a cobasis for* $Ax = b$*. For* $k \leq m - d$ *and for any entering variable* x_s*, if there is a candidate leaving variable* x_t *with* $t \geq k$ *then the leaving variable chosen by the lexicographic ratio test is identical to that chosen by the* k*-restricted lexicographic ratio test.*

Proof. Let β denote $U^{-1}b$. As above, let ρ denote $R^{-1}b_K$. One consequence of (5.5) is that $\rho = \beta_K$. If there is exactly one candidate leaving variable, then by the assumptions of the lemma it must have index at least k, and both ratio tests will find the same minimum. If on the other hand there is a tie in the minimum ratio test applied to β then a variable with index at least k will always be preferred by the unrestricted lexicographic ratio test, since in the columns of U^{-1} with index less than k, these variables will have ratio 0. □

The up (backtracking) edges in the reverse search tree consist of of lex pivots where the lowest indexed cobasic variable with positive cost coefficient is chosen as the entering variable (i.e. lowest indexed tight constraint that can profitably be loosened). The previous lemma tells us that for a fixed entering variable and cobasis, the restricted and unrestricted reverse search will choose the same leaving variable. It remains to show that in a backtracking pivot towards the optimum cobasis they will choose the same entering variable. Given a fixed set of halfspaces $\{h_1^+, h_2^+, \ldots, h_d^+\}$ (a cobasis) and a fixed vector ω^* (direction of optimization), the signs of the cost vector depend only on the signs of $\omega^* h_i$, $1 \le i \le d$. We can in fact show something slightly stronger, since our objective vector ω (with respect the slack representation) does not involve hyperplanes with index less than k. As above, let $K = \{k \ldots m\}$ and $\widehat{B} = K \setminus C$. Analogous to the definition of a k-restricted basis matrix, we define the k*-restricted cost row* for cobasis C as $\omega_C - \omega_{\widehat{B}} R^{-1} \widehat{A}_C$ where R is the k-restricted basis matrix and $\widehat{A}_C$ is the last $m-d-k+1$ rows of A_C.

Lemma 5.13 *For objective vector $\omega = [\mathbb{O}^{m-d}, \omega']$, for $k \le m-d$, the cost row and the k-restricted cost row are identical.*

Proof. As before, let R and U be the restricted and unrestricted basis matrices respectively. From the form of the objective vector, we know $\omega_B = [\mathbb{O}^{k-1}, \omega_{\widehat{B}}]$. By (5.5),

$$\begin{aligned} \omega_B U^{-1} A_C &= \begin{bmatrix} \mathbb{O}^{k-1} & \omega_{\widehat{B}} \end{bmatrix} \begin{bmatrix} I & -MR^{-1} \\ \mathbb{O} & R^{-1} \end{bmatrix} \begin{bmatrix} A' \\ \widehat{A}_C \end{bmatrix} \\ &= \omega_{\widehat{B}} R^{-1} \widehat{A}_C \end{aligned}$$

□

In the case of down edges in the reverse search tree, each possible entering variable (hyperplane to leave the cobasis) is tested in turn, in order of increasing index. Thus if the previous backtracking pivot to a cobasis was identical in the two algorithms, the next down edge will be also. Reverse search is just depth first search on a particular spanning tree; hence it visits the nodes of the tree in a sequence (with repetition) defined by the ordering of edges. The ordering of edges at any node in the reverse search tree is in turn determined by the numbering of hyperplanes.

Lemma 5.14 *Let P be a polytope. Let H_0 be a subset of $\mathcal{H}(P)$ with bounded intersection. Let $v_0 \in \mathcal{V}(H_0) \cap \mathcal{V}(P)$. The set of cobases visited by* pdReverseSearch(H_0, v_0) *is the same as that visited by* ReverseSearch$(\mathcal{H}(P), v_0)$ *if* ReverseSearch *is given the same halfspace numbering.*

Proof. We can think of the sequences of cobases as chains connected by pivots. Let $\mathcal{C}_r = \langle C_1, C_2, \ldots \rangle$ be the chain of cobases visited by pdReverseSearch(H_0, ν_0). Let $\mathcal{C}_u$ be the chain of cobases visited by ReverseSearch$(\mathcal{H}(P), \nu_0)$. Both sequences start at the same cobasis since the starting cobasis is the one with the lex maximum set of indices. Now suppose the two sequences are identical up to element j; further suppose that $\bigcup_{i \le j} C_i = k+1 \ldots m$. There are two cases. If the edge in $\mathcal{C}_r$ from C_j to C_{j+1} is a reverse (down) edge, then we start by pivoting from C_j to C_{j+1} by fixing some entering variable and choosing the leaving variable lexicographically. C_{j+1} contains at most one variable not present in C_i, $i \le j$; this variable is numbered k, if present. Let s denote the position of the entering variable in C_j (i.e. the column to leave the cobasis matrix). Since the cobasis in position C_j will have occurred $s-1$ times in both sequences, we know that ReverseSearch and pdReverseSearch will choose the same entering variable. By Lemma 5.12, they will choose the same leaving variable. The test IsPivot(C_{j+1}, C_j) depends only on the cost row, so by Lemma 5.13 the next cobasis in $\mathcal{C}_u$ will also be C_{j+1}. Suppose on the other hand the pivot from C_j to C_{j+1} is a forward pivot. We know from Lemma 5.13 that both invocations will choose the same entering variable, and we again apply Lemma 5.12 to see that they will choose the same leaving variable. □

Theorem 5.4 *Given $V \in \mathbb{R}^{n \times d}$, let m denote $|\mathcal{H}(V)|$, and let ϕ denote the number of lexicographically positive bases of $\mathcal{H}(V)$.*

(a) *we can compute $\mathcal{H}(V)$ in time $O(\phi m d^2)$ and space $O((m+n)d)$.*

(b) *we can decide if* conv V *is simple in time $O(n^2 d + n d^3)$.*

Proof

(a) Total cost for finding an initial set of halfspaces is $O(nkd^2)$, where k is the size of the initial set. Since every DeleteVertex call finds a new halfspace, the total cost for these calls is $O(nmd^2)$. In every call to ComputeNeighbour2, each pivot except the last one discovers a new halfspace. Those which discover new halfspaces have total cost $O(m^2 d)$ which is $O(\phi m d)$; the other pivots cost $O(\phi m d^2)$ as there are ϕd calls to ComputeNeighbour2. The ϕ forward pivots (PivotToOpt) cost $O(\phi m d)$.

(b) At any step of the reverse search, we can read the number of halfspaces satisfied with equality by the current vertex off the dictionary in $O(m)$ time. From the Lower Bound Theorem [78] for simple polytopes, if P is simple then $m \leq 2(n/d + d)$ for $d \geq 2$. If we reach a degenerate vertex, or discover more than $2(n/d + d)$ facets, we stop. If the reverse search terminates, then in $O(nmd)$ time we can compute the number of facets meeting at each vertex. The total cost is thus $O(n(n/d + d)d^2) = O(n^2 d + nd^3)$. □

Theorem 5.4b is of independent interest since the problem of given H, deciding whether $\mathcal{P}(H)$ is simple is known to be NP-complete in the strong sense [24].

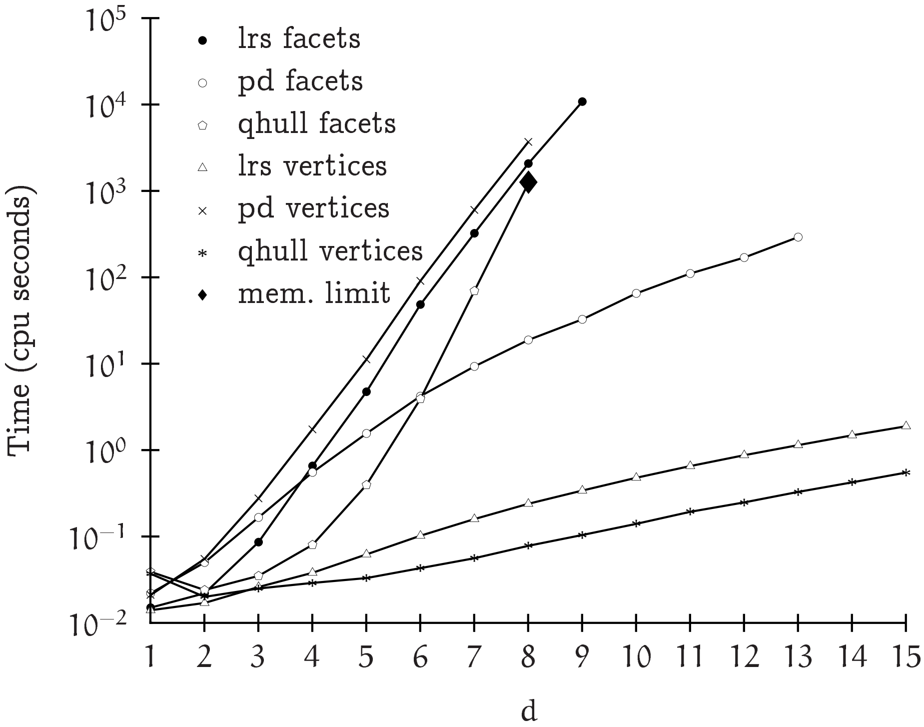
5.5 Experimental Results

Ambros Marzetta has implemented Algorithm 6 using rational arithmetic in C. The memory requirements of this implementation (ignoring the number of bits to store each integer) are twice the input size plus twice the output size, as the program stores four dictionaries: a constant vertex dictionary V and a growing halfspace dictionary H_{cur} in unpivoted form, and a working copy of both. The program uses an earlier version of the preprocessing step, with an upper bound of $O(nd^4)$, compared to the current bound of $O(nd^3)$. The source code is available at `http://wwwjn.inf.ethz.ch/ambros/pd.html`. All of the experiments have been performed on a Digital AlphaServer 4/233 with 256M of real memory and 512M of virtual memory.

In what follows **pd** is Marzetta's implementation of Algorithm 6, **lrs** is Avis' implementation of reverse search [42], and **qhull** is Barber et al.'s incremental algorithm which uses triangulation. Figure 5.10 compares the running time of the three programs on products of two simplices. These 2d-dimensional polytopes have $2d + 2$

facets and $(d+1)^2$ vertices. They are simple (which is ideal for vertex enumeration by **lrs** and facet enumeration by **pd**), but have extremely high triangulation complexity, which is bad for vertex enumeration by **pd** and facet enumeration by **lrs**, because the perturbation of the vertices made by the algorithms induces a triangulation of the polytope's boundary. On the plot, we show the times for enumerating both the facets and the vertices. As expected, **pd** is clearly superior to **lrs** and **qhull** for facet enumeration of these polytopes. Their very few facets are all found by the preprocessing of the current implementation; in fact this accounts for most of the time taken by **pd** on these examples.

Figure 5.10 Running time for products of simplices $T_d \times T_d$



A less asymmetric example is the product of cyclic polytopes $\dddot{C}_{2d}(n)$. In Chapter 3 we saw that both the primal and the dual triangulations of these polytopes are superpolynomial; nonetheless experimentally it seems that the dual triangulations are smaller than the primal ones (see Figure 5.11). Apparently this constant factor is not enough to make up for the difference between floating point arithmetic (**qhull**) and exact rational arithmetic (see Figure 5.12); on the other hand **pd** is able to complete larger problem sets using the same amount of memory.

Figure 5.11 Triangulation size (number of bases computed) of $C_4(n) \times C_4(n)$.

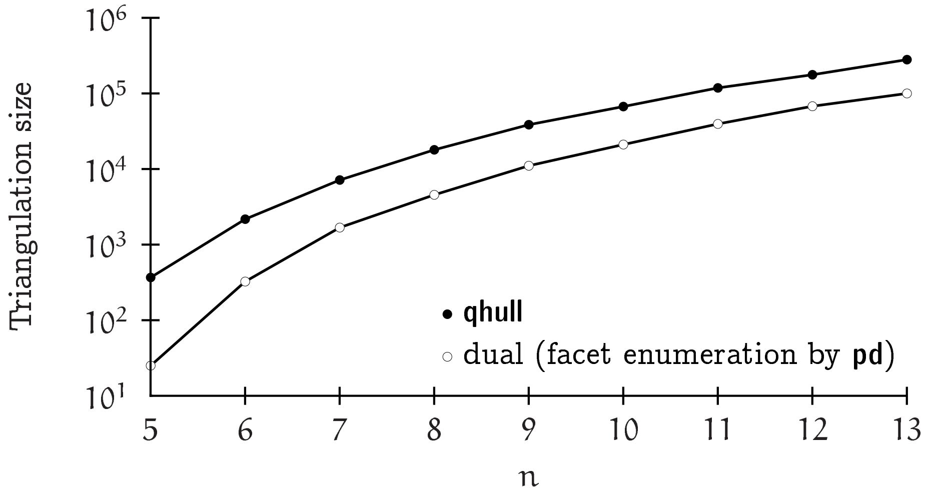
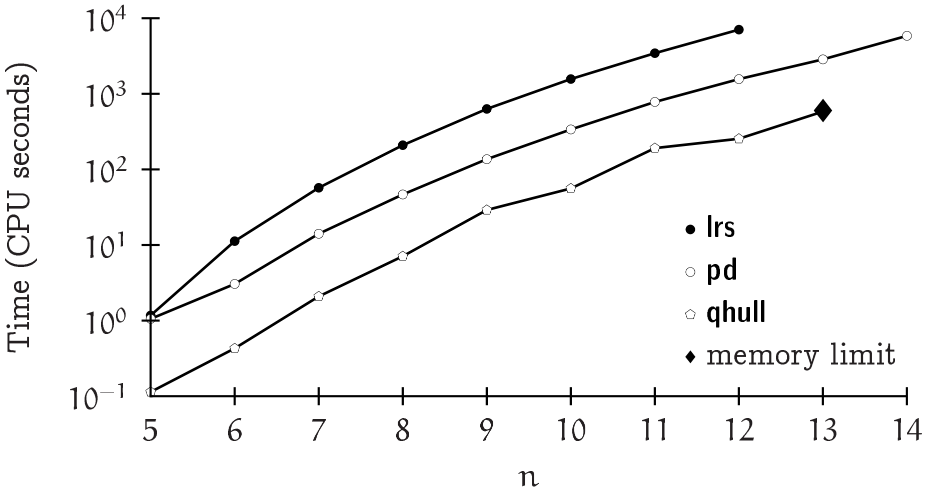


Figure 5.12 CPU time for facet enumeration of $C_4(n) \times C_4(n)$.



5.6 Conclusions

An alternative approach to achieving an algorithm polynomial for the dual-nondegenerate case is to modify the method of Gritzmann and Klee [27]. An idea due to Clarkson [15] can be used to reduce the row size of each of these linear programs to $O(m')$ where m' is the maximum number of facets meeting at a vertex. If we assume that $m' \leq d + \delta$ for some constant δ then we can solve each linear program by brute force in time polynomial in d. It seems that such an approach will be inherently quadratic in the input size since the entire set of input halfspaces is condered to enumerate the vertices of each facet.

It would be interesting to remove the requirement in Theorem 5.1 that the family be facet-hereditary, but it seems difficult to prove things in general about the polytopes formed by subsets of the halfspace description of a known polytope.

Bibliographic Notes

A preliminary version of the material in this chapter first appeared in [8].

Chapter 6

Conclusions

6.1 Summary of Thesis

For any convex polytope P, there are two fundamental algorithmic problems. Transforming from the halfspace representation to the vertex representation is called *vertex enumeration*. Transforming from the vertex representation to the halfspace representation is called *facet enumeration*. A theoretically and practically important question is whether there are algorithms for these problems whose time complexity is polynomial in the input and the output size. Answering this question seems quite difficult; the goals of this thesis are to better understand the performance of existing methods, and to provide algorithms whose performance is polynomial for new (restricted) classes of input. For each known class $\mathcal{C}$ of algorithms, we give a measure $m_{\mathcal{C}}(P)$ with the property that the time complexity an algorithm in class $\mathcal{C}$ on input P is bounded (above, below, or both) by a polynomial in $m_{\mathcal{C}}(P)$. For each such measure, we provide one or more example families of polytopes where the measure (and hence the corresponding class of algorithms) is superpolynomial in the input and output size. It is well known from duality of convex polytopes that a polynomial algorithm for vertex (respectively facet) enumeration of a family Γ of polytopes provides a polynomial algorithm for facet (respectively vertex) enumeration of a family Γ^* dual to Γ. On the other hand, the relationship between the complexity of vertex enumeration for a family Γ and the complexity of facet enumeration for Γ itself has not been widely studied. In this thesis we show that under certain restrictions, an efficient algorithm for vertex (respectively facet) enumeration for Γ can be used as a subroutine to provide an efficient algorithm for facet (respectively vertex) enumeration for Γ.

Broadly speaking, known algorithms for vertex/facet enumeration can be divided into those that enumerate the face lattice, those that use triangulation or perturbation, and those that use incremental construction. We consider three corresponding measures for a d-polytope P:

1. The lattice complexity $\bar{f}(P)$ is defined as the number of non-empty faces of P.

2. The triangulation complexity $\text{tmin}_F(P)$ is the minimum number of $(d-1)$ simplices necessary to triangulate the boundary of P.

3. The intermediate vertex (respectively facet) complexity of P is the maximum size of an intermediate polytope constructed by an incremental vertex (respectively facet) enumeration algorithm.

In all three cases we considered the asymptotic behaviour of these measures relative to the input and output size of the polytope. For convenience we defined the function $\text{size}(P)$ as $d(m+n)$ where m and n are the number of facets and vertices respectively of P.

In Chapter 3 we considered the relationship between lattice complexity, triangulation complexity and perturbation. In general, algorithms for vertex or facet enumeration that enumerate the face lattice will perform well (badly) on families of polytopes with low (high) lattice complexity. That is, there are algorithms to compute the face lattice (see e.g. [24, 39]) whose complexity is polynomial in $\bar{f}(P)$ and $\text{size}(P)$. Facet enumeration algorithms that use perturbation or triangulation will perform well (badly) on families with low (high) triangulation complexity. In particular, pivoting algorithms for facet enumeration such as [5, 11, 85, 19, 37, 39] will be polynomial or superpolynomial depending on the triangulation complexity.

By duality, the same algorithms will be polynomial or superpolynomial for vertex enumeration depending on the triangulation complexity of the dual polytope. Furthermore, in Chapter 5 we argued that if either the primal or the dual triangulation complexity is bounded above by a polynomial then both facet and vertex enumeration are solvable in polynomial time. In Table 6.1 we summarize the bounds presented in this thesis related to lattice, triangulation, and dual triangulation complexity. In this table and the next, s denotes $\text{size}(P) = (m+n)d$, which corresponds to the number of real or rational numbers required to write down the input and the output. The column "bits" is the number of bits required to write down a single coordinate. For families of polytopes with two parameters, asymptotic statements are for d fixed. Of course one could always fix the other parameter to some function of d. For families with only one parameter, namely d, in statements involving s and d, d can be regarded as some slowly growing but non constant function of s.

The performance of incremental algorithms for vertex/facet enumeration depends on the size of the intermediate polytopes constructed. It turns out that for some families this depends on the order the input is processed (the *insertion order*). Even for families where every insertion order causes a superpolynomial blowup, experimentally it seems some insertion orders are much worse than others (see Section 4.4.3). Table 6.2 summarizes the results of Chapter 4 regarding incremental constructions. Unlike the case of lattice and triangulation based bounds, for each family we have theoretical bounds for at most one one of vertex and facet enumeration. For several families, we present experimental results for one or more directions where no theoretical bounds are given. Insertion orders for which upper or lower bounds are given include lexmin (i.e. lexicographically), random, maxcutoff (insert the point or halfspace that causes the largest drop in intermediate size), and minindex (insert the input in the order given). Statements about minindex insertion order can be considered existential statements: if minindex is good (bad) then there is a good (bad) insertion order.

Table 6.1 Summary of bounds for triangulation and lattice complexities

description	notn.	lattice size	$\triangle$ size	dual $\triangle$ size	bits
prod. of cyclic polytopes	$\dddot{C}_{2d}(n)$	$\Theta(s^{\sqrt{d}})$ [1]	$\Omega(s^{\sqrt{d}})$ [2]	$\Omega(s^{\sqrt{d}})$ [2]	$\Theta(n)$
prod. of sum of polygons	$\dddot{\Pi}_{2d}(n)$	$\Theta(s^{\sqrt{d}})$ [1]	$\Omega(s^{\sqrt{d}})$ [2]	$\Omega(s^{\sqrt{d}})$ [2]	$\Theta(\log n)$
prod. of sum of 3-cubes	$\dddot{U}_{3d}$	$\Theta(s^{3\sqrt{d}/\log d})$ [1]	$\Omega(s^{\sqrt{d}/\log d})$ [2]	$\Omega(s^{\sqrt{d}/\log d})$ [2]	1
product of simplices	$\dddot{T}_{2d}$	$\geq s^{d/(2\log d)}$ [3]	$\geq s^{d/(2\log d)}$ [4]	$\frac{s}{d} - 2d - 2$ [5]	1
d-cubes	H_d	$\leq s^{\log 3}$ [6]	$\geq s^{c\log d}$ [4]	$\frac{s}{d} - 2d$ [5]	1
pierced cubes	$\widehat{T}_d \cap H_d$	$\geq s^{d/(4\log d)}$ [3]	*high* [7]	*high* [7]	$\Theta(\log m)$
dwarfed cubes	$\overline{T}_d$	$\geq s^{d/(4\log d)}$ [3]	*high* [8]	$o(s)$ [5]	$\Theta(1)$
dwarfed dual cyclic polytopes		$\geq s2^{\Theta(d)}$ [9]		$o(s)$ [5]	$\Omega(m)$
dwarfed products of polygons		$\geq s2^{\Theta(d)}$ [9]	*high* [8]	$o(s)$ [5]	$\Theta(\log m)$

(1) Assuming $d = k^2$, k integer. §3.1

(2) Assuming $d = k^2$, k integer. §3.2

(3) P or P^* has a simplicial facet, but size(P) is polynomial in d. p. 22

(4) By volume ratios. §3.2.1

(5) Simple polytope.

(6) §3.2.1

(7) Experimentally. See §4.2.2

(8) Experimentally. §5.5

(9) This is only superpolynomial if s subexponential in d. §3.1

(10) *italic* text denotes experimental results

Table 6.2 Summary of bounds for insertion algorithms

description	notn.	Vertex Enumeration		Facet Enumeration	
		bound	order	bound	order
prod. of cyclic polytopes	$\dddot{C}_{2d}(n)$			$\Omega(s^{\sqrt{d}-1})$ [1]	all
prod. of sum of polygons	$\dddot{\Pi}_{2d}(n)$	*good* [2]	lexmin, maxcutoff	$\Omega(s^{\sqrt{d}-1})$ [1]	all
prod. of sum of 3-cubes	$\dddot{U}_{3d}$			$\geq s^{\sqrt{d}/\log d-1}$ [1]	all
product of simplices	$\dddot{T}_{2d}$	*good* [2]	lexmin, maxcutoff	*good* [2]	lexmin
d-cubes	H_d	*good* [2]	lexmin, maxcutoff	*good* [2]	lexmin
pierced cubes	$\widehat{T}_d \cap H_d$	$\geq s^{d/(4\log d)}$ [3]	minindex		
dwarfed cubes	$\overline{T}_d$	$\geq s^{d/(4\log d)}$ [4]	minindex, lexmin, random		
		$\leq s$ [5]	lexmin maxcutoff		
dwarfed dual cyclic polytopes		$\Omega(s^d)$ [4]	minindex, lexmin, random		
dwarfed products of polygons		$\Omega(s^d)$ [4]	minindex, lexmin, random		

(1) Assuming $d = k^2$, k integer. §4.4

(2) Experimentally. §4.4.3

(3) §4.2

(4) §4.3

(5) §4.3.1

(6) *italic* text indicates experimental results

6.2 Future Work

Our understanding of incremental algorithms is still much less satisfactory than for those based on pivoting. Experimental results (see e.g. Table 4.3) suggest that for certain families of polytopes, facet enumeration is much easier than vertex enumeration for incremental algorithms. It would thus be interesting to consider a primal-dual incremental algorithm along the lines of those presented in Chapter 5, where the finding of witnesses is done by an incremental algorithm rather than by pivoting. Unfortunately there are few families for which incremental algorithms are known (in a theoretical sense) to work well for either transformation. Even when polynomial bounds are known for incremental algorithms on a particular family, it is usually only for particular insertion orders (see e.g. [57]). It would be particularly interesting to better understand the relationship (if any) between degeneracy and intermediate size. One curious experimental result is the maxcutoff insertion order seems to be very bad for facet enumeration of polytopes with high triangulation complexity.

The more general question of whether there is a polynomial algorithm for vertex/facet enumeration remains open. If the answer should turn out to be negative, or if the problem continues to defy resolution, then a fruitful line of research is probably to continue to find algorithms that behave well for restricted classes of input.

While there is evidence (see e.g. [56, 57, 52, 54, 61, 62]) that many polytopes that occur in applications are degenerate, and have high intermediate complexity for standard insertion orders, a more systematic experimental study would no doubt be valuable.

Index of Notation

Bibliography

I Algorithms

See also: [43, 85, 99]

[1] W. Altherr. An algorithm for enumerating the vertices of a convex polyhedron. *Computing*, 15:181–193, 1975.

[2] David Avis and David Bremner. How good are convex hull algorithms? In *Proceedings of the 11th ACM Symposium on Computational Geometry*, pages 20–28, 1995.

> Provides classes of hard polytopes for pivoting algorithms using perturbation, algorithms using triangulation, and some insertion algorithms. Gives experimental results for these hard classes on current implementations.

[3] David Avis and David Bremner. How good are convex hull algorithms. Technical Report SOCS-95.1, School of Computer Science, McGill University, Montreal, Canada, 1995.

[4] David Avis, David Bremner, and Raimund Seidel. How good are convex hull algorithms? *Computational Geometry: Theory and Applications*, 7(5–6):265–301, April 1997. `http://www-cgrl.cs.mcgill.ca/~bremner/papers/abs96.ps.gz`

[5] David Avis and Komei Fukuda. A pivoting algorithm for convex hulls and

vertex enumeration of arrangements and polyhedra. *Discrete and Compututational Geometry*, 8:295–313, 1992.

Describes the notion of *reverse search*. By considering the paths taken by the simplex method from every vertex to some optimal vertex, an implicit tree is induced on the vertices (or bases) of the polytope. Depth first search on this tree allows generation of all feasible bases with memory usage proportional to the input.

[6] K.H. Borgwardt. Average complexity for determining the convex hull of randomly given points. *Discrete and Compututational Geometry*, 17:79–109, 1997.

Gives a probabilistic analysis of a gift wrapping algorithm (see e.g. [39, 11]) on random input. The claim is that for random input, redundant points need not be removed by preprocessing.

[7] David Bremner. Incremental convex hull algorithms are not output sensitive. In *Proceedings of the International Symposium on Algorithms and Computation '96*, volume 1178 of *Lecture Notes on Computer Science*, pages 26–35. Springer-Verlag, December 1996. `http://www-cgrl.cs.mcgill.ca/~bremner/papers/b96.ps.gz`

Gives examples of families of polytopes for which any order of incremental construction builds intermediate polytopes with size superpolynomial in the input and output size.

[8] David Bremner, Komei Fukuda, and Ambros Marzetta. Primal dual methods for vertex and facet enumeration (preliminary version). In *Proceedings of the 13th annual ACM Symposium on Computational Geometry*, pages 49–56, 1997. `http://www-cgrl.cs.mcgill.ca/~bremner/papers/bfm97.ps.gz`

[9] E. Burger. Über homogene lineare Ungleichungssysteme. *Zeitschrift für Angewandte Mathematik und Mechanik*, 36:135–139, 1956.

Contains a description of the double description method.

[10] Timothy M. Chan, Jack Snoeyink, and Chee-Keng Yap. Primal dividing and dual pruning: Output-sensitive construction of 4-d polytopes and 3-d voronoi diagrams. In *Proceedings of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 282–291, 1995.

[11] D.R. Chand and S.S. Kapur. An algorithm for convex polytopes. *Journal of the ACM*, 17:78–86, 1970.

A (dual) pivoting algorithm, also known as “gift wrapping”. Analyzed in [39].

[12] A. Charnes. The simplex method: optimal set and degeneracy. In *An introduction to Linear Programming*, Lecture VI, pages 62–70. Wiley, New York, 1953.

Suggests a method for enumerating the optimal solutions to a linear program based on the simplex pivots and depth first search. Includes a perturbation procedure.

[13] Bernard Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete and Compututational Geometry*, 10:377–409, 1993.

An insertion based algorithm that achieves a time bound of $O(n^{\lfloor d/2 \rfloor})$ in fixed dimension d.

[14] N.V. Chernikova. Algorithm for finding a general formula for the non-negative solutions of a system of inequalities. *U.S.S.R. Comput. Math. Phys.*, 5:228–233, 1965.

The double description method [34], with the implementation simplifying assumption that the variables are non-negative.

[15] Kenneth L. Clarkson. More output-sensitive geometric algorithms. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, pages 695–702, 1994. `http://www.cs.att.com/netlib/att/cs/doc/94/2-13.ps.Z`

Gives a method for finding extreme points in d dimensions where the row size of the linear programs solved is bounded above by the output size. Uses this method in a “convex hull” algorithm that computes the face lattice.

[16] Kenneth L. Clarkson and Peter W. Shor. Algorithms for diametral pairs and convex hulls that are optimal, randomized, and incremental. In *Proceedings of the 4th ACM Symposium on Computational Geometry*, pages 12–17, 1988.

A randomized insertion algorithm. Uses triangulation.

[17] George B. Dantzig, Alex Orden, and Philip Wolfe. The generalized simplex method for minimizing a linear form under liner inequality restraints. *Pacific Journal of Mathematics*, 5:183–195, 1955.

The canonical description of the simplex method using lexicographic pivoting.

[18] Lloyd L. Dines. Systems of linear inequalities. *Annals of Mathematics (2)*, 20:191–199, 1918–1919.

An elimination based method for solving systems of inequalities.

[19] Martin E. Dyer. The complexity of vertex enumeration methods. *Mathematics of Operations Research*, 8(3):381–402, 1983.

Uses a result of Klee [99] to show that insertion algorithms that insert the constraints in the order given are exponential in the worst case. Also gives a pivoting algorithm based on breadth first search

[20] Martin E. Dyer and L.G. Proll. An algorithm for determining all extreme points of a convex polytope. *Mathematical Programming*, 12:81–96, 1977.

> A pivoting algorithm based on breadth first search of the pivot graph. Contains about 10 references to applications of vertex enumeration.

[21] Jack Edmonds. Decomposition using Minkowski. Abstracts of the 14th International Symposium on Mathematical Programming, Amsterdam, 1991.

> Gives a recursive procedure that given $P = \{ x \mid Ax \leq b \}$, and a point z interior to P, computes $d + 1$ or fewer extreme points of P that contain z in their convex hull. This procedure can be implemented to run in time $O(md^3)$, where m is the row size of A.

[22] Jean Baptiste Joseph Fourier. *Histoire de l'Academie Royale des Sciences de l'Institute de France*, volume 7, pages xlvii–lv. l'Imprimerie de Du Pont, pere et fils, Paris, 1824. Reprinted in [23]. English translation in [28].

> Sketches the (geometric) simplex method and describes the method for finding all extremal solutions of a system of linear inequalities now known as Fourier-Motzkin Elimination

[23] Jean Baptiste Joseph Fourier. *Œuvres de Fourier*. Gauthier-Villars et Fils, Paris, 1890.

[24] Komei Fukuda, Thomas M. Liebling, and Franęois Margot. Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron. *Computational Geometry: Theory and Applications*, 8:1–12, 1997.

> Contains an algorithm to compute the $\bar{f}$ faces of a polytope with m facets and n vertices in time $O(\bar{f} m \, L(m, n))$ where $L(m, n)$ is the amount of time necessary to solve an m by n linear program.

[25] Komei Fukuda and Alain Prodon. Double description method revisited. In *Combinatorics and computer science (Brest, 1995)*, volume 1120 of *Lecture Notes in Comput. Sci.*, pages 91–111. Springer-Verlag, Berlin, 1996.

[26] Komei Fukuda and Vera Rosta. Combinatorial face enumeration in convex polytopes. *Comput. Geom. Theory Appl.*, 4:191–198, 1994.

> Gives a combinatorial algorithm to enumerate the $\bar{f}$ faces of a d-polytope with m facets and n vertices in time $O(\bar{f}^2 \min(m, n))$. Requires vertex/facet incidence information as input.

[27] Peter Gritzmann and Victor Klee. On the complexity of some basic problems in computational convexity: II. Volume and mixed volumes. In T. Bistztriczky, P. McMullen, and R. Ivic Weis, editors, *Polytopes: abstract, convex and computational (Scarborough, ON, 1993)*, NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci., 440, pages 373–466. Kluwer Acad. Publ., Dordrecht, 1994.

> Gives a vertex enumeration algorithm polynomial for simplicial polytopes based on intersecting the constraints with each bounding hyperplane, using linear programming to remove redundant constraints, and enumerating the vertices of the simplicial facet by brute force.

[28] D.A. Kohler. Translation of a report by Fourier on his work on linear inequalities. *Opsearch*, 10:38–42, 1973. Translation of [22].

[29] Miroslav Mañas and Josef Nedoma. Finding all vertices of a convex polyhedron. *Numerische Mathematik*, 12:226–229, 1968.

> Pivoting algorithm. Does breadth first search on the basis graph.

[30] K. Mehlhorn. *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry*, volume 3 of *EATCS Monographs*

on Theoretical Computer Science. Springer-Verlag, Heidelberg, West Germany, 1984.

[31] Theodore S. Motzkin. *Beiträge zur Theorie der linearen Ungleichungen*. Doktorarbeit, Universität Basel, 1933. Published in Jerusalem in 1936, English translation in [32, 33].

> Sketches the double description method (i.e. successive intersection of halfspaces) in a proof of a generalization of a finite basis theorem of Minkowski (¶ 30). Gives a more complete description of the elimination procedure introduced by Fourier [22] (¶ 87).

[32] Theodore S. Motzkin. *Contributions to the theory of linear inequalites*. Number 22 in RAND Corporaton Translations. Rand Corporation, 1952. English translation of [31]. Reprinted in [33].

[33] Theodore S. Motzkin. Contributions to the theory of linear inequalites. In David Cantor, Basil Gordon, and Bruce Rothschild, editors, *Theodore S. Motzkin: Selected Papers*. Birkhauser, 1983.

[34] Theodore S. Motzkin, H. Raiffa, G.L. Thompson, and R. M. Thrall. The double description method. In H.W. Kuhn and A.W. Tucker, editors, *Contributions to the Theory of Games II*, volume 8 of *Annals of Math. Studies*, pages 51–73. Princeton University Press, 1953.

> The first widely available description of an insertion or incremental algorithm for vertex and extreme ray enumeration. Also describes application to two player games

[35] Günter Rote. Degenerate convex hulls in high dimensions without extra storage. In *Proceedings of the 8th ACM Symposium on Computational Geometry*, pages 26–32, 1992.

Reverse search on facets rather than bases. Requires traversal of the face lattice of each facet to determine ridges.

[36] Raimund Seidel. A convex hull algorithm optimal for point sets in even dimensions. Master's thesis, University of British Columbia, Dept. of Computer Science, 1981. Report 81-14.

Achieves a time bound of $O(n^{\lfloor d/2 \rfloor})$ in fixed even dimension d.

[37] Raimund Seidel. *Output-size sensitive algorithms for constructive problems in computational geometry.* Ph.D. thesis, Dept. Comput. Sci., Cornell Univ., Ithaca, NY, 1986. Technical Report TR 86-784.

Uses shellings to get output sensitive time for fixed dimension and points in general position

[38] Ruth Wycliffe Stokes. A geometric theory of solution of linear inequalities. *Transactions of the Americal Mathematical Society*, 33:782–805, 1931.

Gives a geometric dual to Minkowski's algorithm [93].

[39] Garret Swart. Finding the convex hull facet by facet. *Journal of Algorithms*, pages 17–48, 1985.

Gives an analysis of Chand and Kapur's dual pivoting algorithm [11]. Contains a modification of [11] that computes the face lattice in time polynomial in the input and output size.

[40] T. Yamada, J. Yoruzuya, and S. Kataoka. Enumerating extreme points of a highly degenerate polytope. *Computers and Operations Research*, 21(4):397–410, 1994.

A recursive pivoting algorithm for vertex enumeration that amounts to enumerating the (dual) abbreviated face lattice (see [39]). The paper also details an optimization for the case when a variable occurs in exactly one constraint.

II Implementations

See also: [2, 3]

[41] D. Alevras, G. Cramer, and M. Padberg. *DODEAL*. ftp://elib.zib-berlin.de/pub/mathprog/polyth/dda

Implementation of the double description method, based on [9].

[42] David Avis. A C implementation of the reverse search vertex enumeration algorithm. Technical Report RIMS Kokyuroku 872, Kyoto University, May 1994. (Revised version of: Technical Report SOCS-92.12, McGill University, School of Computer Science). ftp://ftp-cgrl.cs.mcgill.ca/pub/polytope/soft/src/rs/lrs.c.Z

[43] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The Quickhull algorithm for convex hull. Technical Report GCG53, Geometry Center, Univ. of Minnesota, July 1993. ftp://geom.umn.edu/pub/qhull.tar.Z

A variant of the beneath and beyond method that inserts points based how far outside the current polytope they are.

[44] Thomas Christof and Andreas Loebel. porta. Version 1.3.1, March 1997. http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/PORTA/readme.html

Fourier-Motzkin elimination, with exact arithmetic.

[45] Kenneth L. Clarkson. hull 1.0. AT&T Bell Labs. http://www.cs.att.com/netlib/voronoi/hull.html

Incremental method working in exact arithmetic “when possible”.

[46] Ionnis Z. Emiris, John F. Canny, and Raimund Seidel. Efficient perturbations for handling geometric degeneracies. manuscript. `ftp://robotics.eecs.Berkeley.edu/pub/ConvexHull`

Describes an implementation of the beneath and beyond (incremental) method using perturbation. The implementation is available from the same place.

[47] Komei Fukuda. cdd+ Reference manual, version 0.74. ETHZ, Zurich, Switzerland. `ftp://ifor13.ethz.ch/pub/fukuda/cdd`

C++ implementation of the double description method [34] using exact rational or floating point arithmetic. ANSI C version (floating point only) available at the same location.

[48] Michael Müller and Joachim Ziegler. An implementation of a convex hull algorithm. manuscript, January 1995. `ftp://ftp.mpi-sb.mpg.de/pub/LEDA/articles/chull.ps.Z`

An implemention of an incremental algorithm, using LEDA [49]. Uses triangulation.

[49] Stefan Näher and Christian Urhig. *The LEDA User Manual Version R 3.2.* `http://www.mpi-sb.mpg.de/LEDA/leda.html`

C++ library for geometric operations.

[50] Doran K. Wilde. A library for doing polyhederal applications. Technical Report 785, IRISA, Campus Universitaire de Beaulieu – 35042 Rennes CEDEX France, 1993. `ftp://ftp.irisa.fr/local/API`

Describes an implementation of the double description method [34] and its applications to parallelizing compilers and VLSI synthesis.

III Applications and Selected Polyhedra

See also: [34, 50]

[51] Franz Aurenhammer. Power diagrams: properties, algorithms, and applications. *SIAM Journal on Computing*, 16(1):79–96, 1987.

[52] David Avis and Antoine Deza. Solitaire cones. manuscript in preparation, October 1995.

[53] David Avis and Michel Deza. The cut cone, l_1 embeddability, complexity and multicommodity flows. *Networks*, 21:596–617, 1991.

[54] Braams. Polytopes from quantum chemistry. private communication, 1995.

[55] K.Q. Brown. Voronoi diagrams from convex hulls. *Information Processing Letters*, 9:223–228, 1979.

[56] G. Ceder, G.D. Garbulsky, David Avis, and Komei Fukuda. Ground states of a ternary lattice model with nearest and next-nearest neighbor interactions. *Physical Review B*, 49:1–7, 1994.

Uses a linear optimization model with unknown objective function to model the ground states (crystal structure) of alloys.

[57] Antoine Deza, M. Deza, and Komei Fukuda. On skeletons, diameters and volumes of metric polyhedra. In *Combinatorics and computer science (Brest, 1995)*, volume 1120 of *Lecture Notes in Computer Science*, pages 112–128. Springer-Verlag, Berlin, 1996.

[58] Antoine Deza and Michel Deza. The ridge graph of the metric polytope and some relatives. In T. Bistztriczky, P. McMullen, and R. Ivic Weis, editors, *Polytopes: Abstract, Convex, and Computational*, pages 359–372. Kluwer Academic, 1994. Proceedings of the NATO Advanced Study Institute on Polytopes: Abstract, Convex, Computational, Scarborough, Ontario, Canada, August 20-September 3, 1993.

Proves, among other things, that the ridge graph of the metric polytope has diameter 2.

[59] Michel Deza. Matrices de formes quadratiques non négatives pour des arguments binaires. *Comptes rendues de l'Académie des Sciences de Paris*, 277:873–875, 1973.

On the cut polytope.

[60] Michel Deza and M. Laurent. Facets for the cut cone I. *Mathematical Programming*, 56(2):121–160, 1992.

[61] R. Euler and H. Le Verge. Complete linear descriptions of small asymetric travelling salesman polytopes. *Discrete Applied Mathematics*, 62:193–208, 1995.

Uses a version of Chernikova's algorithm [14] to compute the facet descriptions of the asymetric travelling salesman polytopes on 5 and 6 nodes. The 6 node polytope has 319015 facet defining inequalities and 11 equalities.

[62] V. P. Grishukin. Computing extreme rays of the metric cone for seven points. *European Journal of Combinatorics*, 13:153–165, 1992.

[63] James P. Ignizio and Tom M. Cavalier. *Linear Programming*. In *Prentice Hall International Series in Industrial and Systems Engineering* [95], 1994. Part III.

A general reference on multiobjective optimization.

[64] M. Lomonosov. Combinatorial approaches to multiflow problems. *Discrete Applied Mathematics*, 11:1–93, 1985.

Application of the metric polytope to multicommodity flows.

[65] Marek Teichmann. Probabilistic algorithms for efficient grasping and fixturing. Manuscript. Courant Institute, N.Y.U., November 1995.

Using convex hulls to compute the maximum force resistable by a set of fingers grasping an object.

IV Triangulation, Perturbation, and Degeneracy

[66] Margaret M. Bayer. Equidecomposable and weakly neighborly polytopes. *Israel Journal of Mathematics*, 81:301–320, 1993.

A polytope is called *equidecomposable* if every triangulation has the same f-vector. A polytope is called *weakly neighbourly* if every $k+1$ vertices are contained in a face of dimension at most $2k$, $0 \leq k \leq d$. One result of this paper is that weakly neighbourly polytopes are equidecomposable

[67] Louis J. Billera, R. Cushman, and J.A. Sanders. The Stanley decomposition of the harmonic oscillator. *Proceedings of the Netherlands Mathematical Society*, 50(4):375–393, December 1988.

Contains a proof that every triangulation of $T_s \times T_t$ requires exactly $\binom{s+t}{t}$ simplicies.

[68] Jesús de Loera. *Triangulations of Polytopes and Computational Algebra.* PhD thesis, Cornell, 1995.

[69] I.M. Gel'fand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants, resultants, and multidimensional determinants.* Mathematics: theory & applications. Birkhauser, Boston, 1994.

Also contains a proof that every triangulation of $T_s \times T_t$ requires exactly $\binom{s+t}{t}$ simplicies.

[70] M. Haiman. A simple and relatively efficient triangulation of the n-cube. *Discrete and Compututational Geometry*, 6(4):287–289, 1991.

Includes a brief description of the volume based lower bound of $\Omega(\sqrt{n}!)$ lower bound for triangulations of a cube. Contains a volume based proof for the fact that every triangulation of $T_k \times T_l$ requires exactly $\binom{k+l}{l}$ maximal simplices.

[71] Robert B. Hughes and Micheal R. Anderson. Simplexity of the cube. *Discrete Mathematics*, 158:99–150, 1996.

[72] James P. Ignizio and Tom M. Cavalier. *Linear Programming*, pages 118–122. In *Prentice Hall International Series in Industrial and Systems Engineering* [95], 1994.

Description of lexicographic ratio testing

[73] Carl W. Lee. Regular triangulations of convex polytopes. In *The Victor Klee Festschrift*, volume 4 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 443–456. American Mathematical Society, 1991.

[74] Nimrod Megiddo and R. Chandrasekaran. On the ϵ-perturbation method for avoiding degeneracy. *Operations Research Letters*, 8:305–308, 1989.

[75] Raimund Seidel. The nature and meaning of perturbations in geometric computing. To appear Discrete and Compututational Geometry, 1996.

[76] C-K Yap. Symbolic treatment of geometric dependancies. *J. Symbolic Computation*, 10:349–370, 1990.

V Related Theory

See also: [2, 3, 4, 7, 12, 21, 23, 28, 72]

[77] A. Altshuler and M.A. Perles. Quotient polytopes of cyclic polytopes. Part I: Structure and characterization. *Israel Journal of Mathematics*, 36(2):97–125, 1980.

> A quotient polytope is a polytope arrived at by taking repeated vertex figures. The authors characterize the quotients of cyclic polytopes, and show that every simplicial polytope with $d + 3$ vertices can arise as a quotient of a cyclic polytope

[78] David W. Barnette. The minimum number of vertices of a simple polytope. *Israel Journal of Mathematics*, 8:304–308, 1971.

> The lower bound theorem. A simple d-polytope with m facets has at least $\beta(d, m) = (m - d)(d - 1) + 2$ vertices. See also [79]

[79] David W. Barnette. A proof of the lower bound conjecture for convex polytopes. *Pacific Journal of Mathematics*, 46:349–354, 1973.

[80] Margaret M. Bayer and Carl W. Lee. Combinatorial aspects of convex polytopes. In P.M. Gruber and J.M. Wills, editors, *Handbook of Convex Geometry*, volume A, chapter 2.3, pages 485–534. North Holland, New York, NY, 1993.

> A survey of results up to about 1993. Includes a large section on results using algebraic geometry.

[81] Russell V. Benson. *Euclidean Geometry and Convexity*. McGraw-Hill, 1966.

[82] Louis J. Billera and Carl W. Lee. A proof of the sufficency of McMullen's conditions for f-vectors of simplicial convex polytopes. *Journal of Combinatorial Theory A*, 31:237–255, 1981.

[83] Gerd Blind and Roswitha Blind. Convex polytopes without triangular faces. *Israel Journal of Mathematics*, 71:129–134, 1990.

The authors show that for a d-polytope P without a triangular 2-face, $f_k(P) \geq f_k(H_d)$ where H_d is the d-dimensional hypercube.

[84] Arno Brøndsted. *Introduction to Convex Polytopes*. Springer-Verlag, 1981.

Introductory text. Proves the upper and lower bound theorems starting from affine and convex combinations.

[85] Vašek Chvátal. *Linear Programming*. W. H. Freeman, New York, NY, 1983.

Good reference and text on the simplex method. Includes duality, complementary slackness, the revised simplex method. Develops everything from elementary principles. Contains a pivoting based algorithm that does breadth first search of the basis graph (Chapter 18).

[86] George.B. Dantzig. Maximizing a linear function of variables subject to linear inequalities. In Tj. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347. Wiley, New York, 1951.

[87] Antoine Deza. On a lower bound for general convex polytopes. *Mathematica Japonicae*, 40(2):371–380, 1994.

[88] Antoine Deza and Komei Fukuda. McMullen's conditions and some lower bounds for general convex polytopes. *Geometriae Dedicata*, 53:165–173, 1994.

[89] Herbert Edelsbrunner. *Algorithms in Combinatorial Geometry*. Number 10 in EATCS Monographs on Th. Comp. Sci. Springer-Verlag, 1987.

[90] David Gale. Neighborly and cyclic polytopes. In Victor Klee, editor, *Proceedings of the 7th Symposium on Pure Mathematics*, 1961.

[91] A. Ghouila-Houri. Caractérisation des matrices totalment unimodulairs. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences (Paris)*, 254:1192–1194, 1962.

[92] Branko Grünbaum. *Convex Polytopes*, volume XVI of *Pure and Applied Mathematics*. Interscience, London, 1967.

[93] Harris Hancock. *Development of the Minkowski Geometry of Numbers*, chapter 33, pages 100–106. Macmillan, New York, NY, 1939. English translation/paraphrase of [105].

[94] M. Henk, J. Richter-Gebert, and G. M. Ziegler. Basic properties of convex polytopes. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 13, pages 243–270. CRC Press LLC, 1997.

[95] James P. Ignizio and Tom M. Cavalier. *Linear Programming*. Prentice Hall International Series in Industrial and Systems Engineering. Prentice Hall, 1994.

[96] Gil Kalai. The number of faces of centrally symmetric convex polytopes. *Graphs and Combinatorics*, 5:389–391, 1989. Research Problem.

[97] L.V. Kantorovich. My journey in science. *Russian Math. Surveys*, 42:233–270, 1987.

[98] T.P. Kirkman. On the enumeration of x-hedra having triedal summits and an $(x-1)$-gonal base. *Philos. Trans. Royal Soc. London Ser. A*, 146:399–411, 1856.

> Characterizes those 3-polytopes containing a facet that is adjacent to every other facet

[99] Victor Klee. Polytope pairs and their relationship to linear programming. *Acta Math.*, 133:1–25, 1974.

Shows that there exists pairs (P, F) where P is a convex polytope and F is a facet of P such that numbers of vertices close to the extremes of [78, 102] are independently achieved for P and $P \setminus F$. Furthmore, these bounds are achieved for (P, F) where F intersects every other facet of P (a "Kirkman pair", see [98]).

[100] Victor Klee and G.J. Minty. How good is the simplex method? In O. Shisha, editor, *Inequalities-III*, pages 159–175. Academic Press, 1972.

Gave a class of skewed d-cubes for which the simplex method using the largest coefficent method visits all vertices

[101] Ulrich H. Kortenkamp, Jürgen Richter-Gebert, Aravamuthan Sarangarajan, and Günter M. Ziegler. Extremal properties of 0/1-polytopes. *Discrete and Compututational Geometry*, 4(16), 1997.

[102] Peter McMullen. The maximal number of faces of a convex polytope. *Mathematika*, 17:179–184, 1970.

The upper bound theorem: a convex d-polytope with m facets has at most

$$\gamma(d, m) = \binom{m - \lfloor (d+1)/2 \rfloor}{m - d} + \binom{m - \lfloor (d+2)/2 \rfloor}{m - d}$$

vertices.

[103] Peter McMullen. The number of faces of simplicial polytopes. *Israel Journal of Mathematics*, 9:559–570, 1971.

[104] Peter McMullen and G. C. Shephard. *Convex polytopes and the Upper Bound Conjecture*. Cambridge University Press, Cambridge, 1971.

[105] Hermann Minkowski. *Geometrie der Zahlen*, chapter 19, pages 39–45. Teubner, Leipzig, 1910. English translation/paraphrase in [93].

The following theorems are proved (all systems under consideration are homogeneous):

1. Every basic solution of a system of inequalities is extreme.
2. If there is a solution of a system of inequalities, there is a basic solution.
3. Every solution is a positive combination of extreme solutions.

Algorithms are provided to compute the redundant constraints in a system, and to compute the extreme solutions of a system by considering each linearly independent set of constraints.

[106] Theodore S. Motzkin. Comonotone curves and polytopes. *Bulletin of the American Mathematical Society*, 63(35), 1957. Abstract 111.

Conjectures the Upper Bound Theorem (see [102])

[107] James A. Murtha and Earl R. Willard. *Linear Algebra and Geometry*. Holt Rinehart and Winston, New York, 1969.

[108] Nemhauser and Wolsey. *Integer Programming*. Wiley-Interscience, 1988.

[109] Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience Series in Discrete Mathematics. Wily-Interscience, New York, 1986.

[110] Richard P. Stanley. The number faces of a simplicial convex polytope. *Advances in Mathematics*, 35:236–238, 1980.

[111] L. Fejes Tóth. *Regular Figures*. McMillan/Pergamon, 1964.

[112] Günter M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer-Verlag, 1994.

[113] Victor Klee. Extreme points of convex sets without completeness of the scalar field. *Mathematika*, 11:59–63, 1964.

[114] K.G. Murty. The gravitational method for linear programming. *Opsearch*, 23:206–214, 1986.