How Good are Convex Hull Algorithms?

David Avis*

School of Computer Science
McGill University
Montreal, QC, CANADA, H3A 2A7

David Bremner*

School of Computer Science
McGill University
Montreal, QC, CANADA, H3A 2A7

Raimund Seidel†

Computer Science Division
University of California Berkeley
Berkeley CA 94720

Fachberich Informatik
Universität des Saarlandes
D-66041 Saarbrücken, GERMANY

February 1996

Abstract

A *convex polytope* P can be specified in two ways: as the convex hull of the vertex set $\mathcal{V}$ of P, or as the intersection of the set $\mathcal{H}$ of its facet-inducing halfspaces. The *vertex enumeration problem* is to compute $\mathcal{V}$ from $\mathcal{H}$. The *facet enumeration problem* it to compute $\mathcal{H}$ from $\mathcal{V}$. These two problems are essentially equivalent under point/hyperplane duality. They are among the central computational problems in the theory of polytopes. It is open whether they can be solved in time polynomial in $|\mathcal{H}| + |\mathcal{V}|$.

In this paper we consider the main known classes of algorithms for solving these problems. We argue that they all have at least one of two weaknesses: inability to deal well with "degeneracies," or, inability to control the sizes of intermediate results. We then introduce families of polytopes that exercise those weaknesses. Roughly speaking, *fat-lattice* or *intricate* polytopes cause algorithms with bad degeneracy handling to perform badly; *dwarfed* polytopes cause algorithms with bad intermediate size control to perform badly.

We also present computational experience with trying to solve these problem on these hard polytopes, using various implementations of the main algorithms.

1 Introduction

A d-dimensional *convex polyhedron* is the intersection of a finite number m of non-redundant halfspaces $\mathcal{H} = \{ H_1, H_2, \ldots, H_m \}$ of $\mathbb{R}^d$. A bounded convex polyhedron is called a *polytope*. A classic theorem from convexity states that every polytope P can be expressed as the convex hull of its n extreme points (or vertices) $\mathcal{V}$. These descriptions of P will be referred to as the *halfspace* and

*Supported by NSERC Canada, and FCAR Québec. Email: `[avis,bremner]@cs.mcgill.ca`

†Supported by NSF Presidential Young Investigator award CCR-9058440. Email: `seidel@cs.uni-sb.de`

vertex descriptions, respectively. The size of a polytope, denoted $size(P) = (m+n)d$, is the space required to store both descriptions of a polytope. There are three closely related computational problems concerning the two descriptions of a polytope:

- The *vertex enumeration problem* asks to compute $\mathcal{V}$ from $\mathcal{H}$.
- The *facet enumeration problem* asks to compute $\mathcal{H}$ from $\mathcal{V}$.
- The *polytope verification problem* asks to decide whether a given vertex description $\mathcal{V}$ and halfspace description $\mathcal{H}$ define the same polytope.

If redundant elements are allowed as inputs to the polytope verification problem, then all three problems are polynomially equivalent. The first two problems are equivalent under point/hyperplane duality. Either of the first two can be trivially used to solve the third. To solve e.g. vertex enumeration using polytope verification, verify that all vertices have been found for each facet of P. If not, recurse on that facet. In at most $d|\mathcal{H}|$ polytope verification calls, the algorithm finds a new vertex.

This paper will not be concerned with the polytope verification problem per se, but with vertex enumeration and facet enumeration. We already stated that these two problems are essentially equivalent under point/hyperplane duality. Thus it would suffice if we restricted the discussion to just one of those two. However, some aspects and phenomena are more easily described in the context of facet enumeration, others more easily in the context of vertex enumeration. For that reason we will feel free to switch back and forth between those two problems, with the understanding that all examples and results stated for vertex enumeration also hold, appropriately dualized, for the facet enumeration problem, and vice versa. However, most of the ensuing discussions will be in terms of vertex enumeration.[1]

We are interested in the computational difficulty of vertex enumeration. When measuring the efficiency of a vertex enumeration algorithm it is important to take into account the vast possible variation in output size. By the upper and lower bound theorems of McMullen [29] and Barnette [6] a polytope P specified as the intersection of m halfspaces in $\mathbb{R}^d$ can have as few as

$$\beta(d,m) = (m-d)(d-1)+2$$

vertices (in the non-simple case even fewer) and as many as

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

This large possible range suggests that the performance of vertex enumeration algorithms be measured not only in terms of input size dm but also in terms of output size dn, where n is the number of vertices produced.

An outstanding question from both a theoretical and a practical point of view is whether vertex enumeration can be solved in time polynomial in input size dm and output size dn, or equivalently, polynomial in $size(P) = d(m+n)$. One of the central purposes of this paper is to show that all known main types of vertex enumeration algorithms actually do have superpolynomial worst case

[1] Actually, in some ways it would be more elegant to apply homogenization and frame our discussion in terms of extreme ray and facet enumeration for polyhedral cones. We will refrain from doing so, mainly since the affine setting provides better intuition. For the most part we will ignore the issue of unboundedness and extreme rays, although it has some aspects that are interesting in their own right.

running time. This is done by providing explicit example families of polytopes for which the various algorithms perform poorly. We also corroborate those findings by computational experiments.

At this point a comparison to the situation in Linear Programming may be called for. Obviously Linear Programming and vertex enumeration are closely related. The title of our paper was inspired by the famous paper of Klee and Minty [26] where they showed that the simplex algorithm with a certain natural pivoting rule can have superpolynomial running time. Our goal is more ambitious in that we want to show superpolynomial running time not just for one specific algorithm but for all known main classes of vertex enumeration algorithms. At the same time our findings are more devastating, since in contrast to the simplex algorithm, which on problems arising in practice essentially never exhibits its possible worst case behaviour, our experience suggests that vertex enumeration algorithms do exhibit their superpolynomial worst case behaviour on problems arising in practice (see e.g. [9, 21]). Fortunately vertex enumeration as a problem arises much less frequently than linear programming. Still, we want to stress that contrary to prevailing opinions in the computational geometry community, the current situation with respect to vertex and facet enumeration is unsatisfying both in the practical and in the theoretical sense. In Linear Programming the current state of affairs appears to be somewhat satisfactory for practical purposes (and of course from a theoretical point of view there are polynomial algorithms for Linear Programming in the bit model of computation).

1.1 The main algorithms

Geometrically the main vertex enumeration algorithms can all be viewed as not just generating all vertices of a polytope P but actually the 1-skeleton of P, i.e. the graph formed by P's vertices and edges. Dually, facet enumeration algorithms can be viewed as generating the "facet graph," i.e. a graph whose nodes are the facets of the polytope and with two facets adjacent iff they share a common ridge.

In essence there are only two main classes of algorithms for producing these graphs: *graph traversal* algorithms and *incremental* algorithms.

The *graph traversal* algorithms first find some node of the graph in question and then attempt to identify all nodes and edges of the graph by traversing it in some fashion. In the case of vertex enumeration each vertex v of a d-polytope P can be identified by a *basis,* i.e. d facets that contain v and whose spanning hyperplanes are affinely independent. Two vertices of P are connected by an edge of P if they have bases that differ in exactly one member. Going from a vertex to an adjacent one during the graph traversal amounts to changing this one member of the basis. This operation is known as *pivoting* in the simplex algorithm for linear programming. For this reason graph traversal algorithms for vertex enumeration are also known as *pivoting algorithms*. In the context of facet enumeration going from one facet to a neighboring can be viewed as rotating a supporting hyperplane about the common ridge. In analogy to a 3-dimensional physical realization this operation is therefore known as a *gift-wrapping step*.

Representatives of this class of graph traversal algorithms are the *gift wrapping* algorithm of Chand and Kapur [10], Seidel's algorithm [35] and the *reverse search* algorithm of Avis and Fukuda [4].

Incremental algorithms for the vertex enumeration problem compute the vertex description by intersecting the defining halfspaces sequentially. An initial simplex is constructed from a subset of $d+1$ halfspaces and its vertices and 1-skeleton are computed. Additional halfspaces are introduced sequentially and the vertex description and 1-skeleton are updated at each stage. Essentially such an update amounts to identifying and removing all vertices that are not contained in the new halfspace, introducing new vertices for all intersections between edges and the bounding hyperplane of the new halfspace, and generating the new edges between these new vertices. Although the first explicit description of such an algorithm, now widely known as the *double description method*, appeared in the pioneering 1953 paper of Motzkin et al. [30], this paper seems to have been overlooked by the Computational Geometry community. Many of the same ideas were rediscovered and refined in the *beneath and beyond method* of Seidel [34] (in the facet enumeration setting), the *randomized algorithm* of Clarkson and Shor [14] and the *derandomized algorithm* of Chazelle [11]. Finally we should mention that so-called Fourier-Motzkin elimination can be viewed just as a dual formulation of the double description method and thus falls in the class of incremental algorithms.

All incremental algorithms can employ different insertion orders. One can distinguish between *static* insertion orders, which are determined from the inputs before the actual incremental algorithm is started, and *dynamic* insertion orders, where the next halfspace (vertex) to be considered is a function of the current polytope and the remaining halfspaces (vertices). Typical static orderings are **minindex** (process in order as the input happens to be), **lexicographic** (process halfspaces in lexicographic order of their suitably normalized coefficient vectors), and **random**. Typical dynamic orderings are **maxcutoff** and **mincutoff** (as next halfspace choose the one whose complement contains the most or least vertices of the current polytope). As we shall see, the same algorithm applied to the same input can have vastly different running times when different insertion orders are used. Thus choosing a good insertion order is crucial. As a manifestation of this consider the incremental algorithms of Seidel [34] and of Chazelle [11]. If performance is measured only in terms of input size and the dimension d is kept fixed, then these algorithms are asymptotically worst case optimal (for even d in case of [34], and for general d in case of [11]). Seidel's algorithm relies crucially on the use of a lexicographic insertion order. The biggest part of Chazelle's algorithms is the determination of a very sophisticated dynamic insertion order.

One of the main obstacles to achieving polynomiality in $size(P)$ is *degeneracy*. In the case of vertex enumeration this means more than d facets contain a vertex and hence vertices do not have a unique basis; the polytope is not simple and vertices can be incident to more than d edges. For pivoting algorithms this creates problems since a non-simple polytope may have many more bases than vertices, and a naive pivoting algorithm visits every basis. For facet enumeration, degeneracy dualizes to having more than d vertices contained in a facet. Many incremental facet enumeration algorithms (e.g. [34, 11, 33]) require that the intermediate polytopes be simplicial (or equivalently triangulated) in order to efficiently (i.e. not just in polynomial time) find the new ridges.

There are two general methods to deal with degeneracy. They apply equally well to pivoting and to insertion algorithms. The first method is to generalize the algorithm so that it generates the entire face lattice of the polytope. We call such algorithms *lattice producing*. Examples are the gift wrapping algorithm of Chand and Kapur [10], Swart [36], Seidel [34], [17], [35], and Rote [32].

The second method employs perturbations in order to simulate non-degeneracy. Since in the case of facet enumeration perturbing the input vertices results in triangulating the facets we call such algorithms *triangulation producing*. All algorithms that work correctly in the non-degenerate case are amenable to the perturbation method. One advantage of triangulation producing algorithms is that they can easily be adapted to compute the volume of a polytope. Using the reverse-search technique of Avis and Fukuda [4] one can compute the volume of a polytope using space linear in the input size.

The only exception to the classification "lattice producing" versus "triangulation producing" appears to be the double description method of Motzkin et al. [30]. This incremental method does not maintain an explicit description of the 1-skeleton, but maintains the incidence information between facets and vertices, from which adjacency information between vertices can be recovered.

This method appears to deal surprisingly well with degenerate inputs.

In summary, we have classified algorithms along two lines. We have graph traversal algorithms versus incremental algorithms (with various insertion orders). And we have lattice producing algorithms versus triangulation producing algorithms, plus the special case of the double description method (which happens to be an incremental algorithms).

1.2 The example polytope families

We consider three classes of polytopes: *fat-lattice* polytopes, *intricate* polytopes, and *dwarfed* polytopes. Loosely speaking, a fat-lattice polytope P is one for which the total number of its faces (of all dimensions) is "much larger" than $size(P)$. An intricate polytope P is one for which the number of simplices required to triangulate all of its facets is "much larger" than $size(P)$. Finally, a dwarfed polytope P is one that can be represented as $Q \cap H$, where P has one more facet than Q but $size(Q)$ is "much larger" than $size(P)$.

We now make precise what we mean by the term "much larger". Given some function A of $m = |\mathcal{H}|$, $n = |\mathcal{V}|$, and the dimension d, we say that A is "much larger" than $size(P)$ if it is superpolynomial in $size(P)$ i.e. if A is not bounded by any polynomial in m, n, and d, and hence not by any polynomial in $size(P)$.

For all three types of polytopes, fat-lattice, intricate, and dwarfed, we provide example families of two kinds, *uniparametric*, and *biparametric*. A uniparametric family $\mathcal{F}$ comprises d-polytopes P of arbitrarily large dimension d but with $size(P)$ a function of d. A biparametric family contains for infinitely many $d \in \mathbb{N}$ a subfamily $\mathcal{F}_d$ that comprises d-dimensional polytopes P with $size(P)$ arbitrarily large.

Note that if for the individual subfamilies $\mathcal{F}_d$ of a biparametric family $\mathcal{F}$ we have only the polynomial bounds $A = \Omega(size(P)^{c_d})$, then, considering all of $\mathcal{F}$, it is still the case that "A is much larger than $size(P)$," provided that c_d is arbitrarily large with d large enough. (Also note that the asymptotic expression $\Omega(size(P)^{c_d})$ makes sense, since $\mathcal{F}_d$ contains polytopes of arbitrarily large size.)

It will be easy to argue that lattice producing algorithms behave badly on fat-lattice polytopes, and triangulation producing algorithms behave badly on intricate polytopes. According to our classification this leaves only the double description method uncovered. However this is an incremental method and we will show that incremental algorithm behave badly on our dwarfed polytopes, if they employ one of the usual static insertion order or the dynamic mincutoff insertion order. If they use the dynamic maxcutoff insertion order, then they seem to behave badly on our intricate polytopes, although we can prove this only for one specific uniparametric family of intricate polytopes.

1.3 Previous work

There has been some work by Dyer [16] and by Swart [36], showing that incremental algorithms can potentially have very bad behaviour. They built on examples of Kirkman [24] and Klee [25]. However, they did not pursue a detailed study of the various possible insertion orders, and in particular ignored dynamic insertion orders. Moreover, our examples are more extreme and more easily specified as the ones in those references. There is a sizeable literature on polytope degeneracy, mostly in the context of linear programming. A "selected bibliography" by Gal [20] contains 123 references. Of particular interest from a vertex enumeration point of view are the following. Provan [31] showed that there are network polytopes that yield a superpolynomial number of perturbed vertices under lexicographic perturbation (and also provided a polynomial algorithm in our sense for enumerating the vertices of network polytopes). Armand [1] showed that (in the worst case), perturbing the $m > d$ facets adjacent to a single vertex can yield $\gamma(d,m) - (m-d+1)$ vertices. Here we consider classes of polytopes whose *best* perturbation is superpolynomial.

1.4 Organization of the paper

In Section 2 we briefly review basic properties of convex polytopes. In Section 3 we introduce our example classes of polytopes and derive their properties. In Section 4 we discuss how those example classes force bad behaviour on various algorithms. Our examples of hard polytopes apply to computational models that include most published vertex and facet enumeration algorithms. A number of these algorithms have been implemented by various people. But an implementation is rarely completely faithful to the algorithm from which it is derived. For this reason we present and discuss in Section 5 actual computational experience obtained by trying various implementations on our hard polytope classes of the previous sections.

2 Preliminaries

For a set $X \subset \mathbb{R}^d$ a *convex combination* of X is a point $\sum_{x \in X} \lambda_x x$, where $\lambda_x \geq 0$ for each x, and $\sum_{x \in X} \lambda_x = 1$. The *convex span* of X is the set of all convex combinations of X. A point $x \in X$ is called *extreme* iff it cannot be represented as a convex combination of $X \setminus \{x\}$.

A halfspace of $\mathbb{R}^d$ is a set representable as $\{x \in \mathbb{R}^d \mid \langle x, n \rangle \leq c\}$, where $c \in \mathbb{R}$ and n is a non-zero vector. Such a halfspace is bounded by the *hyperplane* $\{x \in \mathbb{R}^d \mid \langle x, n \rangle = c\}$. An element H of a set of halfspaces $\mathcal{H}$ is called *irredundant* if $\bigcap \mathcal{H} \neq \bigcap(\mathcal{H} \setminus \{h\})$. For a set $X \subset \mathbb{R}^d$ let $\mathcal{H}_X$ denote the set of halfspaces that contain X. The set $\bigcap \mathcal{H}_X$ is called the *convex hull* of X.

It is one of the basic results of convexity theory that the convex span and the convex hull of X are the same — usually denoted as $\mathrm{conv}X$. Moreover, if X is finite, then $\mathcal{H}_X$ has a finite number of irredundant halfspaces. Similarly, if $\mathcal{H}$ is a finite set of halfspaces with $\bigcap \mathcal{H}$ bounded, then $\bigcap \mathcal{H}$ has a finite number of extreme points.

These definitions and results naturally lead to the following four computational problems:

Extreme point: Given a finite set $X \subset \mathbb{R}^d$, determine its extreme points.

Irredundancy: Given a finite set $\mathcal{H}$ of halfspaces, determine its irredundant elements.

Facet enumeration: Given a finite set $X \subset \mathbb{R}^d$, determine the irredundant halfspaces of $\mathcal{H}_X$.

Vertex enumeration: Given a finite set $\mathcal{H}$ of halfspaces with $\bigcap \mathcal{H}$ bounded, determine the extreme points of $\bigcap \mathcal{H}$.

The first two problems, which are related by duality, can be solved using $|\mathcal{V}|$ and $|\mathcal{H}|$ linear programs respectively. Clarkson [13] has recently discovered a method to reduce the size of each linear program to A by d where A is the number of elements of the result set and d is the dimension. A more sophisticated method for constant dimension has been presented by Matoušek and Schwarzkopf [28].

The last two problems are the subject of this paper. Their names derive from the combinatorial structure of the boundary of *polyhedra*, i.e. sets representable as the intersection of a finite family of halfspaces. Bounded polyhedra are called *polytopes*. By the discussion above $\mathrm{conv}X$ is a polytope for every finite X. Let P be a polytope and H a halfspace with bounding hyperplane h so that $P \subset H$ and $P \cap h \neq \emptyset$. Hyperplane h is then called a *supporting hyperplane* of P, and $P \cap h$ is called a *face* of P. The empty set and P itself are also considered (improper) faces of P. The faces of P are closed under intersection. They are themselves polytopes and their faces are also faces of P. Thus the faces of P form a lattice, called the *face lattice* of P.

A face of dimension i is called an *i-face*. If P is a k-polytope (i.e. a polytope of dimension k), then its faces of dimension $k-1$, $k-2$, 1, and 0 are respectively called its *facets, ridges, edges*, and *vertices*. Note that the vertices of P are exactly the extreme points of P. We denote them by $\mathcal{V}(P)$. The facets of P correspond to the irredundant halfspaces of $\mathcal{H}_P$. They are denoted by $\mathcal{H}(P)$.

For a point $q \neq 0$ let H_q denote the halfspace $\{x \mid \langle x, q\rangle \leq 1\}$. Let P be a d-polytope in $\mathbb{R}^d$ that contains the origin in its interior. The *dual* of P is the set $P^* = \bigcap\{H_q \mid q \in P\}$. Since $p \in Q$ is extreme iff is H_p irredundant in $\{H_q \mid q \in Q\}$ and since $q \in H_p$ iff $p \in H_q$, we get that P^* is a polytope with H_v inducing a facet iff $v \in \mathcal{V}(P)$ and p a vertex iff $H_p \in \mathcal{H}(P)$. More generally, there is a 1-1 correspondence between i-faces of P and $(d-1-i)$-faces of its dual P^*. Because of this duality facet enumeration for P is equivalent to vertex enumeration of P^*.

A d-polytope is called *simple* iff each of its vertices is contained in exactly d facets. Dually, a d-polytope is called *simplicial* iff each of its facets contains exactly d vertices. In this case for $i < d$ each i-face is an *i-simplex*, which is the convex hull of $i+1$ affinely independent points.

For a polytope P let $f_i(P)$ denote the number of i-faces of P, and let $\overline{f}(P) = \sum_{i \geq 0} f_i(P)$. For a k-simplex S_k we have $f_i(S_k) = \binom{k+1}{i+1}$ and $\overline{f}(S_k) = 2^{k+1} - 1$. Of considerable interest is the question of the extreme possible values of $f_i(P)$ given that P is a d-polytope with m facets. McMullen [29] has shown that

$$f_i(P) \leq \gamma_i(d,m) = \sum_{0 \leq j \leq d} \binom{j}{i} \binom{m-1-\max\{j, d-j\}}{\min\{j, d-j\}}.$$

This bound is realized by the duals of *cyclic polytopes*. This is a polytope whose m vertices lie on a so-called *d-th order algebraic curve* c, which has the defining property that every hyperplane intersects c in at most d points. If $S = \{p_1, \ldots, p_m\}$ is a set of points ordered along such a curve c, and $J \subset M = \{1, \ldots, m\}$ is an index set with $|J| = d$, then $\{p_j \mid j \in J\}$ spans a facet of conv$S$ iff for all $k, \ell \in M \setminus J$ the number of indices in J that lie between k and ℓ is even. This is known as *Gale's evenness condition* and it follows directly from the fact that if a d-th order algebraic curve c intersects a hyperplane h in d points, then it must "cross" the hyperplane in those points. Gale's evenness condition implies that the face structure of a cyclic d-polytope with m vertices is independent of the curve and of the choice of points along the curve. Thus we generically denote a cyclic d-polytope with m vertices by $C_d(m)$. Examples of d-th order algebraic curves are the moment curve $c(t) = (t, t^2, \ldots, t^d)$, the binomial curve $c(t) = (\binom{t}{1}, \binom{t}{2}, \ldots, \binom{t}{d})$, for even $d = 2\delta$ Carathéodory's curve $c(t) = (\cos t, \sin t, \cos 2t, \sin 2t, \ldots, \cos \delta t, \sin \delta t)$, and — for the numerically courageous — $c(t) = (1/(t+1), 1/(t+2), \ldots, 1/(t+d))$.

For simple m-faceted d-polytopes P Barnette [6] has shown the lower bound

$$f_i(P) \geq \beta_i(d,m) = \begin{cases} (m-d)(d-1)+2 & \text{if } i = 0 \\ (m-d)\binom{d}{i+1} + \binom{d}{i} & \text{for } 1 \leq i \leq d\,. \end{cases}$$

This bound is realized by so-called truncation polytopes, which can be defined inductively as follows: a d-simplex is a truncation polytope with $d+1$ facets; an m-faceted truncation d-polytope is obtained by intersecting an $(m-1)$-faceted one with a halfspace that contains all but one of the vertices in its interior and does not contain the remaining vertex. Cyclic polytopes with m facets show that for non-simple P the face count may be considerably smaller than indicated by $\beta_i(d,m)$.

Note that for d constant and m growing we have $\beta_i(d,m) = \Theta(m)$, whereas $\gamma_i(d,m) = \Theta(m^{\lfloor d/2 \rfloor})$ for $i \leq \lceil d/2 \rceil$.

Of great importance in this paper is the product construction of polytopes. Let P be a polytope in $\mathbb{R}^a$ and Q be a polytope in $\mathbb{R}^b$. The *product* of P and Q is defined to be the set

$$P \times Q = \{(p,q) \mid p \in P,\ q \in Q\}.$$

We will view $P \times Q$, which is a subset of $\mathbb{R}^a \times \mathbb{R}^b$, as naturally embedded in $\mathbb{R}^{a+b}$. The following holds:

Lemma 1 *Let P be a k-polytope and Q be an ℓ-polytope.*

1. *$P \times Q$ is a $(k+\ell)$-polytope. If P and Q are simple, then so is $P \times Q$.*
2. *If F is an i-face of P and G is a j-face of Q, with $i, j \geq 0$, then $F \times G$ is an $(i+j)$-face of $P \times Q$. Moreover, this yields all non-empty faces of $P \times Q$.*
3. *The vertex count of $P \times Q$ is the product of the vertex counts of P and Q, whereas its facet count is the sum of the facet counts. Finally, its total face count is the product of the total face counts, i.e. $\overline{f}(P \times Q) = \overline{f}(P) \cdot \overline{f}(Q)$.*

Note that the coordinate representation of a vertex (p,q) of $P \times Q$ can be obtained by concatenating the coordinates of p and q. If $a_1x_1 + \cdots + a_kx_k \leq a_0$ defines a facet inducing halfspace of P and $b_1y_1 + \cdots b_\ell y_\ell \leq b_0$ defines a facet inducing halfspace of Q, then each defines a facet inducing halfspace of $P \times Q$, if $\mathbb{R}^{k+\ell}$ is considered coordinatized by $(x_1, \ldots, x_k, y_1, \ldots, y_\ell)$.

The product construction dualizes to forming the convex hull of P and Q, where P and Q are contained in orthogonal subspaces and each contains the origin in its interior.

Using the product construction it is easy to build up polytopes with many faces. If $P(m)$ is a convex polygon with m edges, then $P(m) \times P(m)$ is a 4-polytope with m^2 vertices and $2m$ facets. More generally, the δ-fold product $P(m) \times \cdots \times P(m)$ yields a 2δ-polytope with m^δ vertices and δm facets, which thus has a face complexity that for fixed dimension is asymptotically worst possible and the same as the one of dual cyclic polytopes.

More information on polytopes can be found in the books of Grünbaum [22], Brøndsted [8], and Ziegler [37].

3 Polytope Families

In this section we introduce three types of polytopes, fat-lattice, intricate, and dwarfed. For each type of polytope we give explicit infinite uniparametric and biparametric families. Recall that a uniparametric family comprises polytopes P of arbitrarily large dimension d but with $size(P)$ a function of d. A biparametric family is the union of infinitely many families, $\mathcal{F}_d$, each containing d-dimensional polytopes P with $size(P)$ arbitrarily large.

At the beginning of each of the three subsections we state theorems that introduce the various families of polytopes. On first reading the reader may want to skip their proofs (although they are not particularly difficult).

3.1 Fat-lattice polytopes

For us a family of *fat-lattice polytopes* consists of polytopes P with $\overline{f}(P)$, the total number of faces of P, much larger than $size(P)$. The existence of uniparametric fat-lattice families is well known (see e.g. [16]). It may come as a surprise that biparametric fat-lattice families also exist.

We start with some simple examples.

Theorem 1 *The set of simplices T_d forms a family of fat-lattice polytopes.*

Proof: The d-simplex T_d has $d+1$ vertices and $d+1$ facets and hence $size(T_d) = 2d(d+1)$. But $\overline{f}(T_d) = 2^{d+1} - 1 \geq 2^{\sqrt{size(T_d)/2}} - 1$, and thus $\overline{f}(T_d)$ is superpolynomial in $size(T_d)$. ■

In general any polytope family with size polynomial in d and at least one "large dimensional" simplex face (or, since duality preserves the size of the face lattice, a "small dimensional" face whose face figure is a simplex) will be fat-lattice.

Concerning biparametric families let us first consider as an example the family $CC_8(n) = C_4(n) \times C_4(n)$ of 8-polytopes formed by taking the product of two 4-dimensional cyclic polytopes with n vertices each. By Lemma 1 a polytope $CC_8(n)$ has n^2 vertices and $2\gamma(4,n) = n(n-3)$ facets, and thus we have $size(CC_8(n)) = \Theta(n^2)$. However, by the same lemma, for the total number of faces we have

$$\overline{f}(CC_8(n)) = \overline{f}(C_4(n))^2 = \Theta(n^4) = \Theta(size(CC_8(n))^2)\,.$$

Thus in dimension 8 we can achieve a quadratic relationship between size and total face count. By considering higher dimensions and using repeated products of cyclic polytopes any polynomial relationship can be achieved as we will show now:
Let d be such that $2d > 4$, let $a = \left\lceil\sqrt{d}\right\rceil$, $b = \lfloor d/a \rfloor$, and $c = d \bmod a$, and define

$$CC_{2d}(n) = \underbrace{C_{2a}(n) \times \cdots \times C_{2a}(n)}_{b \text{ times}} \times C_{2c}(n)\,.$$

Thus, roughly speaking, $CC_{2d}(n)$ is the $\sqrt{d}$-fold product of $(2\sqrt{d})$-dimensional cyclic polytopes with n vertices each. We refer to this family of polytopes simply as *products of cyclic polytopes.*

Theorem 2 *The products of cyclic polytopes $CC_{2d}(n)$ form a biparametric family of fat-lattice polytopes.*

Proof: Let $2d > 4$ be fixed and let $a = \left\lceil\sqrt{d}\right\rceil$, $b = \lfloor d/a \rfloor$, and $c = d \bmod a$, as defined before. Obviously for fixed $2d$ the family contains polytopes $CC_{2d}(n)$ of arbitrarily large size. By Lemma 1 the number of vertices of $CC_{2d}(n)$ is n^{b+1} (or n^b in the case $c = 0$) and the number of facets is $b \cdot \gamma(2a, n) + \gamma(2c, n) = \Theta(n^a)$, and thus $size(CC_{2d}(n)) = \Theta(n^a)$. For the total face count we have

$$\overline{f}(CC_{2d}(n)) = \overline{f}(C_{2a}(n))^b \cdot \overline{f}(C_{2c}(n)) = \Theta((n^a)^b \cdot n^c) = \Theta(n^d) = \Theta(size(CC_{2d}(n))^{d/a})\,.$$

Thus we have

$$\overline{f}(CC_{2d}(n)) = \Omega(size(CC_{2d}(n))^{c_{2d}})\,,$$

with $c_{2d} = d/\left\lceil\sqrt{d}\right\rceil$, which is arbitrarily large if d is large enough, as required. ■

Please note that the cyclic polytopes in this construction could be replaced by any other polytope class with similar complexity, as for instance dual products of polygons. By using integral points on parabolas as the corners of those polygons one can realize the n^{b+1} vertices of such an altered $CC_{2d}(n)$ using integral coordinates of size not more than n^2.

The smallest dimension for which this construction of products of cyclic polytopes yields a non-trivial result is $2d = 6$. It is an interesting open problem whether there exists a family of 4-dimensional polytopes with $\overline{f}(P) = \omega(size(P))$.

3.2 Intricate polytopes

We define a family of *intricate polytopes* as one consisting of polytopes P for which the number of (maximal) simplices required to triangulate all facets of P is much larger than $\mathit{size}(P)$. In other words, every triangulation of the boundary of P contains many more simplices than P has vertices and facets.

Families of uniparametric intricate polytopes have been known for a long time. The family $H_d = [0,1]^d$ of unit hypercubes was pointed out to us by Günter Rote, the family $TT_{2d} = T_d \times T_d$ of products of simplices by Bernd Sturmfels. It will turn out that any biparametric family of fat-lattice polytopes is also intricate.

Theorem 3 *1. Uniparametric families of intricate polytopes are provided by*

- *hypercubes* $H_d = [0,1]^d$, *and*
- *products of simplices* $TT_{2d} = T_d \times T_d$.

2. A biparametric family of intricate polytopes is provided by

- *products of cyclic polytopes* $CC_{2d}(n)$.

This theorem follows from the following three lemmas:

Lemma 2 *The number of $(d-1)$-simplices required to triangulate all facets of H_d is superpolynomial in $s_d = \mathit{size}(H_d)$, in particular, it is at least $s_d^{\frac{1}{4}\log\log s_d - 1/2}$.*

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

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

A lower bound on the number of n-simplices necessary to triangulate H_n can be obtained using the following volume-based argument (see e.g. [23]): H_n has volume 1. Any n-simplex of a triangulation of H_n has all its vertices on a sphere of diameter $\sqrt{n}$, and the maximal possible volume of such an inscribed n-simplex (realized by the equilateral one) is $V_n = (n+1)^{(n+1)/2}/(2^n n!)$. Hence the number of n-simplices in any triangulation of H_n is at least $1/V_n$.

Thus to triangulate the $2d$ facets of H_d (each of which is a $(d-1)$-cube) requires at least

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

$(d-1)$-simplices. But $(d/2) = \frac{1}{4}\log u_d \ge \frac{1}{4}\log s_d$ (here we use the binary logarithm) and thus triangulating the boundary of H_d requires at least

$$(d/2)^{d/2} \ge \left(\frac{1}{4}\log s_d\right)^{\frac{1}{4}\log s_d} = s_d^{\frac{1}{4}\log\log s_d - 1/2}$$

$(d-1)$-simplices, as claimed. ■

The superpolynomiality achieved by hypercubes is only very slight. Products of simplices achieve a much bigger bound.

Lemma 3 *The number of $(2d-1)$-simplices required to triangulate all facets of TT_{2d} is superpolynomial in $s_d = size(H_d)$, in particular, for $d > 3$ it is at least $2^{\sqrt[3]{s_d}}$.*

Proof: ¿From Lemma 1 we know that $TT_{2d} = T_d \times T_d$ has $(d+1)^2$ vertices and $2(d+1)$ facets. Thus for $d > 3$ we have

$$size(TT_{2d}) = 2d(d+1)(d+3) = s_d < (2d-1)^3 = u_d\,.$$

Each facet of TT_{2d} is combinatorially equivalent to $T_d \times T_{d-1}$. Now it is an interesting and useful fact that *every* triangulation of $T_s \times T_t$ requires *exactly* $\binom{s+t}{t}$ simplices. For a volume based proof see [23]. Thus triangulating the boundary of TT_{2d} requires

$$2(d+1)\binom{2d-1}{d} \geq 2^{2d-1}$$

simplices — the inequality here follows from the fact that $\binom{2d-1}{d}$ is the largest of the $2d$ binomial coefficients $\binom{2d-1}{i}$, which sum up to 2^{2d-1}. But $2d-1 = \sqrt[3]{u_d} > \sqrt[3]{s_d}$, and the lemma follows. ■

The biparametric part of Theorem 3 is an easy consequence of the following lemma:

Lemma 4 *Every biparametric family of fat-lattice polytopes is also a biparametric family of intricate polytopes.*

This follows directly from the following:

Lemma 5 *Any triangulation of the boundary of a d-polytope P contains at least $(\overline{f}(P)-1)/2^d$ maximal simplices.*

Proof: Let Δ be a triangulation of the boundary of P, i.e. a set of $(d-1)$-simplices. For $0 \leq i \leq d-1$ every proper i-face of P must contain at least one i-face of a simplex of Δ. But the total number of i-faces of simplices in Δ summed over all i is smaller than $|\Delta| \cdot 2^d$ since every simplex has 2^d faces. Thus

$$|\Delta| \cdot 2^d > \sum_{0 \leq i \leq d-1} f_i(P) = \overline{f}(P) - 1$$

and the lemma follows. ■

3.3 Dwarfed polytopes

In this section the notion of a *dwarfing halfspace* or *dwarfing constraint* will be crucial. We say a halfspace H (or its defining constraint) *dwarfs* a d-polytope P with m facets iff $P \cap H$ is a truncation d-polytope with $m+1$ facets (and hence has a minimum possible number $\beta(d, m+1) = (d-1)(m+1-d)+2$ of vertices).

A *family of dwarfed polytopes* is a family containing polytopes of the form $P' = P \cap H_P$, where H_P dwarfs P and $size(P)$ is much larger than $size(P')$. In other words, P has many vertices but the intersection with halfspace H_P removes most of the vertices, but no facet.

We will first state some families of uniparametric and biparametric dwarfed polytopes. Then we state the Dwarfing Theorem, an easy characterization of dwarfing halfspaces. We apply this theorem to show that we indeed have families of dwarfed polytopes, and then we finally prove the Dwarfing Theorem.

Theorem 4 (Dwarfed Cubes) *Let K_d be the d-dimensional cube specified by the $2d$ constraints $0 \leq x_i \leq 2$ for $1 \leq i \leq d$, and let H_d be the halfspace specified by $\sum_{1 \leq i \leq d} x_i \leq 3$.*

The family $DK_d = K_d \cap H_d$ is a family of dwarfed polytopes. In particular, we have

$$size(K_d) = d(2d + 2^d) \qquad \text{and} \qquad size(DK_d) = d^3 + 2d^2 + 2d\,.$$

For biparametric families we first state a theorem showing that dwarfing can take on the most extreme form, in that a d-polytope with m facets and a maximum possible number of $\gamma(d,m)$ vertices can be dwarfed to a d-polytope with $m+1$ facets and a minimum possible number of $\beta(d, m+1)$ vertices.

Theorem 5 (Dwarfed Dual Cyclic Polytopes) *For every m-facet d-polytope P that is dual to a cyclic polytope there is a dwarfing halfspace.*

Although dwarfed dual cyclic polytopes are theoretically intriguing, they are somewhat problematic from a more practical point of view. The specification of cyclic polytopes (and their duals) tends to require rather large numbers. The specification of the dwarfing constraint whose existence we assert requires much bigger numbers yet. For this reason we consider another family of dwarfed polytopes, namely dwarfed products of polygons, whose "dwarfing performance" is asymptotically similar to the one of dwarfed dual cyclic polytopes.

The following is set in *even* dimension $d = 2\delta$. For convenience we will refer to the d coordinates as $x_1, \ldots, x_\delta$ and $y_1, \ldots, y_\delta$.

Theorem 6 (Dwarfed Products of Polygons) *For $d = 2\delta \geq 4$ and $s \geq 3$ let $PP_d(s)$ be the polytope specified by the constraints*

$$\begin{array}{rcll} y_k & \geq & 0 & \text{for } 1 \leq k \leq \delta \\ s x_k - y_k & \geq & 0 & \text{for } 1 \leq k \leq \delta \\ (2i+1)x_k + y_k & \leq & (2i+1)(s+i) - i^2 + s^2 & \text{for } 1 \leq k \leq \delta \text{ and } 0 \leq i < s-3 \\ (2s-3)x_k + y_k & \leq & 2s(2s-3) & \text{for } 1 \leq k \leq \delta, \end{array}$$

and let $H_{d,s}$ be the halfspace specified by the constraint

$$x_1 + x_2 + \ldots + x_\delta \leq 2s - 1\,.$$

The family $DPP_d(s) = PP_d(s) \cap H_{d,s}$ is a biparametric family of dwarfed polytopes. In particular, we have (with $\delta = d/2$)

$$size(PP_d(s)) = d(s^\delta + \delta s) \qquad \text{and} \qquad size(DPP_d(s)) = d^2(\delta(s-2)+2) + 2d\,.$$

The preceding three theorems are all corollaries to the following dwarfing theorem.

Theorem 7 (Dwarfing Theorem) *Let P be a simple d-polytope with m facets and H a halfspace with no vertex of P on its boundary. If the vertices and edges of P that are contained in H form a tree with $m+1-d$ nodes, then H is a dwarfing halfspace for P.*

Thus $P' = P \cap H$ has $m+1$ facets, $\beta(d, m+1) = (d-1)(m+1-d) + 2$ vertices, and $size(P') = d^2(m+2-d) + 2d$.

Before we prove this Dwarfing Theorem, we apply it to prove Theorems 4 through 6. We will refer to the vertices and edges of P that are contained in H as *surviving* and to edges of P that are intersected by the bounding hyperplane of H as *cut edges*.

Proof (for dwarfed cubes): The polytope K_d is a d-cube, which has 2^d vertices and $m = 2d$ facets. Thus $size(K_d) = d(2d + 2^d)$.

The vertices of K_d are the 2^d points in $\mathbb{R}^d$ with all coordinates 0 or 2. The surviving vertices, i.e. the ones contained in H_d, are the $d+1$ points with at most one non-zero coordinate. The only surviving edges are the d edges that connect the origin to the other d surviving vertices, giving as graph of surviving vertices and edges a tree with $d+1$ nodes. Now apply the Dwarfing Theorem. ■

Proof (for dwarfed dual cyclic polytopes): Suppose P has a 2-face F that is a polygon with $m+2-d$ vertices. Let v be one of those vertices and let W be the remaining $m+1-d$ vertices. Clearly there is a hyperplane h that separates W from the remaining vertices of P (let ℓ be a line that separates v from W in the 2-plane aff$F$, and let h be a hyperplane containing ℓ that is a suitably small perturbation of a hyperplane that supports P in the face F). Let H be the closed halfspace bounded by h that contains W. 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 $\gamma(d-2, m)$ 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 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 $(d-2)$-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. ■

Proof (for dwarfed products of polygons): For any k one can easily check that the constraints listed above describe an s-sided convex polygon $P_k(s)$ in the x_k-y_k-plane, whose s vertices lie on the parabola $y_k = -(x_k - s)^2 + s^2$ and have integral coordinates with the x_k-coordinates drawn from the set $W(s) = \{0, s, s+1, s+2, \ldots, 2s-4, 2s-3, 2s\}$ (see Figure 3.3). The polytope $PP_d(s)$ is then the product $P_1(s) \times P_2(s) \times \cdots \times P_\delta(s)$. Thus we know from Lemma 1 that $PP_d(s)$ is simple, that it has s^δ vertices and $m = \delta s$ facets. Thus $size(PP_d(s)) = d(s^\delta + \delta s)$, as claimed.

By considering just the "x-coordinates" the vertex set can be identified with $W(s)^\delta$. Moreover that natural orthogonal lattice on $W(s)^\delta$ (with wrap-around between $2s$ and 0) yields the δs^δ edges of $PP_d(s)$.

It is easy to see that considering halfspace $H_{d,s}$ the only surviving vertices are the $\delta(s-2)+1$ vertices whose "x-coordinates" are all 0 except for possibly one, which however must not be $2s$. The surviving edges form δ paths emanating from the origin with $s-2$ edges each, one along each "x-coordinate" direction.

Thus the surviving vertices and edges form a tree with $\delta(s-2)+1 = m-d+1$ nodes. Now apply the Dwarfing Theorem. ■

Now it just remains to prove the Dwarfing Theorem. We do this in the following sequence of lemmas.

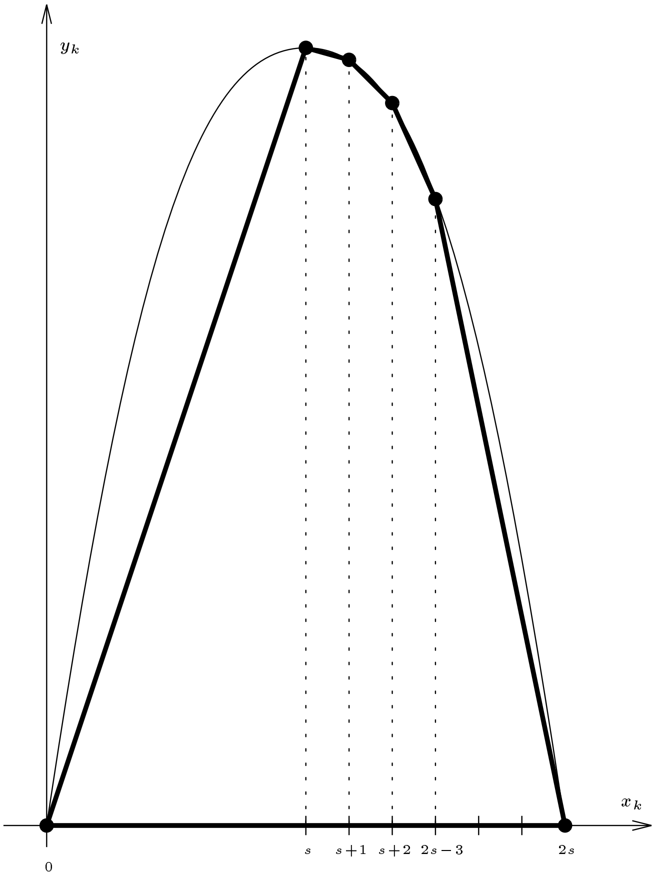

Figure 1: The polygon $P_k(6)$

Lemma 6 *Let P be a d-polyhedron and H a closed halfspace in $\mathbb{R}^d$ with bounding hyperplane h so that h contains no vertex of P.*

1. *The vertex set of $P' = P \cap H$ comprises (i) all surviving vertices of P, and (ii) all points of the form $e \cap h$, where e is an edge of P.*
2. *The facet set of P' comprises (a) the "new facet" $F' = P \cap h$, and (b) the "old facets" $F' = F \cap H$ where F ranges over all facets of P that contain some surviving vertex.*

Proof: See [8], Theorem 11.11. ■

Lemma 7 *If P is bounded, then the subgraph of the skeleton of P formed by the surviving vertices and edges 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. From the correctness of the simplex method with Bland's pivot rule (see e.g. [12]), there is a path in $P \cap H$ to the optimum face F. 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 along surviving edges. By Balinski's Theorem (see e.g. [7]), the skeleton of F is connected, hence the graph formed by surviving vertices and edges is connected. ∎

Lemma 8 *Let P be a simple d-polytope, and let H be such that h contains no vertex of P and so that the graph G formed by the surviving vertices and edges forms a tree with t nodes.*

Then $P' = P \cap H$ is a truncation polytope with $t + d$ facets (and $\beta(d, t + d) = (d - 1)t + 2$ vertices).

Proof: We first show that P' has the claimed number of vertices. ¿From Lemma 6 we know that every one of the t surviving vertices is a vertex of P'. It now suffices to show that there are $(d - 2)t + 2$ cut edges, i.e. edges of P that are intersected by h, since they yield the remaining vertices of P'. Each cut edge must be incident to exactly one surviving vertex. Since P is simple the total number of incidences to the t surviving vertices is dt. Since G is a t-node tree there are exactly $t - 1$ surviving edges, each of which is incident to two surviving vertices. Thus of the dt incidences exactly $2(t - 1)$ are consumed by the surviving edges, leaving $(d - 2)t + 2$ incidences to cut edges, as claimed.

Next we need to show that P' has $t + d$ facets. According to Lemma 6 it suffices to show that in total $t - 1 + d$ facets of P are incident to surviving vertices.

Since P is simple every vertex is incident to exactly d facets. If two vertices are connected by an edge of P, then the sets of incident facets differ by one.

Now root the tree G at any node v. With each node $w \neq v$ in G associate the facet F_w that is incident to w but not to the parent of w. Thus we have $t - 1 + d$ facets associated to surviving vertices: the $t-1$ facets F_w and the d facets incident to the root v. All these facets must be distinct, since G contains no cycle but, as a consequence of Lemma 7, the graph formed by the surviving vertices and edges that lie in one facet must be connected.

Since h contains no vertex of P we thus have a simple d-polytope P' with $t + d$ facets and $(d - 1)t + 2$ vertices. By Barnette's lower bound theorem [6] this vertex number is minimum possible.

Finally, P' is a truncation polytope. For $d > 3$ this is implied by Barnette's theorem. However, in our case it can be seen directly for all d: successively removing leaves w from the tree G and listing the halfspaces defining the corresponding facets F_w in reverse order yields a "truncation order." ∎

Obviously this last lemma immediately implies the Dwarfing Theorem.

4 Families of Algorithms

Now with our polytope classes in place it will be relatively easy to provide hard examples for the various facet and vertex enumeration algorithms.

Recall that a *lattice producing* algorithm is a vertex (or facet) enumeration algorithm that produces not just the vertices (facets) of the polytope in question but also computes all faces of that polytope.

Theorem 8 *If a lattice producing algorithm is used to enumerate the vertices (facets) of a fat-lattice polytope P, then the number of steps taken by the algorithm is much larger than $size(P)$. In particular, the number of steps taken for a polytope P with $s = size(P)$ is at least*

- $2^{\sqrt{s/2}}$, *if P is a simplex T_d, and*
- $\Omega(s^{d/\lceil\sqrt{d}\rceil})$ *for every fixed d, if P is a product of cyclic polytopes $CC_d(n)$.*

Proof: Immediate consequence of Theorems 1 and 2. ∎

Recall that a *facet* enumeration algorithm is *triangulation producing* if besides the facets of a polytope it also produces a triangulation of those facets. A *vertex* enumeration algorithm is considered triangulation producing if its dual interpretation is a triangulation producing facet enumeration algorithm.

For any halfspace $H = \{\, x \mid \langle a, x\rangle \leq 1 \,\}$ containing the origin in its interior, and for any vector ϵ let $H(\epsilon)$ denote the halfspace $\{\, x \mid \langle a+\epsilon, x\rangle \leq 1 \,\}$. Given a set of halfspaces $\mathcal{H} = \{\, H_1 \ldots H_m \,\}$, we say that a $\mathcal{H}' = \{\, H_1(\epsilon_1), \ldots H_m(\epsilon_m) \,\}$ is a *perturbation* of $\mathcal{H}$ if the following conditions hold:

1. If $\{\, H_{i_1}(\epsilon_{i_1}) \ldots H_{i_d}(\epsilon_{i_d}) \,\}$ is a basis of $P' = \bigcap \mathcal{H}'$, then $\{\, H_{i_1} \ldots H_{i_d} \,\}$ is a basis of $P = \bigcap \mathcal{H}$.
2. If v is a vertex of P, there is some basis $\{\, H_{i_1} \ldots H_{i_d} \,\}$ of v such that $\{\, H_{i_1}(\epsilon_{i_1}) \ldots H_{i_d}(\epsilon_{i_d}) \,\}$ is a basis of P'.

The second condition is necessary in order to correctly enumerate the vertices of P by enumerating the vertices of P'. The first is also useful, although not strictly necessary (since we could weed out bad bases in a postprocessing step), but is implied by the usual requirement that a perturbation be "sufficiently small". When proving lower bounds on the number of bases of the perturbed polytope (and thus on the performance of a pivoting algorithm that visits every basis) we may always assume without loss of generality the we perturb onto a simple polytope. To see why, consider the dual situation of perturbing the vertices of a polytope. If the resulting polytope is not simplicial, we can always triangulate without increasing the number of bases.

All algorithms that perturb onto a simple polytope in order to deal with degeneracies are triangulation producing (see [33] for a survey of perturbations). Perturbations have been reasonably successful in practice even on highly degenerate examples (see e.g. Ceder et al. [9]) and there had been some hope that by judicious choice of the perturbation the sizes of the produced triangulations could be kept reasonably small. However, intricate polytopes show that this is in general not the case.

Theorem 9 *If a triangulation producing algorithm is used to enumerate the facets of an intricate polytope P, then the number of steps taken by the algorithm is much larger than size(P). In particular, the number of steps taken for a polytope P with $s =$ size(P) is at least*

- $s^{\frac{1}{4}\log\log s - 1/2}$, *if P is a hypercube H_d,*
- $2^{\sqrt[3]{s}}$, *if P is a product of simplices $TT_{2d} = T_d \times T_d$, and*
- $\Omega(s^{d/\lceil\sqrt{d}\rceil})$ *for every fixed d, if P is a product of cyclic polytopes $CC_d(n)$.*

Proof: Immediate consequence of Theorem 3 and its following lemmas. ∎

All currently known graph traversal based algorithms for vertex or facet enumeration are either lattice producing or triangulation producing. The last two theorems show that they all definitely do not have polynomial worst case running time. We should stress, however, that this bad behaviour of graph traversal based algorithms only happens in the presence of degeneracies. If the input is non-degenerate, a reasonably implemented graph traversal based algorithm does have running time polynomial in *size*(P). For instance, in order to enumerate the n vertices of a *simple* d-polytope specified by m constraints the reverse search method [4] requires $O(d^3mn)$ time and only $O(dm)$ space.

4.1 Incremental algorithms with static orders

Let us now turn our attention to incremental algorithms. There is a fair number of such algorithms that for the degenerate case are lattice producing or triangulation producing. For such algorithms of course the lower bounds supplied by Theorems 9 and 8 apply, and thus polynomial worst case running time is not achievable. There is at least one incremental method though, namely the double description method of Motzkin et al. [30], that deals with degeneracies in an entirely different manner. We will now show that, irrespective of how incremental methods address the issue of degeneracy, they can have bad running times because they cannot control the sizes of the intermediate polytopes during the incremental construction. Moreover, this can happen for a host of natural insertion orders.

Consider enumerating the vertices of the d-polytope $P = \bigcap \mathcal{H}$. An incremental algorithm does this by considering the halfspaces in $\mathcal{H}$ in some order $H_1, H_2, \ldots, H_m$, and inductively computing some description of $P_i = \bigcap_{1 \le k \le i} H_k$ from the description of P_{i-1} and H_i, for $i = 1, \ldots, m$. From the description of the final P_m, the vertices of $P = P_m$ are then recovered. Typical "descriptions" of P_i are the skeleton of P_i, or, in the case of the double description method, the incidence information between the vertices and the facets of P_i. Let us ignore for the time being the issue of how such an algorithm gets off the ground and how it deals with unbounded P_i's.

In order for such an algorithm to be polynomial in *size*(P), the size of each of the intermediate polytopes P_i must be polynomial in *size*(P). Of course whether or not this is the case depends very much on the ordering of the halfspaces in $\mathcal{H}$.

There are several plausible insertion orders for insertion algorithms:

minindex Insert the halfspaces in the order given by the input.

random Insert the halfspaces in random order.

lex Insert the halfspaces in lexicographically increasing (or decreasing) order of coefficient vectors.

Such a lexicographic ordering is not canonical unless the inequalities describing the halfspaces are brought into some canonical form. We will consider two such canonical forms. The first, **unit form** requires inequalities $a_1x_1 + \cdots a_dx_d \le a_{d+1}$ with the largest indexed nonzero a_i as ± 1; the second, **reduced integer form** requires the a_i's to be relatively prime integers. Even after establishing a canonical form one can distinguish two more subcases, the *forward* case, where one considers a_1 the most significant component in $(a_1, \ldots, a_d, a_{d+1})$, and the *backward* case, where one considers a_{d+1} most significant.

The orderings described so far are all *static* in the sense that the ordering can easily be precomputed before the actual incremental enumeration algorithm has started. Somewhat more sophisticated are *dynamic* orderings in which the i-th halfspace to be inserted is some function of P_{i-1} and the not yet considered halfspaces. Before we go on to specific dynamic orderings we state our results for the listed static orderings.

Theorem 10 *Let $P = \bigcap \mathcal{H}$ be a polytope from a family of dwarfed polytopes, and $H \in \mathcal{H}$ is the dwarfing constraint. Assume an incremental algorithm is used to enumerate the vertices of P.*

minindex *If an ordering of $\mathcal{H}$ is used that considers the dwarfing halfspace H last, then the number of steps taken by the algorithm is much larger than size(P).*

random *If an ordering of $\mathcal{H}$ is chosen uniformly at random, then the expected number of steps taken by the algorithm is much larger than size(P).*

lex *For any type of lexicographic ordering there is an affine transformation taking $P = \bigcap \mathcal{H}$ to $P' = \bigcap \mathcal{H}'$ so that if the incremental algorithm is applied to $\mathcal{H}'$ using this lexicographic ordering, then the number of steps taken is much larger than size(P').*

Considering dwarfed cubes and dwarfed products of polygons as specific families of dwarfed polytopes one gets the following explicit statements.

Theorem 11 *Although the size of a dwarfed cube DK_d is only size$(DK_d) = d^3 + 2d^2 + 2d$, the (expected) number of steps taken by in incremental algorithm to enumerate the vertices of DK_d is at least*

- 2^d, *if an ordering is used that considers the dwarfing constraint last, for instance forward or backward lexicographic increasing order with respect to integer form,*
- $2^d/(d+1)$, *if a random ordering is used,*
- 2^{d-1}, *if forward or backward lexicographic increasing order is used with respect to unit form.*

Theorem 12 *Although the size of a dwarfed product of polygons $DPP_d(s)$ is only size$(DPP_d(s)) = d^3(s-2)/2+2d^2+2d$, the (expected) number of steps taken by an incremental algorithm to enumerate the vertices of $DPP_d(s)$ is at least*

- $s^{d/2}$, *if an ordering is used that considers the dwarfing constraint last, for instance forward lexicographic order with respect to unit form,*
- $s^{d/2}/(d+1)$, *if a random ordering is used,*
- $s^{d/2-1}$, *if forward lexicographic increasing order is used with respect to reduced integer form, and*
- $(s-3)^{d/2}$, *if backward lexicographic decreasing order is used with respect to unit or also reduced integer form.*

The bounds in Theorem 10 are provided by the size of the polytope that has been built up by the time the dwarfing constraint is inserted.

The first point follows directly from the properties of dwarfed polytopes. Let halfspace H dwarf polytope P to $Q = P \cap H$. If an incremental algorithm considers H last, then by that time a description of P must have been built up, which must have taken at least *size*(P) steps. But by the definition of families of dwarfed polytopes we have *size*(P) is much larger than *size*(Q).

The second point about random orderings follows from the following lemma:

Lemma 9 *Let $P = \bigcap \mathcal{H}$ be an m-faceted d-polytope with n vertices that is dwarfed by halfspace H to $Q = P \cap H$. Let C be the polytope formed by the intersection of the halfspaces in $\mathcal{H}$ that precede the dwarfing constraint H in a random ordering of the halfspaces in $\mathcal{H} \cup \{H\}$.*

Then the expected number of facets of C is at least $m/2$ (which is $m/(d+1)$ at least), and the expected number of vertices of C is at least $n/(d+1)$.

Proof: Note that if a halfspace in $\mathcal{H}$ induces a facet of P and it appears in the ordering before the dwarfing halfspace H, then it also induces a facet of C. Since in a random ordering every one of the m facet-inducing halfspaces in $\mathcal{H}$ has a $1/2$ chance of appearing before the dwarfing halfspace, the expected number of facets of C is at least $m/2$.

Similarly, note that if d halfspaces in $\mathcal{H}$ induce a vertex of P and they all appear in the ordering before the dwarfing halfspace H, then they also induce a vertex of C. The d halfspaces appearing before H is the same as saying that H has to be the last out of those $d+1$ halfspaces. In a random ordering this happens with probability $1/(d+1)$. Thus every one of the n vertices of P has at least a $1/(d+1)$ chance of being a vertex of C and hence the expected number of vertices of C is at least $n/(d+1)$. ■

The statement in Theorem 10 about lexicographic order follows from the following lemmas:

Lemma 10 *Let P be any d-polytope and let F be a facet of P. Polytope P can be translated so that, after possible renaming and negating of coordinates, among all facet defining constraints the one defining F comes first (last) in the lexicographic ordering with respect to unit form (for the forward or backward case, as desired).*

Proof: Note that for constraints of the form $a_1x_1 + \cdots a_dx_d \leq 1$, forward lexicographic order just reflects the order in which the described hyperplanes intersect the x_1-coordinate axis, the order starting at the origin, going towards $-\infty$ wrapping around through infinity and coming back to the origin (as in a shelling). Ties are broken lexicographically by considering the intersection order along the x_2-axis, x_3-axis, etc.

Now let i be such that F is not parallel to the x_i-axis. Rename coordinates so that x_i becomes x_1. Pick a line ℓ that is parallel to the x_1-axis and that intersects F in its relative interior. Pick a point $u \in \ell$ in the interior of P, and perform a translation so that u becomes the origin.

After this transformation all constraints describing P can be put in the form $a_1x_1 + \cdots a_dx_d \leq 1$; the line ℓ has become the x_1-axis, and (after possibly reversing the axis' direction) the hyperplane containing F comes first (or last) in the described intersection order, and the corresponding constraint therefore is first (or last) in the lexicographic order.

To achieve the same for backwards lexicographic order let x_d play the role of x_1.

Note that by picking u to be a rational point the transformed P can be described rationally provided the original P can be so described. ■

Lemma 11 *If polytope P is described rationally and contains the origin in its interior, then it is possible to scale the coordinate system so that lexicographic orders with respect to unit form and reduced integer form agree.*

Proof: Under these assumptions every facet defining constraint can be written as $(b_1/c_1)x_1 + (b_2/c_2)x_2 + \cdots + (b_d/c_d)x_d \leq 1$, with the b_i's integers and the c_i's positive integers. For each i let C_i be the least common multiple of all denominators c_i, ranging over all constraints. Let

$a_i = (b_i/c_i) \cdot C_i$ and $y_i = x_i/C_i$. Then $a_1y_1 + \cdots + a_dy_d \leq 1$ is equivalent to the constraint above, and is in reduced integer form, since the a_i's are integers and the right hand side is 1. As C_1 is positive, the order among the coefficients b_1/c_1 is the same as among the a_1's. ∎

The bounds stated in Theorems 11 and 12 are simply provided by the (expected) number of vertices of the polytope that has been created by the time the dwarfing constraint is to be considered in the respective ordering. This is clear for the case when the dwarfing constraint is added last. It follows from Lemma 9 for the random case. Regarding the various lexicographic orderings we leave the checking of the details of the dwarfed cube case to the reader. This leaves us with the lexicographic orderings for dwarfed products of polygons $DPP_d(s) = PP_d(s) \cap H_{d,s}$. Recall that $PP_d(s)$ is specified by

$$\begin{array}{rcll} -y_k & \leq & 0 & \text{for } 1 \leq k \leq d/2 \\ -sx_k + y_k & \leq & 0 & \text{for } 1 \leq k \leq d/2 \\ (2i+1)x_k + y_k & \leq & (2i+1)(s+i) - i^2 + s^2 & \text{for } 1 \leq k \leq d/2 \text{ and } 0 \leq i < s-3 \\ (2s-3)x_k + y_k & \leq & 2s(2s-3) & \text{for } 1 \leq k \leq d/2. \end{array}$$

Here we have normalized so that the right hand sides are upper bounds. The dwarfing constraint $H_{d,s}$ is given by

$$x_1 + x_2 + ... + x_{d/2} \leq 2s - 1.$$

Note that all constraints are already in reduced integer form, since each contains some unit coefficient.
Let us arrange the coordinates as $(x_1, \ldots, x_{d/2}, y_1, \ldots, y_{d/2})$.

Consider first forward lexicographic increasing ordering: with respect to unit form the dwarfing constraint $H_{d,s}$ comes last, i.e. all of $PP_d(s)$ has been built up and its $s^{d/2}$ vertices will have been created; with respect to reduced integer form form only the $s-2$ constraints with positive coefficient for x_1 come after $H_{d,s}$. Thus by the time $H_{d,s}$ is inserted $s^{d/2-1}$ vertices will have been generated.

In the backward lexicographic decreasing ordering with respect to both unit form and reduced integer form only the d constraints of the form $-y_k \leq 0$ and $-sx_k + y_k \leq 0$ come after the dwarfing constraint. This means that by the time dwarfing halfspace H is inserted a polytope with $(s-3)^{d/2}$ vertices has been constructed.

4.2 Incremental algorithms with dynamic orders

Some dynamic orders commonly used in incremental algorithms are the following:

maxcutoff Insert the halfspace next that causes the maximum number of vertices and extreme rays to become infeasible.

mincutoff Insert the halfspace next that causes the minimum number of vertices and extreme rays to become infeasible.

mixcutoff Insert the halfspace next whose bounding hyperplane produces the least balanced cut of the current vertices and extreme rays.

Note that using these dynamic orderings incurs overhead in that considerable computational effort may have to be expended in order to determine the next halfspace in the ordering.

The reader will notice that we have finally reached the point where we cannot continue to completely ignore unboundedness. Recall that any polyhedron P can be written a Minkowski sum

$P = B + C$, where B is a polytope, and C is a cone (called *recession cone*) (see e.g. Theorem 1.2 in [37]). "Vertex" enumeration for such a P in essence amounts to enumerating the vertices of the bounded part B and enumerating the extreme rays of the recession cone C. Note that the number of extreme rays of C may be much smaller than the number of extreme rays of P.

The orders defined above are not completely specified yet since it is not indicated how possible ties are to be broken. We will allow arbitrary resolution of ties, but with one exception, namely the beginning of the ordering. We require that the first d halfspaces be explicitly specified.

We have not been able to prove anything about **mixcutoff**, however for **mincutoff** and **maxcutoff** we have the following:

Theorem 13 *Consider the $2d+1$ constraints $0 \leq x_i \leq 2$ for $i = 1, \ldots, d$ and $\sum_{1\leq i\leq d} x_i \leq 3$ describing the dwarfed cube DK_d.*

In the **mincutoff** *ordering with initial d constraints $0 \leq x_i$ for $i = 1, \ldots, d$ the dwarfing constraint $\sum_{1\leq i\leq d} x_i \leq 3$ will come last.*

Thus incremental vertex enumeration of DK_d via such an ordering requires at least 2^d steps, which is much larger than size(DK_d).

Theorem 14 *For $d = 2\delta \geq 4$ and $s \geq 3$ consider the $\delta s + 1$ constraints*

$$\begin{array}{rcll} y_k & \geq & 0 & \text{for } 1 \leq k \leq \delta \\ s x_k - y_k & \geq & 0 & \text{for } 1 \leq k \leq \delta \\ (2i+1)x_k + y_k & \leq & (2i+1)(s+i) - i^2 + s^2 & \text{for } 1 \leq k \leq \delta \text{ and } 0 \leq i < s-3 \\ (2s-3)x_k + y_k & \leq & 2s(2s-3) & \text{for } 1 \leq k \leq \delta, \end{array}$$

and

$$x_1 + x_2 + \ldots + x_\delta \leq 2s - 1,$$

that describe the dwarfed product of polygons $DPP_d(s)$.

In the **mincutoff** *ordering with the initial d constraints $y_k \geq 0$ and $sx_k - y_k \geq 0$ for $i = 1, \ldots, \delta$ the dwarfing constraint $x_1 + x_2 + \ldots + x_\delta \leq 2s - 1$ will come last.*

Thus incremental vertex enumeration of $DPP_d(s)$ via such an ordering requires at least s^δ steps, which is much larger than size$(DPP_d(s))$.

These two theorems suggest a more general statement, which, however, we have been unable to prove.

Conjecture 1 *Let P be a dwarfed d-polytope described by the halfspaces in $\mathcal{H}$. There is a choice of d initial halfspaces so that in the ensuing* **mincutoff** *ordering the dwarfing halfspace will come last.*

Proving something analogous for the **maxcutoff** ordering has turned out to be much more difficult. Obviously dwarfed polytopes are not fruitful candidates since **maxcutoff** will very quickly cause the dwarfing constraint to be considered. The only result we have been able to prove is the following:

Theorem 15 *Consider the $(2d)$-polyhedron R_{2d} specified by the $2d + d^2$ constraints*

$$x_i \geq 0 \quad \text{for } i = 1, \ldots, 2d \tag{15.1}$$

$$x_i + x_j \geq 1 \quad \text{for } i = 1, \ldots, d \text{ and } j = d+1, \ldots, 2d. \tag{15.2}$$

The polyhedron R_{2d} has only 2 vertices and $2d$ extreme rays. However, for **maxcutoff** *order with the initial d constraints given by (15.1), the intersection of the first $2d+k$ halfspaces has 2^k vertices for $k = 0, \ldots, d$ (and $2d$ extreme rays).*

Thus incremental vertex enumeration of R_{2d} via such an ordering requires at least 2^d steps, which is much larger than size(R_{2d}).

The polyhedron R_{2d} is actually not a complete stranger. It is nothing but the dual of the product of simplices $TT_{2d} = T_d \times T_d$, where T_d is the d-simplex spanned by the origin and the unit coordinate vectors, and the "facet" of R_{2d} dual to the origin vertex of TT_{2d} is at infinity. We know that the polytopes TT_{2d} are intricate. We venture the following:

Conjecture 2 *Assume an incremental vertex enumeration algorithm is applied to the dual of an intricate d-polytope P using* **maxcutoff** *ordering and an arbitrary d initial halfspaces.*

The number of steps taken by the algorithm will be much larger than size(P).

Why should this conjecture be plausible? Consider the dual problem of incremental facet enumeration. The maxcutoff rule dualizes to adding next the vertex that "sees" the most facets of the current polytope. Now intricate d-polytope not only require "many" $(d-1)$-simplices to triangulate their boundary, their own triangulation naturally also requires "many" d-simplices. Intuitively this means that d-simplices spanned by the vertices of an intricate polytope must have "small" volume. In the incremental construction of such a polytope the dual maxcutoff rule adds the point that "sees" the most facets – and not necessarily facets of large volume. Thus it is likely that most of those facets are actually $(d-1)$-simplices. When the new point is added, most of the pyramids it spans with the visible facets are d-simplices, and hence little volume is added. Moreover, most of the new facets will be $(d-1)$-simplices again.

Of course this intuition is still far away from a proof. However our experimental results for products of cyclic polytopes (and also for dwarfed cubes, which happen to be an intricate polytope family also) support this conjecture. Finally, there is reason to believe that random ordering is similarly bad for intricate polytopes.

It remain to prove Theorems 13 through 15:

Proof of Theorem 13: Let Q be the positive orthant. For $i \in D = \{1, \ldots, d\}$ let r_i be the ray formed by the positive i-the coordinate axis and let $H^{(i)}$ be the halfspace specified by the constraint $x_i \leq 2$. Finally let H be the dwarfing halfspace specified by $\sum_i x_i \leq 3$.

For $I \subset D$ consider the polyhedron $K_I = Q \cap \bigcap\{H^{(i)} | i \in I\}$. It has $2^{|I|}$ vertices, namely all those vertices of the d-cube $K_d = [0,2]^d$ whose x_i-coordinates are 0 for all $i \notin I$. It has $d - |I|$ extreme rays, namely r_i with $i \notin I$.

Any halfspace $H^{(j)}$ with $j \notin I$ cuts off one extreme ray of K_I (namely r_j) and no vertex.

The dwarfing halfspace H cuts off all $d - |I|$ extreme rays, and also all but $|I|+1$ of the vertices.

This implies that as long as $I \neq D$ the mincutoff rule will prefer every $H^{(j)}$ with $j \notin I$ over the dwarfing halfspace H. ∎

Proof of Theorem 14: Let us write down once more the constraints for *DPP*. There is the dwarfing halfspace H given by

$$x_1 + x_2 + \ldots + x_{d/2} \leq 2s - 1\,,$$

and for each $k \in D = \{1, \ldots, \delta\}$ we have the constraint set $\mathcal{H}_k$ given by

$$
\begin{array}{rcll}
y_k & \geq & 0 & (1) \\
y_k & \leq & s x_k & (1) \\
(2i+1)x_k + y_k & \leq & (2i+1)(s+i) - i^2 + s^2 \quad \text{for } 0 \leq i < s-3 & (2) \\
(2s-3)x_k + y_k & \leq & 2s(2s-3) & (2).
\end{array}
$$

¿From now on let $\mathcal{I}_k$ be a subset of $\mathcal{H}_k$ that contains at least the two constraints labelled (1). Fix k, and consider the polygon $P(\mathcal{I}_k)$ in the x_k-y_k-plane formed by $\bigcap \mathcal{I}_k$. If $|\mathcal{I}_k| = 2$, then this polygon is the cone spanned by the two extreme rays r_k and r'_k in the positive orthant supported by the lines through the origin with slope 0 and s, respectively. Any constraint in $\mathcal{H}_k \setminus \mathcal{I}_k$ cuts off both those ray.

If $|\mathcal{I}_k| > 2$, then this polygon has $|\mathcal{I}_k|$ edges and vertices, of which one is the origin, another has x-coordinate at least $2s$, and the remaining ones have x-coordinates between s and $2s-3$. Any constraint in $\mathcal{H}_k \setminus \mathcal{I}_k$ cuts off exactly one vertex.

Let $\mathcal{I} = \mathcal{I}_1 \cup \ldots \cup \mathcal{I}_\delta$. Note that $\bigcap \mathcal{I}$ is the product of the polygons $P(\mathcal{I}_k)$ (or equivalently, their Minkowski sum). We can characterize $\bigcap \mathcal{I}$ as follows:

Let $K \subset D$ comprise those indices k for which $|\mathcal{I}_k| > 2$. The polyhedron $\bigcap \mathcal{I}$ is given by

$$
\underbrace{(\Pi_{k \in K} P(\mathcal{I}_k))}_{B} \times \underbrace{(\Pi_{k \notin K} P(\mathcal{I}_k))}_{C},
$$

where B is a polytope with $\Pi_{k \in K} |\mathcal{I}_k|$ vertices and C is a cone with $d - 2|K|$ extreme rays (namely r_k, r'_k with $k \notin K$.

Now consider a halfspace $I_k \in \mathcal{H}_k \setminus \mathcal{I}_k$ with $k \in K$: It cuts off no extreme ray, but it cuts off $\Pi_{\substack{j \in K \\ j \neq k}} |\mathcal{I}_j|$ vertices.

Next consider a halfspace $I_k \in \mathcal{H}_k \setminus \mathcal{I}_k$ with $k \notin K$: It cuts off the two extreme rays r_k and r'_k but no vertex.

Finally consider the dwarfing constraint H: It cuts off all $d - 2|K|$ extreme rays. Moreover, by the same reasoning as in the proof of Theorem 6 it is a dwarfing halfspace for B. Thus it cuts off all but $(\sum_{k \in K} |\mathcal{I}_k|) - 2|K| + 1$ vertices of B.

Thus if $K \neq D$ the mincutoff rule will prefer any halfspace in $\mathcal{H}_k \setminus \mathcal{I}_k$ with $k \notin K$ over the dwarfing halfspace H. If $K = D$ then using that $|\mathcal{I}_k| \geq 3$ it is straightforward to conclude that the mincutoff rule will prefer every other halfspace over the dwarfing halfspace H. ∎

Proof of Theorem 15: For $0 \leq k \leq d$ let W_k be the polyhedron specified by the following $2d + k$ constraints $\mathcal{H}_k$: the $2d$ non-negativity constraints in (15.1) and the k constraints $x_i + x_{d+i} \geq 1$ for $i = 1, \ldots k$ from (15.2). Let r_j denote the ray formed by the positive x_j axis.

Consider coordinate plane π_i spanned by x_i and x_{d+i}. If $k < i \leq d$, then the constraints in $\mathcal{H}_k$ that involve x_i and x_{d+i} specify the positive quadrant Q_i in π_i, which is a cone spanned by the two extreme rays r_i and r_{d+i}.

If $1 \leq i \leq k$, then the constraints in $\mathcal{H}_k$ that involve x_i and x_{d+i} specify an unbounded polygon P_i in π_i whose two vertices are the points $(0,1)$ and $(1,0)$ and whose two extreme rays are given by r_i and r_{d+i}. In other words, $P_i = B_i + Q_i$, where B_i is the two vertex polytope spanned by $(0,1)$ and $(1,0)$ and the recession cone is Q_i, the cone spanned by the extreme rays r_i and r_{d+i}.

Since W_k is the product of the polygons in the planes π_i we can conclude that W_k can be represented as Minkowski sum

$$W_k = \sum_{1 \le i \le k} (B_i + Q_i) + \sum_{k < i \le d} Q_i = \underbrace{\sum_{1 \le i \le k} B_i}_{C_k} + \underbrace{\sum_{1 \le i \le d} Q_i}_{Q},$$

i.e. W_k is the Minkowski sum of the non-negative orthant Q, which is spanned by the $2d$ extreme rays r_j, and of the k-dimensional cube C_k, which has 2^k vertices. Moreover, these vertices are exactly the 0-1 vectors with exactly one 1 and one 0 in position i and $d+i$ for $i = 1, \ldots, k$, and with only 0's in position i and $d+i$ for $i = k+1, \ldots, d$.

We now want to argue inductively that for $k = 0, \ldots, d$ maxcutoff order will produce exactly those polyhedra W_k. This is clear for $k = 0$ since $W_0 = Q$, which is the polyhedron spanned by the non-negativity constraints in (15.1), with we assume to be the initial constraints in the ordering.

Assume inductively that W_k has been produced, for $k < d$. No constraint from (15.2) can cut off any extreme ray r_j. A constraint $x_i + x_j \ge 1$ with $i \in \{1, \ldots, k\}$ or $j \in \{d+1, \ldots, d+k\}$ can cut off at most half of the vertices W_k. But any constraint $x_i + x_j \ge 1$ with $i \in \{k+1, \ldots, k\}$ and $j \in \{d+k+1, \ldots, 2d\}$ cuts off *all* vertices of W_k. Without loss of generality we may assume (by relabelling of coordinate indices if necessary) that the maxcutoff strategy therefore chooses as next constraint $x_{k+1} + x_{d+k+1} \ge 1$, and the constraint set is extended from $\mathcal{H}_k$ to $\mathcal{H}_{d+1}$, i.e. we have W_k intersected with the new constraint is W_{k+1}.

To complete the proof of the theorem it remains to show that R_{2d} has indeed only 2 vertices and $2d$ extreme rays.

It is clear that for $1 \le j \le 2d$ each r_j is an extreme ray, since it satisfies $2d-1$ of the constraints in (15.1) with equality and satisfies all other constraints. There can be no other extreme ray, since otherwise the recession cone would be larger than the non-negative orthant Q, which is clearly impossible because of the constraints in (15.1).

We claim that $(\underbrace{1, \ldots, 1}_{d}, \underbrace{0, \ldots, 0}_{d})$ and $(\underbrace{0, \ldots, 0}_{d}, \underbrace{1, \ldots, 1}_{d})$ are the only two vertices of R_{2d}. It is easy to check that the matrix given by the constraints is totally unimodular. This implies that all vertices of R_{2d} must be integral. This means that all vertices must be 0-1-vectors, since a larger component would not allow to satisfy any of the constraints in (15.2) with equality (which one would need, since a vertex must satisfy $2d$ independent constraints). Any 0-1-vector that has a 0 in in position $i \le d$ and in position $j > d$ cannot be a vertex since it violates that constraint $x_i + x_j \ge 1$. So assume all positions in, say, the second half are 1 and some position $i \le d$ in the first half is also 1. In this case no constraint involving x_i can be satisfied with equality, which means one cannot find $2d$ independent constraints that are satisfied with equality, as would be necessary for a vertex.

So this leaves as possible vertices only the two 0-1-vectors that have all 1's in one half and all 0's in the other. It is easy to check that they indeed do satisfy $2d$ independent constraints with equality, so therefore they are vertices. ■

5 Experimental Results

Using the examples described in the previous sections, we tested one pivoting algorithm that uses perturbation (`lrs`), one insertion algorithm that uses triangulation (`qhull`) and two "pure" insertion algorithms (`cdd+` and `porta`) based on the double description method [30].

What	Where
`cdd+`	ftp://ifor13.ethz.ch/pub/fukuda/cdd/cdd+-073.tar.gz
`lrs`	ftp://ftp-cgrl.cs.mcgill.ca/pub/polytope/soft/src/rs/lrs.c.Z http://www-cgrl.cs.mcgill.ca/polytope/soft/
`porta`	ftp://elib.zib-berlin.de/pub/mathprog/polyth/porta
`qhull`	ftp://geom.umn.edu/pub/software/qhull.tar.Z http://www.geom.umn.edu/software/qhull/
example polytopes	ftp://ftp-cgrl.cs.mcgill.ca/pub/polytope/examples/hgch/hgch_input.tar.gz http://www-cgrl.cs.mcgill.ca/polytope/examples

Table 1: Availability of Software and Data by anonymous ftp and WWW.

All of the software and data files described in this paper are available by anonymous ftp, and on the world wide web. See Table 1 for details. `cddf+` and `cddr+` are version 0.73 of Fukuda's implementation of the double description method [18], compiled respectively to use floating point and exact rational arithmetic. `porta` is version 1.2.1 of Christof, Loebel, and Stoer's implementation of the double description method. `qhull` is Barber and Huhdanpaa's implementation of "Quickhull" (a variant of the beneath and beyond algorithm), version 2.2 [5]. `lrs` is Avis' implementation of reverse search [4] using Edmonds Q-pivoting and lexicographic perturbation, version 2.5i. We compiled `qhull` to use double precision; `cddf+` uses double precision by default The option `C-0` was used to force `qhull` to merge the generated simplicial facets. Random insertion order for `cdd+` was simulated by permuting the order of constraints 100 times using the *Combined Random Number Generator* of L'Ecuyer [27] and reporting the average time.

In the following Vsize denotes the summed intermediate complexity for insertion algorithms. In particular, for `cdd+` it denotes the sum of number of extreme rays at each stage of the double description method. For `qhull` it denotes the summed number of *untriangulated* facets of intermediate polytopes. Tsize denotes the measured triangulation complexity of the polytope. For `qhull` this the sum of the triangulation sizes of the intermediate polytopes. For `lrs` it is the number of bases (and vertices) of the perturbed polytope. The notation *memory limit* on a graph indicates that this was the last run of a series to complete due to memory limitations. Times are measured in CPU seconds. All timings on a DEC3000/500 Alpha with 96M of physical memory and 342M of virtual memory, running OSF/1 1.3 (except where noted). Most of the examples were translated from those described in Section 3 so that the origin was contained in the interior in order to facilitate the use of the same files with both vertex and facet enumeration programs.

5.1 Rational versus Floating Point Arithmetic

The convex hull problem has the nice property that it is possible to perform all computations in exact rational arithmetic; this is especially desirable in applications such as combinatorial optimization where an exact answer is desired rather than just an approximation. One question currently being investigated by several researchers is the relative cost of using exact rational arithmetic instead of floating point or some hybrid scheme. To minimize the number of variables in the experiment, we used Fukuda's program `cdd+` which can be compiled to use rational or floating point arithmetic. In examples where input numbers are very large such as the products of cyclic polytopes, `cddr+` was thousands of times slower than `cddf+` on some inputs. Moreover, in the ratio between `cddr+` and `cddf+` gets worse as input size increases in the intricate polytope classes (products of simplices and products of cyclic polytopes) tested. On the other hand for the dwarfed cubes, the performance ratio is not nearly so bad, and tends to get better as the input size increases, probably because the set theoretic operations involved in maintaining vertex facet adjacency in the double description method begin to dominate the cost of computation.

In general `porta` far outperformed `cddr+` if both used the same insertion order. This is most likely because `cddr+` uses the very general purpose GNU rational arithmetic package while `porta` uses its own, presumably more highly tuned rational arithmetic library.

5.2 Products of Simplices

In this subsection we interpret the problem under consideration as a convex hull problem. As expected from Theorem 3 the triangulation complexity of the products of simplices explodes rather quickly (see Table 2). The degeneracy of the facets of TT_{2d} does not seriously effect the algorithms based on the double description method since they represent the intermediate polytopes by the extreme rays of a homogenization; in this case a "degenerate" polytope simply means that many of these extreme rays happen to lie on a given facet. On the other hand, choosing the maxcutoff insertion order does cause poor performance (see Figure 2). This is not too surprising since in transforming our input points for use as input to `cdd+`, we are just taking a dual that preserves boundedness, as opposed to translating one facet to infinity as in Theorem 15.

				Tsize	
d	*facets*	*vertices*	*size*	qhull	lrs
4	6	9	60	25	18
6	8	16	144	102	80
8	10	25	280	428	350
10	12	36	480	1768	1512
12	14	49	756	7414	6468
14	16	64	1120	31353	27456
16	18	81	1584	(a)	115830
18	20	100	2160		486200
20	22	121	2860		2032316

(a) System thrashed.

Table 2: Triangulation complexity for products of simplices

5.3 Dwarfed Cubes

In this subsection, the problems are interpreted as vertex enumeration problems. The data files for the dwarfed cubes have the dwarfing constraint last in the file so the minindex insertion order is guaranteed to build the entire d-cube. The dwarfed cubes are simple, so the lone pivoting algorithm (`lrs`) performs extremely well (see Figure 4). A comparison of Figures 3 and 4 shows that the asymptotic performance of `cddr+` and `cddf+` appears different, even though the number of rays in the intermediate cones is the same (see Table 3). It appears that the intermediate polytopes computed by `qhull` are not simple, since the triangulation complexity for `qhull` grows much

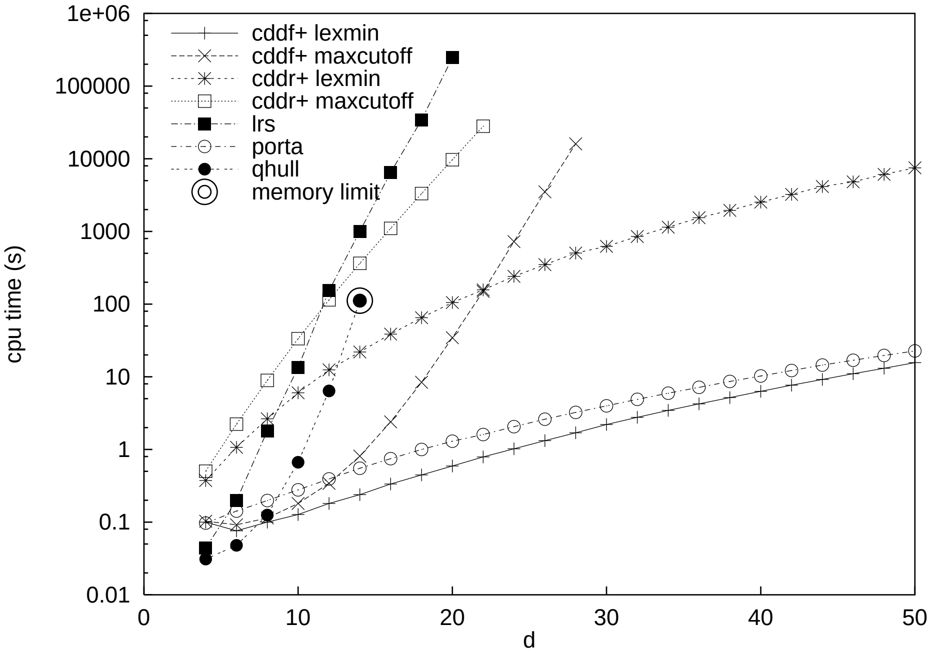

Figure 2: Timing results for products of simplices.

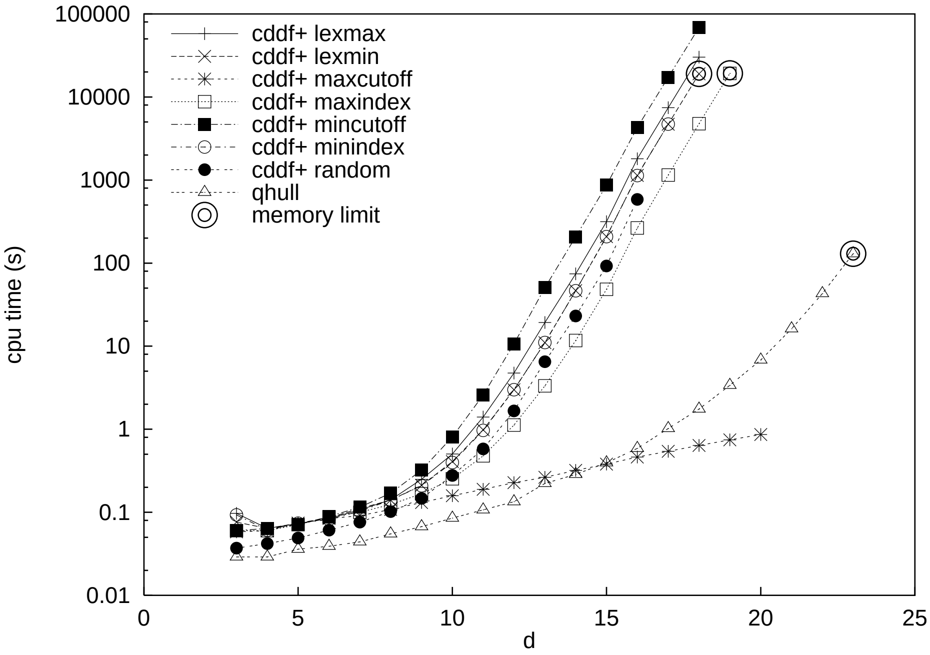

Figure 3: CPU time for dwarfed cubes, floating point arithmetic

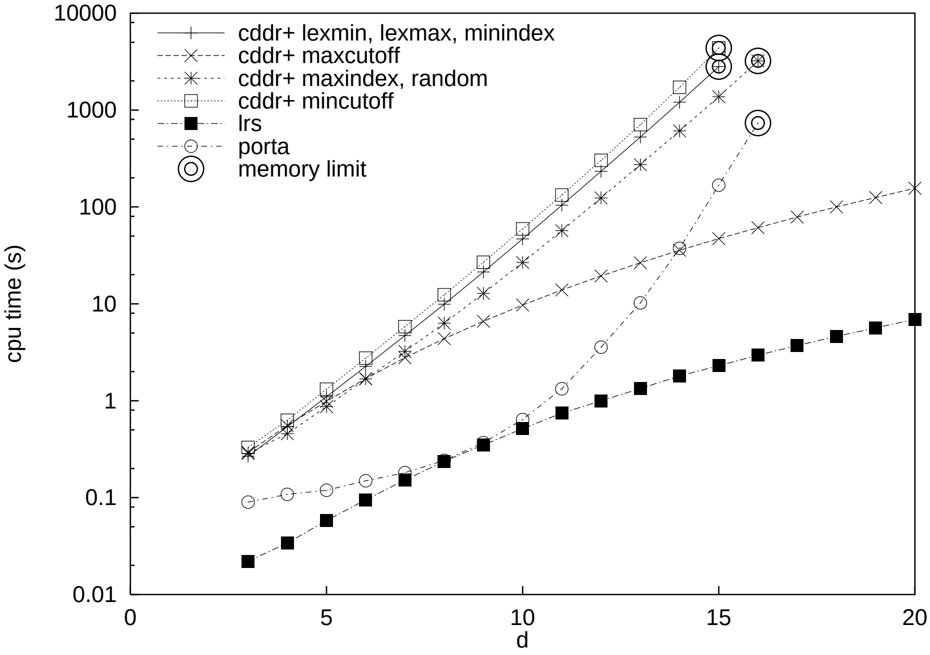

Figure 4: CPU time for dwarfed cubes, rational arithmetic

faster than the (untriangulated) intermediate complexity (see Table 4). The initial configuration of halfspaces chosen by `cdd+` (by lexicographic order) is exactly that specified by Theorem 13, so the poor performance of **mincutoff** on these polytopes is not a surprise. The only insertion order that performs well on these polytopes is **maxcutoff**.

5.4 Dwarfed products of polygons.

As in the dwarfed cube examples, **maxcutoff** works quite well and **mincutoff** performs extremely poorly on these polytopes (see Figure 5). Here again the initial configuration chosen by `cdd+` is lexicographic and matches that of Theorem 14. These polytopes are simple, so both `qhull` and `lrs` perform quite well. As we have seen with dwarfed cubes, even if the final polytope is simple, the intermediate polytopes are not necessarily simple. Here, however the intermediate polytopes are also products of polygons, hence simple. In Figure 6 we show how the performance of `cddf+` varies with dimension and number of input points for a fixed (bad) insertion order. Recall that in terms of the parameters on the graph the dimension is 2δ and the number of input points is $s\delta$.

5.5 Products of Cyclic Polytopes

Products of cyclic polytopes provide families in fixed dimension (in our experiments, $d = 8$) that are hard for for lattice producing and triangulation producing algorithms. We did not have any implementations of lattice producing algorithms to test, but our experience with the two triangulation producing algorithms (`lrs` and `qhull`) bears out the theory (see Figure 5). These polytopes also seem quite difficult for insertion methods. It is interesting to note that numerical instability

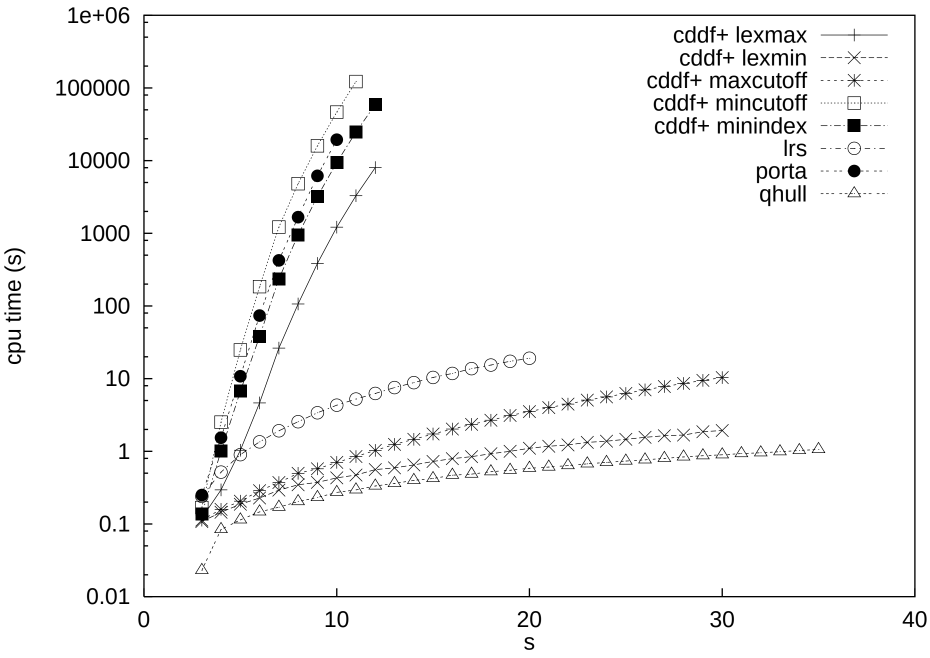

Figure 5: Performance of various insertion orders and programs on $\mathrm{DDP}_{10}(s)$.

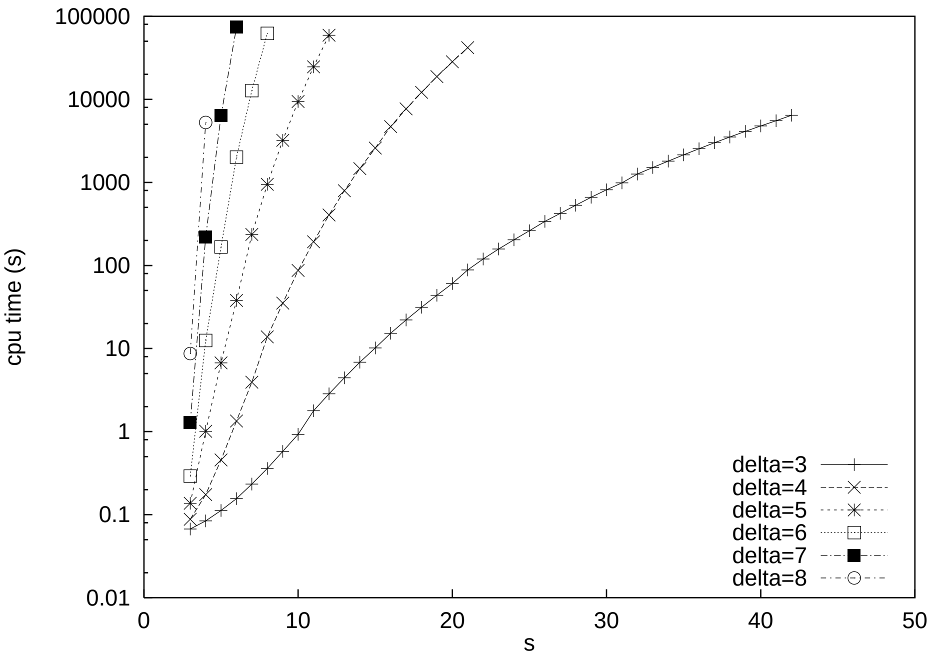

Figure 6: Performance of `cddf+` on $\mathrm{DDP}_{2\delta}(s)$ as δ varies, minindex insertion order.

d	*size*	Summed Intermediate Size, `cddf+`[b] minindex	maxindex	maxcutoff	mincutoff	lexmax	lexmin
3	51	10	10	13	11	17	11
4	104	19	26	17	20	32	20
5	185	36	43	31	37	29	37
6	300	69	38	37	70	38	70
7	455	134	71	57	135	71	135
8	656	263	136	65	264	136	264
9	909	520	265	91	521	265	521
10	1220	1033	522	101	1034	522	1034
11	1595	2058	1035	133	2059	1035	2059
12	2040	4107	2060	145	4108	2060	4108
13	2561	8204	4109	183	8205	4109	8205
14	3164	16397	8206	197	16398	8206	16398
15	3855	32782	16399	241	32783	16399	32783
16	4640	65551	32784	257	65552	32784	65552
17	5525	131088	65553	307	131089	65553	131089
18	6516		131090	325	262162	131090	262162
19	7619		262163	381			(a)
20	8840		(a)	401			

(a) Virtual Memory Exceeded.

(b) Intermediate sizes for completed `cddr+` runs are identical, although each ray may take more space to store in `cddr+`.

Table 3: Dwarfed cubes: intermediate size for `cdd+`.

problems caused by the coordinates on the moment curve seem to depend on insertion order.

5.6 Pierced Cubes

These examples are a historically earlier and somewhat more complicated class of polytopes that establish some of the same results as dwarfed cubes [2]. In particular they show that maxindex/minindex and lexicographic orders can be superpolynomial. In these examples, the cube constraints come last in the files, so maxindex is guaranteed to build the entire d-cube. Unlike the case of dwarfed cubes, maxcutoff does not perform well on these polytopes (see Table 6).

5.7 Products of simplices and cubes

It is not difficult to argue that taking the product of an intricate family of polytopes and a dwarfed family of polytopes gives you a family that is both dwarfed and intricate, although not neccesarily with exactly the same bounds. The polytopes tested in this subsection are the cross product of a dwarfed d-cube and the product of two d-simplices. We abbreviate to SSC polytopes, for *simplex* $\times$ *simplex* $\times$ *cube*. These polytopes are hard for the triangulation producing programs **`lrs`** and **`qhull`**, and for the minindex insertion order (see Table 7). Note however that the product

		Vsize				Tsize	
d	*size*	cddf+ random	cddr+ random	porta	qhull	lrs	qhull
3	51	14.690	14.690	33	14	10	14
4	104	27.060	26.910	65	22	17	22
5	185	44.840	46.640	118	32	26	32
6	300	74.480	73.730	208	44	37	44
7	455	124.330	123.470	367	58	50	58
8	656	208.030	217.680	659	74	65	74
9	909	349.200	377.270	1212	92	82	92
10	1220	682.070	652.820	2282	112	101	112
11	1595	1185.730	1229.830	4381	143	122	143
12	2040	2239.070	2251.010	8533	178	145	178
13	2561	4535.340	4670.980	16786	217	170	228
14	3164	8670.420	8448.000	33236	262	197	306
15	3855	16640.440	18004.790	66075	317	226	438
16	4640	34911.800		131687	390	257	676
17	5525			(a)	497	290	1124
18	6516				670	325	1990
19	7619				973	362	3690
20	8840				1534	401	7056
21	10185				2609		13752
22	11660				4710		27106
23	13271				8861		53774
24	15024				(a)		(a)

(a) Virtual Memory Exceeded.

Table 4: Dwarfed cubes: intermediate size and triangulation complexity for `cdd+` random insertion, `lrs`, `porta` and `qhull`

construction does not necessarily preserve lexicographic order without additional transformations, so the performance of lexicographic order on these polytopes is relatively better than that on the dwarfed cubes (compare with Table 3).

5.8 Practical Problems

We conclude with some very recent practical experience using the codes `cdd` and `lrs`. In practice one often has additional information about the polytope that allows an astute choice on insertion order. In [15] the authors describe the so-called *co-clique* ordering. In the vertex enumeration context, facets are grouped into maximal independent sets in the ridge graph of the polytope, and entered in this order. (For the polytopes they study, the ridge graph is known.) Using this ordering they were able to compute the 275,840 vertices of the metric polytope, defined by 140 facets in 21 dimensions with cdd. The computation took about 3 weeks on a Sony News NWS-5000 workstation at TIT. The computation failed for the lexicographic, mincutoff and maxcutoff rules, due to the large size of the intermediate polytopes. In [3] the same ordering produced excellent results for

		CPU time								
		`cddf+`			`cddr+`					
n	*size*	lexmin	maxcutoff	random	lexmin	maxcutoff	random	`lrs`	`porta`	`qhull`
5	280	0.112	0.155	0.104	4.35	15.9	4.03	1.62	0.135	0.134
6	432	0.144	0.318	0.174	17.4	140	37	15.8	0.210	0.719
7	616	0.206	2.24	0.657	65	1028	249	76	0.375	3.45
8	832	0.382	17.1	(b)	225	5285	1126	266	0.750	11.7
9	1080	0.802	83.7		646	20010		920	1.59	40.9
10	1360	1.80	(b)		1683	70537		2652	3.43	108
11	1672	4.01			3859			6538	7.45	191
12	2016	8.64			8442			14186	16.1	499
13	2392	18.1			16563			28051	33.7	900
14	2800	38.0			31882			51677	86.5	1407
15	3240	70.7						90689	176	(c)
16	3712	128							310	
17	4216	227							628	
18	4752	391							1046	
19	5320	679							1833	
20	5920	1113							2982	
21	6552	1797							5070	
22	7216	3265							6909	
23	7912	(b)							10389	
24	8640								15358	
25	9400								22455	

(a) These tests carried on under OSF/1 v3.2c

(b) Incorrect number of facets computed.

(c) System thrashed.

Table 5: Timing results for $CC_8(n)$.

many of the polyhedra considered. The intermediate polyhedra always had sizes between 1.2 and 2 times that of the original polyhedron. (Interestingly, initial experiments suggest that our products of cyclic polytopes provide a class of examples for which this co-clique ordering fares very badly.)

The hardest problem solved in [3] was a polytope in 15 dimensions with 250 facets with 0-1 coefficients and 101 444 extreme rays. This could not be solved by cdd, but was solved in three days by a parallel version of lrs implemented by Ambros Marzetta at ETH Zurich. This parallel version runs on an NEC Cenju-3 with 64 processors. Very recently ([19]) this code completed the enumeration of all bases of the configuration polytope with 71 facets in 60 dimensions (see [9]), which could not be solved by any other method. The polytope had 3 149 579 vertices, 57 613 364 bases, and the computation took 4.5 days (estimated at 130 days on a single processor). Many combinatorial polytopes, in particular the cut and metric polytopes, appear to have high triangulation complexity. It would be interesting to try and prove this.

d	*size*	Vsize cddf+ minindex	cddf+ maxindex	cddf+ maxcutoff	cddf+ mincutoff	cddf+ lexmax	cddf+ lexmin	qhull	porta maxindex
3	105	30	54	33	26	39	32	51	170
4	248	45	54	55	65	129	62	135	552
5	485	101	200	159	261	151	269	301	1523
6	840	108	248	142	505	613	109	806	3444
7	1337	168	537	362	375	209	190	1561	6782
8	2000	213	1536	382	992	2028	210	4339	12454
9	2853	263	521	1418	2656	996	266	7968	20392
10	3920	318	1034	793	4483	1058	894	24277	31613
11	5225	398	2059	577	16119	1429	576	(a)	49236
12	6792	465	4108	808	39345	4107	571		72333
13	8645	537	8205	745		9601	799		103986
14	10808	614	16398	7177		31591	1503		154062
15	13305	696	32783	2329			1201		225727
16	16160	813	65552	8529			2338		336370
17	19397	875	131089	31043			1289		(a)
18	23040	1006	262162	62840			1121		
19	27113	1110		19146					

(a) Virtual Memory Exceeded.

(b) Intermediate sizes for completed `cddr+` runs are identical to those for `cddf+`, although each ray may take more space to store in `cddr+`.

Table 6: Pierced cubes: summed intermediate size.

d	*size*	Vsize porta	qhull	cddf+ minindex	cddf+ lexmin	cddf+ lexmax	Tsize lrs	qhull
9	927	381	246	124	98	109	800	1120
12	2448	883	1338	180	210	188	5950	8471
15	5385	1824	6961	414	412	347	39312	53858
18	10440	3566	(a)	938	602	586	239316	(a)
21	18459	6855		2104	870	933	1372800	
24	30432	13269		4680	1314	1432	7528950	
27	47493	26178		10330	1928	2159		
30	70920	52792		22638	2442	3254		
33	102135	108502		49284	3082	4985		
36	142704	(a)		106652	4082	4131		
39	194337			(a)	5332	8230		

(a) Virtual Memory Exceeded.

(b) Intermediate sizes for `cddr+` are the same as for `cddf+`

Table 7: Intermediate size measurements and triangulation complexity for SSC polytopes.

Acknowledgements

We would like to thank the authors of the software tested, Bernd Sturmfels and Günter Rote for suggestions of families of polytopes with high triangulation complexity and an anonymous referee for pointing out the polynomial equivalence of polytope verification and vertex/facet enumeration.

References

[1] P. Armand. Bounds on the number of vertices of polyhedra. *Annals of Operations Research*, 47:249–269, 1993.

[2] D. Avis and D. Bremner. How good are convex hull algorithms? In *Proc. 11th Annu. ACM Sympos. Comput. Geometry*, pages 20–28, 1995.

[3] D. Avis and A. Deza. Solitaire Cones. Manuscript in preparation, 1996.

[4] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Comput. Geom.*, 8:295–313, 1992.

[5] B. B. Barber, D. P. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hull. Technical Report GCG53, The Geometry Center, University of Minnesota, July 1993.

[6] D. Barnette. The minimum number of vertices of a simple polytope. *Israel J. Math.*, 8:304–308, 1971.

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

[8] A. Brøndsted. *Introduction to Convex Polytopes.* Springer Verlag, 1981.

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

[10] D. Chand and S. Kapur. An algorithm for convex polytopes. *J. ACM*, 17:78–86, 1970.

[11] B. Chazelle. An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, 10:377–409, 1993.

[12] V. Chvátal. *Linear Programming.* W. H. Freeman, New York, NY, 1983.

[13] K. L. Clarkson. More output-sensitive geometric algorithms. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, pages 695–702, 1994.

[14] K. L. Clarkson and P. W. Shor. Algorithms for diametral pairs and convex hulls that are optimal, randomized, and incremental. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 12–17, 1988.

[15] A. Deza, M. Deza, and K. Fukuda. On Skeletons, Diameters and Volumes of Metric Polyhedra. Technical report, Laboratoire d'Informatique de l'Ecole Supérieure, Jan. 1996.

[16] M. Dyer. The complexity of vertex enumeration methods. *Math. Oper. Res.*, 8(3):381–402, 1983.

[17] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, chapter 8.4, pages 147–158. Number 10 in EATCS Monographs on Th. Comp. Sci. Springer-Verlag, 1987.

[18] K. Fukuda. cdd Reference manual, version 0.55. EPFL, Lausanne, Switzerland.

[19] K. Fukuda. mit71-61.ine solved. Private communication, Feb. 1996.

[20] T. Gal. Selected bibliography on degeneracy. *Annals of Operations Research*, 47:1–8, 1993.

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

[22] B. Grünbaum. *Convex Polytopes.* Interscience, London, 1967.

[23] M. Haiman. A simple and relatively efficient triangulation of the n-cube. *Discrete Comput. Geom.*, 6:287–289, 1991.

[24] T. 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.

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

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

[27] P. L'Ecuyer. Efficient and portable combined random number generators. *Communications of the ACM*, 31, 1988.

[28] J. Matoušek and O. Schwarzkopf. Linear optimization queries. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 16–25, 1992.

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

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

[31] J. S. Provan. Efficient enumeration of the vertices of polyhedra associated with network LP's. *Mathematical Programming*, 63:47–64, 1994.

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

[33] R. Seidel. The nature and meaning of perturbations in geometric computing. To appear Discrete and Compututational Geometry.

[34] R. Seidel. A convex hull algorithm optimal for point sets in even dimensions. Technical report, University of British Columbia, Dept. of Computer Science, 1981.

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

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

[37] G. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer, 1994.