How Good are Convex Hull Algorithms?*

David Avis David Bremner†

January, 1995

Abstract

A *convex polytope* is the bounded intersection of finite set $\mathcal{H}$ of halfspaces. A classic theorem of convexity theory is that every convex polyhedron can be expressed as the convex hull of its set $\mathcal{V}$ of vertices. There are three closely related computational problems related to the two descriptions of a polytope. The *vertex enumeration problem* is to compute $\mathcal{V}$ from $\mathcal{H}$. The *convex hull problem* it to compute $\mathcal{H}$ from $\mathcal{V}$. The *polytope verification problem* is to decide whether a given vertex description and halfspace description define the same polytope. The first two problems are essentially equivalent under point/hyperplane duality. It is an open problem whether any of these problems can be solved in time polynomial in $|\mathcal{H}| + |\mathcal{V}|$. In this paper we describe hard polytopes for convex hull algorithms based on *pivoting*, those based on *triangulation*, and for some *insertion* algorithms.

1 Introduction

Although the simplex method had long been regarded as a practical and efficient algorithm for linear programming, it was not until the seminal 1972 paper of Klee and Minty [13] that it was demonstrated to be an exponential time algorithm. The authors described a class of hard polytopes that cause the simplex method to make an exponential number of pivots using the greatest cost coefficient pivoting rule. Subsequent papers gave similar results for other pivot rules, and the search for a polynomial pivot rule (or the proof of its non-existence) still continues. In this paper we describe some hard polytopes for various convex hull algorithms. A d dimensional *convex polyhedron* is the intersection of a finite number m of non-redundant halfspaces $\mathcal{H}$ of $\Re^d$. A bounded convex polyhedron is called a *polytope*. A classic theorem from convexity is 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 *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. Each vertex of P lies on d affinely independent hyperplanes each of which bounds a halfspace in $\mathcal{H}$. Such a set of hyperplanes is called a *basis*. A polytope is *simple* if each of its vertices has a unique basis. There are three closely related computational problems concerning the two descriptions of a polytope:

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

*This research supported by NSERC Canada, and FCAR Québec.

†The authors are with the McGill University School of Computer Science. email: bremner@cs.mcgill.ca

It is an open problem whether any of these problems can be solved in time polynomial in $size(P)$. Since the first two problems are essentially equivalent under point/hyperplane duality, we will mainly restrict ourself to the vertex enumeration problem, although all results apply to both problems.

There are two main classes of algorithms for solving these problems: pivoting algorithms and insertion algorithms. A pivoting algorithm for the vertex enumeration problem begins by finding a basis of some vertex of P. Two bases of P are adjacent if they differ in one hyperplane only. This adjacency relation defines a graph on the bases of P, where each edge corresponds to a *pivot*, the replacement of exactly one hyperplane in a basis. All vertices of this graph are generated, from which the extreme points of P are readily computed. Representatives of this class are the *gift wrapping* algorithm of Chand and Kapur [5], Seidel's algorithm [19] and the *reverse search* algorithm of Avis and Fukuda [1]. For a simple polytope P, pivoting algorithms can solve all of the above problems in time polynomial in $size(P)$. For non-simple polytopes the number of bases may be exponential in the size of P. For such polytopes pivoting algorithms rely on perturbation, either symbolic or numerical, to reduce the computation time. In Section 3 we give a fairly general model for pivoting algorithms and valid perturbation schemes for polytopes. We then give examples of polytopes for which no valid perturbation scheme can yield polynomial time pivoting algorithms in this model.

Insertion algorithms for the vertex enumeration algorithm 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 are computed. Additional halfspaces are introduced sequentially and the vertex description is updated at each stage. 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. [17], 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 [18], the *randomized algorithm* of Clarkson and Shor [8] and the *derandomized algorithm* of Chazelle [6]. In some sense the algorithms [18] and [6] can be considered optimal. The upper bound theorem of McMullen states that for any polytope P defined by m halfspaces, $size(P) = O(m^{\lfloor d/2 \rfloor})$ and this bound is achieved (see, e.g. [3]). The algorithms [18] and [6] solve the vertex enumeration problem in this time bound. However, it is also well-known that $size(P) = \Omega(md)$ and this bound is also achieved. An efficient vertex enumeration algorithm for such polytopes should clearly be polynomial in md. It is not known whether such an algorithm exists. Examples that have arisen in practice (see e.g., [4]) suggest that the sizes of polytopes of interest are closer to the lower bound than the upper. Since there is such a wide variation in the output sizes of vertex enumeration problems, we suggest $size(P)$ as the appropriate measure of the problem size. The success of insertion algorithms is widely believed to depend on keeping the vertex complexity of the intermediate polytopes generated as small as possible by judicious choice of the order in which the halfspaces are inserted. While this is necessary for success of the algorithm it is not sufficient. The examples in Section 3 show that insertion algorithms which rely on triangulation (such as the the randomized and derandomized algorithms) can perform very badly indeed, even when the intermediate polytopes have small size and regardless of the order of insertion of the halfspaces.

For insertion algorithms that do not rely on triangulation, the choice of insertion order is indeed a critical factor. In Section 4 we give an example of a class $\mathcal{P}$ of polytopes for which any insertion algorithm that inserts halfspaces in a fixed order (independent of the input) or in lexicographic order must construct exponential sized intermediate polytopes in the worst case. In Section 5 we present experimental evidence that several popular heuristic insertion orders are also exponential for the class $\mathcal{P}$, but that there is a polynomial insertion order for the double description method for this class.

The examples of hard polytopes given in Sections 3 and 4 apply to computational models that include many published convex hull 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 in Section 5 of the paper we give actual computational experience obtained by trying to solve the vertex enumeration and convex hull problems for these hard polytopes.

2 Preliminaries

Given a set of points $X = \{ x_1, \ldots x_k \}$, y is called an *affine combination* of X if

$$y = \sum_{i=1}^{k} \lambda_i x_i \quad \text{and} \quad \sum_{i=1}^{k} \lambda_i = 1.$$

The *affine hull* of a set of points X (written $\mathrm{aff}(X)$) is the set of all affine combinations of X. The set X is called *affinely independent* if no x_i is an affine combination of the other $k-1$ points in X. A *convex of combination* of X is an affine combination such each λ_i is non-negative. A *proper* convex combination is one where each λ_i is positive. A subset K of $\Re^d$ is called a convex set if every convex combination of points in K is also in K. The *convex hull* of X, denoted $\mathrm{conv}(X)$, is the set of all convex combinations of X. The relative interior of a convex set K, written $\mathrm{relint}(K)$, is the interior of K in the affine hull of K.

Lemma 1 (Theorem 3.3 [3]) *For any convex set C, any point x in* $\mathrm{relint}(C)$*, and any point $y \in C$, the open line segment from x to y is contained in* $\mathrm{relint}(C)$.

Lemma 2 (Theorem 3.5 [3]) *For any convex set C and any point $x \in C$, the following conditions are equivalent:*

(a) $x \in \mathrm{relint}(C)$.

(b) For any $y \in C$ distinct from x, there exists a $z \in C$ such that x is a proper convex combination of y and z; i.e. any segment $\overline{yx}$ can be extended beyond x in C.

A d dimensional *convex polyhedron* is the intersection of a finite number of halfspaces of $\Re^d$. A bounded convex polyhedron is called a *convex polytope*. A halfspace H is called a *valid constraint* for convex polyhedron P if $P \subseteq H$. Given a set of halfspaces $\mathcal{H}$, $H \in \mathcal{H}$ is is called *non-redundant* if there is some point feasible for $\mathcal{H} \setminus \{ H \}$ but not feasible for $\mathcal{H}$. We use $\mathcal{H}(P)$ to denote the non-redundant halfspace description of P and $\mathcal{V}(P)$ to denote the vertex description of P. We abbreviate a d dimensional convex polyhedron (polytope) to a d-polyhedron (d-polytope). Let P be a d-polytope, containing the origin in its interior, with vertices $\{ p_1, p_2, \ldots, p_n \}$ and defining halfspaces $\{ a_i x \leq 1 \mid i = 1, \ldots, m \}$. The *dual* of P is the polytope with vertices $\{ a_1, a_2, \ldots, a_m \}$ and is defined by the intersection of the halfspaces $\{ p_j x \leq 1 \mid j = 1, \ldots, n \}$. It is well known and easy to show that the vertex enumeration problem for P is identical to the convex hull problem for the dual of P, and vice versa.

Given a convex polytope P, a point $x \in P$ is an *extreme* point of P if it is not a proper convex combination of any two points in P. The set of extreme points of a polyhedron P is written ext(P). A *supporting halfspace* of a convex polyhedron P is a halfspace H with bounding hyperplane h (called a *supporting hyperplane*) such that $P \subseteq H$ and $h \cap P \neq \emptyset$. A *face* of a convex polyhedron P is the intersection of P and one or more supporting hyperplanes of P. We abbreviate a k dimensional face to a k-face. The 0, 1, $d-2$ and $d-1$ dimensional faces of a d-polyhedron P are called the *vertices*, *edges*, *ridges*, and *facets* of P. Given a facet F of a polytope P, F^+ denotes the closed halfspace induced by aff(F) containing P and F^- the other closed halfspace induced by aff(F). We denote the interior of P as int(P).

Let the *centroid* of a set of points $\{ x_1, \ldots x_k \}$ denote the point

$$\frac{1}{k} \sum_{i=1}^{k} x_k.$$

Define the centroid of a convex polytope as the centroid of its vertices. Observe that the centroid a set X of points is a proper convex combination, hence by Lemma 1 is contained in the relative interior of conv(X).

The convex hull of a set of $d+1$ affinely independent points in d dimensions is called a *simplex*.

Lemma 3 (Theorem 4.1.2 [11]) *Any d vertices of a d-simplex P determine a facet of P.*

Given a polyhedron P, a hyperplane h such that $h \cap \text{int}(P) = \emptyset$, and a point x, we say that x is *beneath* (respectively *beyond*) h if x is in the halfspace induced by h containing int(P) (respectively not containing int(P)). The point x is *strictly* beneath (respectively *strictly beyond*) h if it is beneath h (beyond h) but not in h. A hyperplane h *strictly separates* two points if one is in one open halfspace defined by h and one is in the other. We take beneath or beyond a facet F to mean beneath or beyond aff(F). If P^* is the convex hull of $P \cup \{ x \}$ where x is beyond exactly one facet of P, we say that P^* is a *stellation* of P. The following lemma characterizes the structure of the result of stellating a facet of a polytope.

Lemma 4 (Theorem 5.2.1 [11]) *Let P be a polytope. Let x be a point that is strictly beyond exactly one facet F^* of P and strictly beneath every other facet of P. Let $P' = \text{conv}(P \cup \{ x \})$ be a stellation of P. The set F' is a face of P' if and only if one of the following holds.*

(a) F' is a face of P other than F^*, or

(b) $F' = \text{conv}(F \cup \{ x \})$ for some face F of P strictly contained in F^*

3 Perturbation, Triangulation and Pivoting

Let P be a d-polytope with m facets and n vertices. A pivoting algorithm for the vertex enumeration problem starts at some basis for a vertex of P and generates all other feasible bases for P. From these bases the vertices of P are obtained, with repetitions if P is non-simple. This is efficient for simple polytopes, but can be extremely inefficient for polytopes with one or more highly degenerate vertices. In the worst case P may have a single vertex and as many as $\binom{m}{d}$ bases. Even though this is an extreme case, highly degenerate polytopes seem to arise in practice and generating all of the bases is impractical. To reduce the computation, P is perturbed to a simple polytope $\tilde{P}$ with m facets and $\tilde{n}$ vertices, where hopefully $\tilde{n}$ is much smaller than the number of bases of P. The vertices of the simple polytope $\tilde{P}$ are computed from which the vertices of the original polytope P are obtained. Let $\mathcal{B}$ and $\tilde{\mathcal{B}}$ denote the sets of bases of P and $\tilde{P}$ respectively. We call a perturbation *valid* if the following conditions are satisfied:

(i) $\tilde{\mathcal{B}} \subseteq \mathcal{B}$.

(ii) Each vertex of P has a basis B contained in $\tilde{\mathcal{B}}$.

It is clear that under the above conditions, all vertices of P can be generated as follows. Generate $\tilde{\mathcal{B}}$ and list all vertices in P corresponding to these bases, removing duplicates. If $\tilde{P}$ is simple then the number of bases is at most $O(m^{\lfloor d/2 \rfloor})$, by the upper bound theorem. A perturbation scheme can be either numerical or symbolic, and general framework for describing such schemes is contained in Yap [20]. A common numerical perturbation is obtained by perturbing the right hand side vector of the system of halfspaces defining P. This perturbed vector is used in calculations. A simple way of transforming the perturbed vertices back to vertices of P is simply to carry an extra column which is the original right hand side vector. This is pivoted along with the rest of the constraint matrix but does not play any role in the calculations. Instead of outputting the perturbed right hand side (corresponding to a vertex of $\tilde{P}$) this additional column is output (corresponding to a vertex of P); duplicates can be removed off-line.

In practice, perturbation has been extremely useful. In Ceder et al. [4], the 4862 vertices of a highly degenerate 8-dimensional polytope defined by 729 halfspaces were computed in exact arithmetic by the reverse search pivoting scheme. In this case, computing all the bases of the original polytope was infeasible, and even the perturbed polytope had 477,421 bases.

The effectiveness of a perturbation scheme obviously lies in the ability to generate a simple polytope $\tilde{P}$ with as few vertices as possible. The number of vertices in the perturbed polytope is therefore a natural measure of the quality of the perturbation. This measure of the quality of a perturbation does not seem to have been studied in the literature. The bound on the number of bases of the perturbed polytope given above assumes the worst case that P is perturbed onto the dual of the cyclic polytope. On the other hand, if a valid perturbation onto a lower bound polytope is possible, we would get very good results indeed. Here we answer a fundamental question: Is it always possible to find a perturbation such that the number of vertices in $\tilde{P}$ is polynomial in $size(P)$? A positive answer would lead to a polynomial time algorithm for the problems described in Section 1. Unfortunately, the answer is negative.

In order to see why a good perturbation is sometimes impossible, consider the following example suggested by Günter Rote. It is convenient here to consider the convex hull problem rather than the vertex enumeration problem. The input is the set of 2^d vertices of a d dimensional hypercube H_d, and the output is a list of its $2d$ facets. A basis of a facet is a full dimensional simplex spanned by $d+1$ of the 2^{d-1} vertices defining the facet. For the convex hull problem, a perturbation is applied to the input points to give a simplicial polytope, i.e. a polytope for which each facet has a unique basis. It can be checked that the criteria (i) and (ii) above for a valid perturbation dualize into the condition that the facets of the convex hull of the perturbed point set induce a triangulation of the facets of the unperturbed hypercube. The duality between triangulations and perturbations was previously observed by Lee [16]. Note that the facets of the hypercube are $(d-1)$-cubes. It is well known (see for example [12]) that every triangulation of H_d requires at least $\Omega(\sqrt{d!})$ simplices. Now $size(H_d) = d^2 2^{d+1}$, and hence the number of simplices required to triangulate the facets of a H_d is super polynomial in its size. Therefore there is no valid perturbation scheme that yields a perturbed point set with polynomial size. Dualizing the argument, we conclude that the halfspaces defining a d-cross polytope cannot be perturbed into a simple polytope with size polynomial in the size of the cross-polytope. Therefore pivoting algorithms for the vertex enumeration problem that rely on perturbation are super polynomial in the worst case.

Although the above argument gives a non-polynomial lower bound, the bound is only just super polynomial. If $N = size(H_d)$, then the number of pivots is $\Omega(e^{log^2 N})$. The following polytopes were suggested by Bernd Sturmfels and give sharper lower bounds. Let w be a d-vector of all ones and for $1 \le i \le d$, let e_i denote the d-vector which is all zero except for $d+1$ in position i. We denote by T_d the simplex in $\Re^d$ spanned by the vertices $\{ e_1 - w, e_2 - w, \ldots, e_d - w, -w \}$. Note that the centroid of T_d is the origin. In $\Re^{s+t}$ we construct the product $T_s \times T_t$ of the two simplices T_s and T_t in orthogonal $\Re^s$ and $\Re^t$ subspaces. The product contains the $(s+1)(t+1)$ vertices formed by concatenating each vertex of T_s with each vertex of T_t. The following results are known about these polytopes [10]:

(a) $T_d \times T_d$ has $n = (d+1)^2$ vertices and $m = 2d+2$ facets.

(b) Each triangulation of $T_s \times T_t$ requires exactly $\binom{s+t+2}{s+1}$ simplices.

(c) Each facet of $T_d \times T_d$ is combinatorially equivalent to $T_{d-1} \times T_d$.

From these facts we obtain the following result.

Theorem 1 *The polytope $T_d \times T_d$ has size $N = 2d(d^2+4d+3)$ and every valid perturbation yields a simple polytope of size $\Omega(2^{\sqrt[3]{N}})$.*

This theorem implies that solving the convex hull problem for $T_d \times T_d$ by a pivoting algorithm using perturbation requires the generation of at least $\Omega(2^{\sqrt[3]{N}})$ bases, where N is the size of the polytope. The bounds also apply to insertion algorithms that construct triangulations of the polytope, or of the facets of the polytope, such as the randomized [8] and derandomized [6] algorithms. The bound does not, however, apply to the double description method [17], which seems empirically to run in polynomial time on these polytopes. A similar result applies to the vertex enumeration problem for the dual of $T_d \times T_d$. In Section 5 we illustrate these results in practice on currently available implementations of these algorithms.

To conclude this section we observe that a similar argument can be applied to another technique which is frequently suggested for dealing with degeneracy: truncating degenerate vertices. Here the idea is to introduce an additional halfspace which contains in its interior all of the vertices of the polytope, except some known degenerate vertex. All vertices of this truncated polytope are enumerated, and those lying on the cutting hyperplane are associated with the original degenerate vertex. Such a technique can be applied on-line whenever a degenerate vertex is encountered. Arguments similar to those above again show this cannot lead to a polynomial time algorithm. As above, consider the dual convex hull problem. The cutting halfspace dualizes into the addition of a new point above a non-simplicial facet. The lower bounds on triangulation depend on volume arguments that still apply in this new setting, giving super polynomial lower bounds similar to those above.

4 Insertion Algorithms and Intermediate Size

An insertion algorithm for vertex enumeration computes the vertices of $k \equiv |\mathcal{H}| - d$ intermediate polytopes, where $\mathcal{H}$ is the input set of halfspaces. Initially a simplex P_0 is constructed, using $d+1$ elements of $\mathcal{H}$. At each succeeding step, a halfspace $H \in \mathcal{H}$ is chosen according to some *insertion order* and the vertices of $P_i \equiv P_{i-1} \cap H$ are computed. The final output is the vertices of $P_k \equiv \bigcap_{H \in \mathcal{H}} H$. In order for such an algorithm to be polynomial in $size(P_k)$, the size of each of these intermediate polytopes must be polynomial in $size(P_k)$. In this section, we give examples of a polytope family and insertion orders for which this not the case.

The general outline of our construction is as follows. For input size n, we construct a hard polytope of size n as follows. We start with a polytope Q_n with vertex complexity exponential in n and facet complexity polynomial in n. We then construct a polytope P_n such that

1. $\mathcal{H}(P_n \cap Q_n) = \mathcal{H}(P_n) \cup \mathcal{H}(Q_n)$; i.e. no constraint of P_n or Q_n is redundant for $P_n \cap Q_n$, and

2. $P_n \cap Q_n$ has vertex complexity polynomial in n.

Our hard polytope will be $P_n \cap Q_n$. The lower bounds will result from forcing certain insertion orders to build the entire polytope Q_n before inserting any constraints of P_n. In Section 4.1 we derive general conditions under which this is true. In Section 4.2 we give examples of polytope families P_n and Q_n for which these general conditions hold.

4.1 Pairs of Piercing Polytopes

In this subsection we define a set of general conditions on a pair of polytopes P and Q such that for any pair of polytopes that satisfies these conditions, $size(P \cap Q)$ is polynomial in $|\mathcal{V}(P)| + |\mathcal{H}(Q)|$, and $\mathcal{H}(P \cap Q) = \mathcal{H}(P) \cup \mathcal{H}(Q)$.

The edges and vertices of a polytope P define an undirected graph in an obvious way. Denote this graph $\mathcal{G}(P)$. Let V be a subset of the vertices of a polytope P: $\mathcal{G}(P \mid V)$ denotes the subgraph of $\mathcal{G}(P)$ induced by V.

Lemma 5 *Let P be a polytope with vertices $\mathcal{V}$ and halfspace description $\mathcal{H}$. Let H be a halfspace. Let $\mathcal{V}_H$ denote $\mathcal{V} \cap \operatorname{int}(H)$. The graph $\mathcal{G}(P \mid V_H)$ is connected.*

Proof: Define a linear program with the constraints $\mathcal{H} \cup \{ H \}$ and an objective function of the inward normal of H. Every vertex of $P \cap H$ is either in $\mathcal{V}_H$ or contained in the bounding hyperplane of H. From the correctness of the simplex method with Bland's pivot rule (see e.g. [7]), there is a path in $\mathcal{G}(P \cap H)$ from any vertex of $\mathcal{V}_H$ to a unique optimum vertex of $P \cap H$. Since the simplex method monotonically increases the value of the objective function (i.e. the distance from the bounding hyperplane of H), this path does not intersect the bounding hyperplane of H, hence is entirely contained in $\mathcal{G}(P \mid \mathcal{V}_H)$. ■

Given polytopes P and Q, a *connected component* of P with respect to Q is a maximal subset C of vertices of P such such such that $\mathcal{G}(P \mid C)$ is connected and no edge of P between vertices of C intersects Q. A connected component C of P w.r.t. Q *properly pierces* a facet F of Q if $\operatorname{aff}(F)$ strictly separates C from $\mathcal{V}(P) \setminus C$ and and $P \cap \operatorname{aff}(F) \subset \operatorname{relint}(F)$. The *proper intersection* of polytopes P and Q denotes $\operatorname{relint}(P) \cap \operatorname{relint}(Q)$.

Theorem 2 *Let P and Q be d-polytopes such that $P \cap Q \neq \emptyset$ and $\mathcal{V}(P) \cap Q = \emptyset$. The following conditions are equivalent.*

(a) Every connected component of P w.r.t. Q properly pierces some facet of Q.

(b) No ridge of Q intersects P.

(c) A point x is a vertex of $P \cap Q$ iff it is the proper intersection of an edge of P with a facet of Q.

Proof:

(a⇒b) Suppose (a) holds. Let F be a facet of Q. If F is properly pierced by a connected component of P w.r.t. Q, then by definition no ridge of Q contained in F intersects P. Otherwise let $\mathcal{V}_F$ be the subset of $\mathcal{V}(P)$ strictly beyond F. By Lemma 5, $\mathcal{G}(P \mid \mathcal{V}_F)$ is connected. By convexity, no edge with both endpoints in $\mathrm{int}(F^-)$ can intersect Q. It follows that $\mathcal{V}_F$ is contained in some connected component C' of P w.r.t. Q. From (a), C' must properly pierces some facet F' of Q distinct from F.

Let E_F denote the intersection of the edges of P with the affine hull of F. We know the following:

$$\begin{aligned} P \cap F &= (P \cap \mathrm{aff}(F)) \cap F \\ &= \mathrm{conv}(\mathrm{ext}(P \cap \mathrm{aff}(F))) \cap F \\ &= \mathrm{conv}(E_F) \cap F \end{aligned}$$

We argue that $\mathrm{aff}(F')$ is a strict separating hyperplane for E_F and F. Each point in E_F is contained in the relative interior of a line segment with one endpoint in C' and one endpoint in $\mathrm{relint}(F')$. The hyperplane $\mathrm{aff}(F')$ strictly separates C' from Q; if it did not, there would be a vertex v of C' in F^+. By the connectedness of $\mathcal{G}(P \mid C')$, there would have to be some edge connecting the subset of C' strictly beyond $\mathrm{aff}(F')$ with the subset of C' containing v. But such an edge would have to intersect $\mathrm{relint}(F')$, contradicting the fact that v was in the connected component C'. By Lemma 1 any point in the relative interior of a line segment with one endpoint in $\mathrm{aff}(F')$ and the other strictly beyond $\mathrm{aff}(F')$ must be strictly beyond $\mathrm{aff}(F')$. It follows that $\mathrm{aff}(F')$ strictly separates E_F from F, hence $P \cap F = \emptyset$.

(a⇐b) Suppose (b) holds. Let C be a connected component of P w.r.t. Q. Let z be a point in $P \cap Q$. Let v be a vertex of C. Let F be the facet of Q intersected by segment $\overline{vz}$ (the uniqueness of F follows from (b)). We argue that C properly pierces F.

First, we argue that $\mathrm{aff}(F)$ strictly separates C from $\mathcal{V}(P) \setminus C$. By Lemma 5, any vertex of P strictly beyond $\mathrm{aff}(F)$ must be in C. Suppose there were some vertex v_1 of C in F^+. Let x denote $\overline{vz} \cap F$. By convexity, $x \in P$. From (b), $x \in \mathrm{relint}(F)$. From the connectedness of $\mathcal{G}(P \mid C)$, there must be some edge e of P that intersects $\mathrm{aff}(F)$ but does not intersect Q. Let y denote $e \cap \mathrm{aff}(F)$. By convexity, $y \in P$ and $\overline{xy} \subset P$. Since x and y are both in the affine hull of F, but one is in the relative interior of F, and one completely outside it, the segment $\overline{xy}$ must intersect some ridge of Q. Since this is a contradiction, every vertex of C is strictly beyond F. Since every vertex of P strictly beyond F is in C and every vertex of C is strictly beyond F, $\mathrm{aff}(F)$ properly separates C from $\mathcal{V}(P) \setminus C$.

Next we argue that $P \cap F \subset \mathrm{relint}(F)$. Consider an edge e' with exactly one endpoint C that intersects P. The edge e' must intersect $\mathrm{aff}(F)$ since $\mathrm{aff}(F)$ is a separating hyperplane for C and P. If e' does not intersect $\mathrm{relint}(F)$ then we again have a line segment $\overline{xy} \subset P$ with $x \in \mathrm{relint}(F)$ and $y \in \mathrm{aff}(F) \setminus \mathrm{relint}(F)$, contradicting (b).

(b⇒c) Every vertex of $P \cap Q$ must be the intersection of at least d supporting hyperplanes of P or Q. Every intersection of 2 or more supporting hyperplanes of Q must be part of some ridge of Q, hence Q can contribute at most one hyperplane to a vertex of $P \cap Q$. Any intersection of d supporting hyperplanes of P is a vertex of P, hence Q must contribute at least one hyperplane to a vertex of $P \cap Q$. It follows that every vertex of $P \cap Q$ is the proper intersection of an edge of P and a facet of Q.

Suppose there were an edge e of P and a facet F of P such that $\text{relint}(e) \cap \text{relint}(F) \neq \emptyset$ but $x \equiv \text{relint}(e) \cap \text{relint}(F)$ is not a vertex of $P \cap Q$. In order for this to be the case x must be infeasible for $P \cap Q$; but this is impossible since $x \in e \subset P$ and $x \in F \subset Q$.

(b⇐c) We prove the contrapositive. Suppose some ridge r of Q has non-empty intersection with P. The vertices of $r \cap Q$ must be vertices of $P \cap Q$, but again cannot be proper intersections of edges of P and facets of Q. ∎

If the conditions of Theorem 2 hold for polytopes P and Q, we say that P *pierces* Q.

Corollary 6 *If d-polytope P pierces d-polytope Q then*

$$f_0(P \cap Q) \leq 2f_1(P).$$

Corollary 7 *If d-polytope P pierces d-polytope Q and every connected component of P w.r.t. Q has precisely one vertex, then*

$$f_0(P \cap Q) = 2f_1(P).$$

The following technical lemma is useful in establishing that no constraint in $\mathcal{H}(P) \cup \mathcal{H}(Q)$ is redundant for the polytope defined by $P \cap Q$.

Lemma 8 *Let P be a d-polytope and F a facet of P. Let K be a convex set. If* $\text{relint}(F) \cap \text{relint}(K) \neq \emptyset$ *and* $K \not\subseteq \text{aff}(F)$ *then* $\text{relint}(K) \cap \text{int}(P) \neq \emptyset$.

Proof: Let x be a point in $\text{relint}(F) \cap \text{relint}(K)$. The point x strictly satisfies every constraint of $\mathcal{H}(P)$ other than F^+; it follows that for some positive ϵ there is an ϵ-radius d-ball B centered at x such that any point in B also strictly satisfies every constraint of $\mathcal{H}(P)$ other than F^+.

There must be some point z in $K \setminus \text{aff}(F)$ such that z is strictly beneath F. This can be seen as follows. Let y be a point in $K \setminus \text{aff}(F)$; y is either strictly beneath F, in which case we are done, or strictly beyond $\text{aff}(F)$. Suppose y is beyond F. By Lemma 2, there exists a point $z \in K$ such that x is a proper convex combination of y and z. This implies that z is strictly beneath F.

Let w be the intersection of the line segment $\overline{xz}$ and the boundary of B (i.e. the point on $\overline{xz}$ at distance ϵ from x). By Lemma 1, w is in $\text{relint}(K)$. By the convexity of F^+, w is strictly beneath $\text{aff}(F)$; from the definition of B, w is strictly beneath every other $H \in \mathcal{H}(P)$. It follows that $w \in \text{relint}(K) \cap \text{int}(P)$. ∎

If d-polytope P completely pierces d-polytope Q and every facet of Q is properly pierced by some component of P, we say that P *completely pierces* Q.

Theorem 3 *If d-polytope P completely pierces d-polytope Q and every connected component of P w.r.t. Q has exactly one vertex, then*

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

Proof: Let F_p be a facet of P. Let v be a vertex of F_p. By Theorem 2, there exists some facet F_q of Q such that $\text{aff}(F_q)$ separates v from $\mathcal{V}(F_p) \setminus \{v\}$ and $F_p \cap \text{aff}(F_q) \subset \text{relint}(F_q)$. The facet F_p has at least two vertices so F_p must intersect $\text{aff}(F_q)$. Let F' denote $F_p \cap \text{aff}(F_q)$. Suppose $F' \cap \text{relint}(F_p) = \emptyset$; it follows that $\text{aff}(F_q)$ is a supporting hyperplane for F_p, but this is a contradiction. It follows that $\text{relint}(F_p) \cap \text{relint}(F_q) \neq \emptyset$. By Lemma 8, the relative interior of F_p

intersects the interior of Q. This means that there is a point that strictly satisfies every constraint of $(\mathcal{H}(Q) \cup \mathcal{H}(P)) \setminus \{ F_p^+ \}$ and satisfies F_p^+ with equality, which implies that F_p^+ is not redundant.

Since every facet F_q of Q is properly pierced by some connected component of P w.r.t. Q, the above argument implies that F_q^+ is not redundant either. ■

4.2 Truncated Cubes

We now give examples of polytope families that satisfy the conditions established in Section 4.1. We start with a scaled d-cube C_d. We then construct a simplicial polytope S_d with $2d$ vertices and $d^2 - d + 2$ facets that pierces each face of C_d. Our final construction is then the intersection of S_d and C_d, which we will show has $d^2 + d + 2$ facets and $3d^2 - d$ vertices ($d \geq 3$). The lower bound argument is based on showing that for some insertion orders, an insertion algorithm must compute all 2^d vertices of C_d.

We wish to construct a class of polytope whose facet and vertex complexity is small in terms of the dimension, but that cuts off all of the vertices of a scaled d-cube; our class will be *stacked* polytopes like those used in the proof of the Lower Bound Theorem. A polytope is called *stacked* if it is the stellation of a d-simplex, or the stellation of stacked polytope. Let $P_{n,d}$ be a stacked d-polytope with n vertices. Let $f_j(P)$ denote the number of j-dimensional faces of a polytope P. The following is known about the face complexity of a stacked polytope (see e.g. [3]).

$$f_j(P_{n,d}) = \begin{cases} \binom{d}{d-j} n - \binom{d+1}{d-j} j, & \text{if } 0 \leq j \leq d-2; \\ (d-1)n - (d+1)(d-2), & \text{if } j = d-1 \end{cases} \tag{1}$$

We define the d-polytope S_d as the convex hull of $d+1$ *simplex vertices* t_i with coordinates:

$$t_{ij} = \begin{cases} \;\;1 - d^{-1} & \text{if } 1 \leq i < d \text{ and } j = i; \\ \;\;1 & \text{if } i = j = d; \\ -1 & \text{if } i = d+1 \text{ and } j = d; \\ \quad\; - d^{-1} & \text{otherwise} \end{cases}$$

and $d-1$ *pyramid vertices* p_i with coordinates

$$p_{ij} = \begin{cases} -\frac{3}{2d}, & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases}$$

In the rest of this section in order to simplify the analysis we assume that $d \geq 3$. We now show how that S_d is a stacked polytope.

For any dimension d, we define a simplex T_d, with vertices $\{ t_i \mid 1 \leq i \leq d+1 \}$. Since the points t_i are affinely independent, T_d is indeed a simplex; by Lemma 3 any d vertices of T_d define a facet. Let F_i, $1 \leq i \leq d+1$ be the simplex facet that does not contain t_i. There are two kinds of facets, those that contain both t_d and t_{d+1}, and those contain only one. By analogy with the three dimensional case, we call the vertices t_d and t_{d+1} *spine* vertices, and those faces that contain both spine vertices *vertical* faces.

Observe that every coordinate except coordinate i of $\sum_{j \neq i} t_j$ sums to 0, and coordinate i sums to -1. It follows that the centroid of vertical facet F_i has coordinate i equal to $-d^{-1}$ and every other coordinate 0. Similarly, the centroid of T_d is the origin. Since the centroid is a proper convex combination, the origin is in the interior of T_d. Any polytope with the origin in its interior has a halfspace representation

$$P = \{ x \mid Wx \leq 1 \}$$

The supporting halfspace of a facet with vertices $\{ v_1, v_2, \dots v_d \}$ is given by $\{ x \mid wx \leq 1 \}$ where w is the solution to $Aw = 1$ and A is the matrix with row i equal to v_i. Let H_i be the supporting halfspace for F_i. For simplex facet F_i, the matrix A will have following form:

$$A_i = \begin{bmatrix} 1-d^{-1} & -d^{-1} & & \cdots & -d^{-1} & 0 \\ -d^{-1} & 1-d^{-1} & & \cdots & -d^{-1} & 0 \\ & & \vdots & & & \\ & & t_{i-1} & & & \\ & & t_{i+1} & & & \\ & & \vdots & & & \\ -d^{-1} & \dots & 1-d^{-1} & & -d^{-1} & 0 \\ -d^{-1} & \dots & -d^{-1} & & 1-d^{-1} & 0 \\ -d^{-1} & \dots & -d^{-1} & & -d^{-1} & 1 \\ -d^{-1} & \dots & -d^{-1} & & -d^{-1} & -1 \end{bmatrix}$$

Before computing particular facets, we make some general observations about linear combinations of simplex vertices. In the following, we refer the row of A_i containing vertex t_i as row t_i.

- For any vertical facet (one that contains both spine vertices), we can subtract row t_{d+1} from row t_d to obtain
$$w_d = 0. \tag{2}$$
- For any vertical facet we can add row t_d and row t_{d+1} to obtain
$$\sum_{i=1}^{d-1} w_i = -d. \tag{3}$$
- If we consider any two rows corresponding to non-spine vertices t_i and t_j $1 \leq i < j < d$, by subtracting row j from row i we get
$$w_i = w_j. \tag{4}$$

Consider a vertical facet F_j. Since F_j contains every simplex vertex except t_j, every entry in column j of A_j is $-d^{-1}$. Every other column sums to zero, so

$$w_j = -d.$$

Combining this with (2), (3), and the fact that $\mathrm{o} \in \mathrm{int}(T_d)$, we get

$$H_j = \{ x \mid x_j \geq -\frac{1}{d} \} \qquad \text{for } 1 \leq j < d$$

Consider F_{d+1}. Since the facet goes through t_d, for any non-spine simplex vertex t_i contained in F_{d+1},

$$w_i = w_d.$$

Combining this with row t_d,

$$\begin{aligned} w_d - \frac{d-1}{d} w_d &= 1 \\ w_d &= d \\ H_{d+1} &= \{ x \mid x_d + \sum_{i=1}^{d-1} x_i \leq \frac{1}{d} \}. \end{aligned}$$

Similarly, for F_d, since the facet goes through t_{d+1}, $w_i = -w_d$, $w_d = -d$,

$$H_d = \{ x \mid x_d - \sum_{i=1}^{d-1} x_i \geq -\frac{1}{d} \}.$$

We now stack vertices above the centroid of each vertical face. We lift these vertices a distance of $1/2d$ beyond the facet along the normal to the facet. Each of these *pyramid* vertices then has j-th coordinate

$$p_{ij} = \begin{cases} -\frac{3}{2d}, & \text{if } i = j; \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

We define a sequence of polytopes $T_d \equiv S(d,0), S(d,1), \ldots, S(d,d-1) \equiv S_d$, where $S(d,k)$ is defined as convex hull of $S(d,k-1) \cup \{ p_k \}$, $k = 1, \ldots d-1$. By Lemma 4 every facet of $S(d,k)$ is either a facet of T_d or the convex hull of pyramid vertex p_k and some $(d-2)$-face of vertical face F_k. We call the latter *pyramid* facets. Each $(d-2)$-face of T_d is a $(d-2)$-simplex; thus each pyramid facet adjacent to p_k contains all of the vertices F_k but one. Let F_{ki} denote the pyramid facet adjacent to p_k not containing t_i. As above, we can solve for the equations of the pyramid facets by solving $A_{ki}w = 1$ for A_{ki} equal to the matrix formed by taking the vertices of F_{ki} as rows. We again consider two cases, the vertical pyramid faces that contain both t_d and t_{d+1} and the non-vertical pyramid faces that contain only one of t_d and t_{d+1}. We can solve immediately for w_k since every coordinate of the pyramid vertices but one is zero. From (5) for any pyramid facet F_{ki},

$$w_k = -\frac{2d}{3}. \tag{6}$$

For vertical pyramid facets, combining this with (2), (4) and (3), we get

$$H_{kj} = \{ x \mid x_j + 2x_k \geq -\frac{3}{d} \}$$

For non-vertical pyramid facets through t_d, combining t_d with (4) and (6), we get

$$\begin{aligned} w_d - \frac{d-1}{d} w_d + \frac{2d}{3} \times \frac{1}{d} &= 1 \\ w_d &= \frac{d}{6} \end{aligned}$$

Along with (4), this yields

$$H_{k(d+1)} = \{ x \mid \sum_{i \neq k} x_i - 4x_k \leq \frac{6}{d} \}$$

A similar argument yields

$$H_{kd} = \{ x \mid x_d - \sum_{i \notin \{ k,d \}} x_i + 4x_k \geq -\frac{6}{d} \}$$

Lemma 9 *S_d is a stacked d-polytope with $2d$ vertices, $d^2 - d + 2$ facets and the following halfspace description:*

$$\begin{array}{lcll} H_{kj} & = & \{ x \mid x_j + 2x_k \geq -\frac{3}{d} \} & 1 \leq k < d, \quad j \notin \{ k,d \} \\ H_{kd} & = & \{ x \mid x_d - \sum_{i \notin \{ k,d \}} x_i + 4x_k \geq -\frac{6}{d} \} & 1 \leq k < d \\ H_{k(d+1)} & = & \{ x \mid \sum_{i \neq k} x_i - 4x_k \leq \frac{6}{d} \} & 1 \leq k < d \\ H_d & = & \{ x \mid x_d - \sum_{i=1}^{d-1} x_i \geq -\frac{1}{d} \} & \\ H_{d+1} & = & \{ x \mid x_d + \sum_{i=1}^{d-1} x_i \leq \frac{1}{d} \} & \end{array}$$

Proof: Consider the sequence of polytopes $S(d,0)\ldots S(d,d-1)$ defined above. In order for $S_d = S(d,d-1)$ to be a stacked polytope, each $S(d,k)$, $k=1,\ldots,d-1$ must be a stellation of $S(d,k-1)$. Each pyramid vertex p_k is by definition beyond vertical facet F_k. Every other facet of $S(d,k-1)$ is either a facet of T_d or a pyramid facet F_{ij}, $i \neq k$. Pyramid vertex p_k is beneath F_{ij} since

$$p_{kj} + 2p_{ki} = \begin{cases} -\frac{3}{2d}, & \text{if } j = k; \\ 0, & \text{otherwise.} \end{cases}$$

Vertex p_k is also beneath H_{kd} and $H_{k,d+1}$ since $\sum_{j \neq i} p_{kj} = -\frac{3}{2d}$ and $p_{ki} = 0$. Similar substitutions show that p_k is also beneath every simplex facet other than F_k. ■

We now consider the intersection of the stacked polytope S_d with a scaled d-cube. Let C_d denote the d dimensional parallelepiped defined by the following constraints.

$$\begin{array}{lcll} L_i & = & \{\, x \mid x_i \geq -\frac{5}{4d} \,\} & \text{for } 1 \leq i < d \\ U_i & = & \{\, x \mid x_i \leq 1 - \frac{5}{4d} \,\} & \text{for } 1 \leq i < d \\ L_d & = & \{\, x \mid x_d \leq 1 - \frac{1}{4d} \,\} & \\ U_d & = & \{\, x \mid x_d \geq -1 + \frac{1}{4d} \,\} & \end{array}$$

Let K_d denote $S_d \cap C_d$. Figure 1 shows K_3.

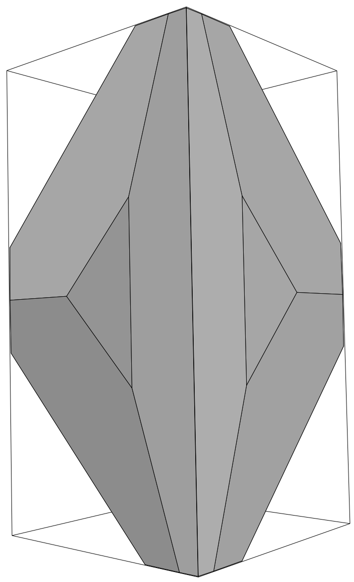

Figure 1: K_3

Theorem 4 *For $d \geq 3$,*

(a) $f_0(K_d) = 2f_1(S_d) = 3d^2 - d$

(b) $f_{d-1}(K_d) = f_{d-1}(S_d) + f_{d-1}(C_d) = d^2 + d + 2$

In order to prove this theorem, we establish two lemmas that will show that S_d completely pierces C_d, and that every connected component of S_d w.r.t. C_d consists of precisely one vertex of S_d. Part (a) of the theorem will then follow from Corollary 7 and part (b) will follow from Theorem 3.

Lemma 10 *For every vertex v of S_d, there exists a facet F of C_d, and for every facet F of C_d, there exists a vertex v of S_d such that:* $\mathrm{aff}(F)$ *strictly separates v from $\mathcal{V}(S_d) \setminus \{ v \}$.*

Proof: We consider three classes of vertices of S_d, and show that vertices of each type are strictly separated from the rest of the vertices of S_d by the affine hull of some facet of S_d.

$\{ p_k \mid 1 \le k < d \}$ If v is pyramid vertex p_k, v violates constraint L_k. Every other pyramid vertex p_j has coordinate k equal to 0, so strictly satisfies L_k. Since $k < d$, the smallest value a simplex vertex has for coordinate k is $-d^{-1}$, which strictly satisfies L_k.

$\{ t_k \mid 1 \le k < d \}$ Simplex vertex t_k violates constraint U_k. Every pyramid vertex and every other simplex vertex t_j has negative k-th coordinate, hence strictly satisfies U_k.

$\{ t_d, t_{d+1} \}$ The vertex t_d violates U_d. Every other vertex of S_d has non-positive d-th coordinate hence strictly satisfies U_d. The vertex t_{d+1} violates L_d. Every other vertex of S_d has non-negative d-th coordinate hence strictly satisfies L_d.

Observe that every constraint of C_d occurred in one of the cases above, hence every facet separates some vertex v of S_d from $\mathcal{V}(S_d) \setminus S_d$. ∎

Lemma 11 *For any facet F of C_d, $S_d \cap \mathrm{aff}(F) \subset \mathrm{relint}(F)$.*

For any facet F of C_d, we consider the $(d-1)$-polytope $F' \equiv \mathrm{aff}(F) \cap S_d$ and show that for any constraint H of F, there is a corresponding valid constraint H' for F' that strictly dominates H; that is any point that satisfies H' strictly satisfies H. We make use in following of the fact that that any positive linear combination of constraints for a convex polyhedron K is a valid constraint for K. Furthermore, since $F' \subset S_d$, any positive combination of constraints of S_d is a valid constraint for F'. Each facet of C_d is a scaled $(d-1)$-cube. Since the facets of C_d adjacent to a given facet are orthogonal to it, we may take the projected constraints of the facet to be precisely those constraints of C_d that are adjacent. The adjacent constraints of C_d to the facet induced by L_k and the facet induced by U_k are L_i, $i \neq k$ and U_i, $j \neq k$. There are four cases, one for each type of facet of C_d.

$(L_k,\ 1 \le k < d)$ We first consider a facet of C_d that separates a single pyramid vertex p_k from the other vertices of S_d (c.f. Lemma 10).

For $j \notin \{ k, d \}$, by projecting pyramid constraint H_{kj} into the hyperplane $x_k = -\frac{5}{4d}$, we obtain

$$x_j \ge -\frac{1}{2d} \tag{7}$$

which strictly dominates L_j for positive d. Summing $H_{k,d}$ and $H_{k,(d+1)}$ and projecting as above,

$$\sum_{i \notin \{ k,d \}} x_i \le \frac{1}{d}$$

Bounding x_i, $i \notin \{j, k, d\}$ by (7), we obtain

$$x_j \leq \frac{d-1}{2d},$$

Which strictly dominates the constraint U_j for positive d.

For $j = d$, combining (7), $H_{k,(d+1)}$, and $x_k = -5/4d$,

$$\begin{aligned} x_d - \frac{d-2}{2d} &\leq \frac{1}{d} \\ x_d &\leq \frac{1}{2}, \end{aligned}$$

which strictly dominates U_d. By a symmetric argument, $x_d \geq -1/2$, which strictly dominates L_d.

(U_k, $1 \leq k < d$) We next consider a facet of C_d that separates a single simplex vertex t_k from the other vertices of S_d.

We first show U_j is strictly dominated for $j \notin \{k, d\}$. Summing H_d and H_{d+1},

$$\sum_{i<d} x_i \leq \frac{1}{d}. \tag{8}$$

Any valid constraint for C_d is also a valid constraint for a facet of $S_d \cap C_d$. Summing constraint L_i over $i \notin \{j, k, d\}$,

$$\sum_{i \notin \{j,k,d\}} x_i \geq \frac{-5d+15}{4d}. \tag{9}$$

Substituting (9) into (8), along with the value $x_k = 1 - \frac{5}{4d}$, we obtain the following bound

$$x_j \leq \frac{d-6}{4d},$$

Which strictly dominates that of U_j for all positive d.

To obtain bounds on x_d, we substitute in for x_k to H_d and H_{d+1} (bounding x_j, $j \notin \{d, k\}$ as above).

$$\begin{aligned} x_d &\leq \frac{d-1}{4d} \\ x_d &\geq -\frac{d-1}{4d} \end{aligned}$$

These strictly dominate U_d and L_d for positive d.

It remains to show that any $L_j, j \notin \{k, d\}$ is dominated. Summing H_{ji} over $i \notin \{j, k, d\}$,

$$\sum_{i \notin j,k,d} x_i \geq (6-2d)x_j + \frac{9-3d}{d} \tag{10}$$

Taking the mean of H_{jd} and $H_{j,(d+1)}$ and substituting in (10) we obtain

$$\begin{aligned}\sum_{i \notin j,k,d} x_i - 4x_j + x_k &\leq \frac{6}{d} \\ (2-2d)x_j &\leq \frac{8d-7}{4d} \\ x_j &\geq -\frac{8d-7}{8d(d-1)}.\end{aligned}$$

This bound dominates L_j for $d \geq 2$.

(U_d, L_d) We next consider facets of C_d that cut off a single spine vertex t_d or t_{d+1}. By symmetry, we need only consider cutting off t_d. Summing the vertical pyramid constraints H_{kj} adjacent to t_d and p_k, we obtain

$$\sum_{i \notin \{ d,k \}} x_i \geq (-2\,d+4)\,x_k - 3 + \frac{6}{d}$$

Substituting in to $H_{k,(d+1)}$, we obtain

$$-2\,dx_k + x_d \leq 3$$

Recalling that in the facet created by cutting off t_d, $x_d = 1 - \frac{1}{4d}$,

$$\begin{aligned}-2\,dx_k &\leq 2 + \frac{1}{4d} \\ x_k &\geq -\frac{8d+1}{8d^2}\end{aligned}$$

This strictly dominates L_k for positive d. To obtain a corresponding upper bound on x_k, we substitute for x_d in $H_{k,(d+1)}$ to obtain

$$\sum_{i<d} x_i \leq \frac{5}{4d} - 1.$$

Bounding the value for x_i, $i \notin \{ d,k \}$ by L_i,

$$x_k \leq \frac{1}{4} - \frac{5}{4d},$$

which is tighter than U_k for all positive values of d. ■

There are several well known insertion orders for insertion algorithms.

minindex (maxindex) Insert the halfspaces in (the reverse of) the order given.

lexmin (lexmax) Insert the halfspaces in (reverse) lexicographic order of coefficient vector. There are several variations; one could reasonably reduce the constraint matrix to some canonical form, and there is the question of how to treat the right hand side vector. Here we assume that the input constraints are taken as is, and the right hand side entry is treated as the most significant digit of the coefficient vector (this is modelled on the program `cdd` [9]).

random Insert the halfspaces in random order.

maxcutoff (mincutoff) Insert the halfspace that causes the maximum (minimum) number of vertices and extreme rays to become infeasible.

Previous experiments (see [9]) have shown that there is no unique one of these orders that works well on all problems.

Theorem 5 *The polytope K_d, $d \geq 3$, has size $N = 4d^3 + 2d$ and any insertion algorithm to enumerate the vertices of K_d creates intermediate polytopes of size $\Omega(2^{\sqrt[3]{N}})$ in the worst case when the halfspaces are inserted in an order independent of the input, or in lexicographic order.*

Proof: In the case of an order such as minindex which ignores the input, it is trivial for an adversary to force an insertion algorithm to construct the entire d-cube. To fool lexicographic ordering, an adversary has merely to scale the constraints appropriately. ∎

Just as the Klee-Minty examples apply only to a particular pivoting order, this theorem applies only to particular insertion orders. Empirically, it would seem that there is indeed a polynomial insertion order for the truncated cubes (see Section 5.2). An interesting open question is whether every polytope has a polynomial insertion order.

5 Experimental Results

The following convex hull/vertex enumeration programs were used: `cdd`, `qhull` and `qrs`. `cdd` is version 0.52b of Fukuda's implementation of the double description method [9] with local modifications to add timing facilities and optimize set operations. `qhull` is Barber and Huhdanpaa's implementation of the beneath and beyond algorithm, version 2.01 [2]. `qrs` is Avis' implementation of reverse search [1] using Edmonds Q-pivoting, Version 2.4. We compiled `qhull` to use double precision. `cdd` uses double precision by default and `qrs` uses arbitrary precision rational arithmetic. The option `C-0` was used to force `qhull` to merge the generated simplicial facets. Times are measured in user CPU seconds.

5.1 Products of simplices

Table 1 shows the results for computing convex hull of $T_{d/2} \times T_{d/2}$, the products of simplices described in Section 3. The column "#simplices" contains the number of simplices necessary to triangulate all facets of the polytope. For `qhull` we had to use the exhaustive search for a starting simplex. For `cdd` we used the default insertion order, lexmin. Figure 2 shows the times for `cdd` and Figure 3 shows the times for `qrs` and `qhull`.

5.2 Truncated Cubes

As well as `qhull`, we tested the following insertion orders for `cdd`: minindex, maxindex, maxcutoff, and random. To simulate random insertion order, we permuted the input 200 times using the *Combined Random Number Generator* of L'Ecuyer [14, 15] and reported the average time. In our data files, the cube constraints come last, so the results of Section 4 tell us to expect exponential performance for maxindex. The *scaled* order scales the cube constraints to be lexicographically larger than the stacked polytope constraints, then uses lexmax. Table 2 gives a summary of results. Figure 4 shows the timings for minindex, which builds the stacked polytope first, up to 35 dimensions. Figure 5 shows timing results for other insertion orders for `cdd` and for `qhull`.

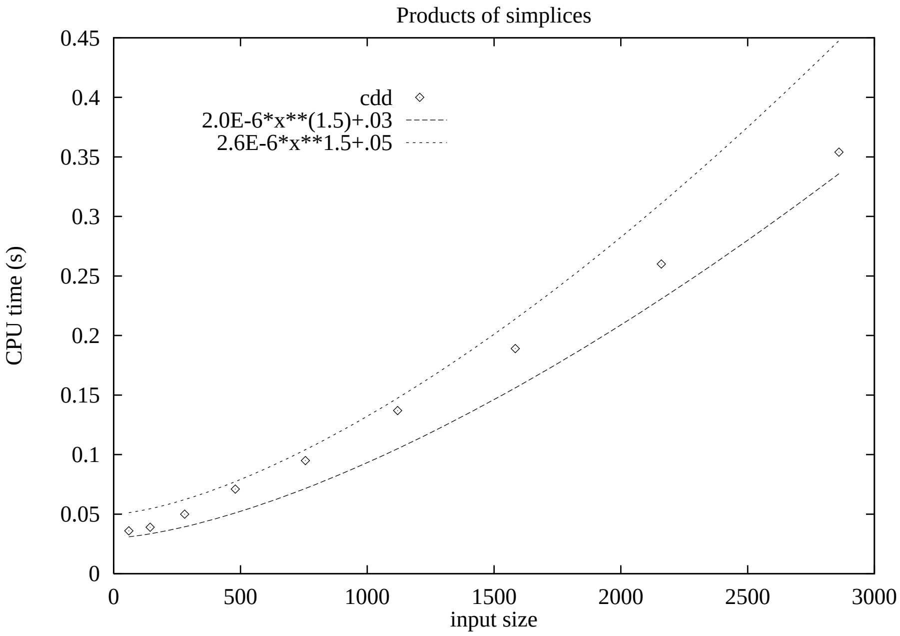

Figure 2: Products of simplices, `cdd`.

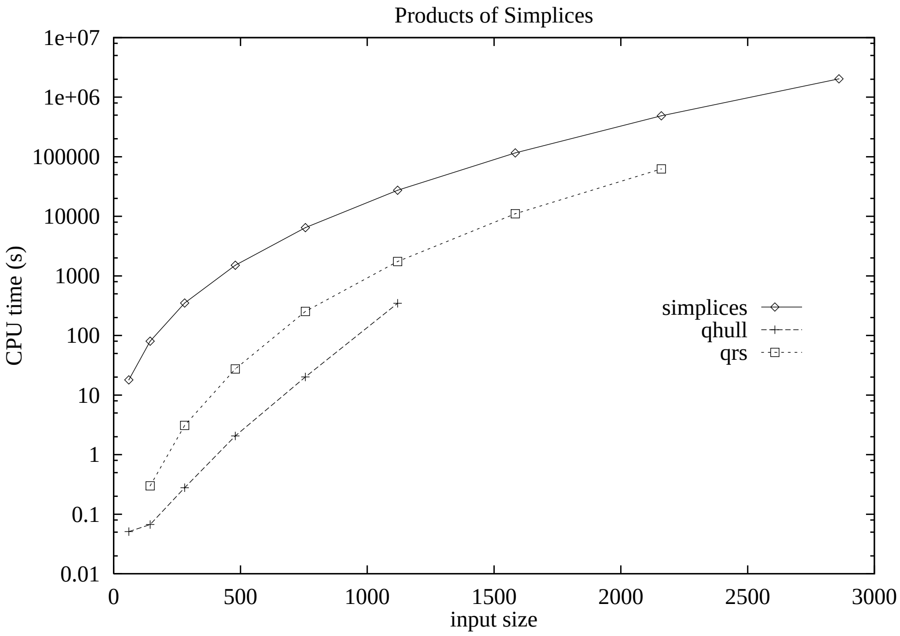

Figure 3: Products of simplices, `qrs` and `qhull`.

d	vertices n	facets m	*size* $(m+n)d$	#simplices	times(CPU seconds) cdd	qhull	qrs	perturbed bases (qrs)
4	9	6	60	18	0.036	0.051	0.0	18
6	16	8	144	80	0.039	0.067	0.3	80
8	25	10	280	350	0.050	0.278	3.1	410
10	36	12	480	1512	0.071	2.06	27.5	1825
12	49	14	756	6468	0.095	20.1	253.5	7024
14	64	16	1120	27456	0.137	346.839	1740	29389
16	81	18	1584	115830	0.189	(a)	11060	117838
18	100	20	2160	486200	0.260		62553	653581
20	121	22	2860	2032316	0.354		432040	2803146

(a) Virtual memory usage exceeded available 342 megabytes.

Table 1: Running times for convex hull on $T_{d/2} \times T_{d/2}$ (product of simplices in R^d). All timings on a DEC3000/500 Alpha with 96M of memory, running OSF/1 1.3.

Note that all insertion orders are monotonically increasing, except for maxcutoff, which is nonetheless bounded from below (in our experiments) by $3 \times 10^{-6} \times 2^{\sqrt[3]{size(P)}}$, as are all of the insertion orders tested except minindex.

In order to abstract away some of the dependence of the timing experiments on a particular machine, we considered two machine in dependant measures of performance. The first is the number of basic (high level) operations performed by `cdd`; the second is the size of the intermediate polytopes generated. `cdd` represents a face f of a polytope by the set of facets containing f. Checking vertices for adjacency involves several intersections of facet sets representing vertices. In Table 3 we report the number of set intersections performed as a measure of the work `cdd` performs in updating the edges. Another measure of the work done by the double description method is the maximum size of an intermediate polytope. Table 4 shows the maximum number of extreme rays in an intermediate polytope (`cdd` lifts each polytope to a homogeneous cone in one higher dimension). This measure is only an approximation, as `cdd` reports it only after every 100 rays generated.

Acknowledgements

The authors would like to thank the Minnesota Geometry Center for the software packages `qhull` and `Geomview`, and Komei Fukuda for `cdd`. We would also like to thank Günter Rote and Bernd Sturmfels for suggesting the examples in Section 3, Luc Devroye for pointers to random number generators, and Jesus deLoera for useful discussions.

		times (CPU seconds)					
d	size	maxindex	scaled	random	maxcutoff	minindex	qhull
2	32	0.006	0.006	0.006	0.006	0.006	0.010
3	114	0.013	0.013	0.013	0.013	0.012	0.018
4	264	0.033	0.035	0.030	0.030	0.024	0.056
5	510	0.100	0.119	0.072	0.056	0.057	0.208
6	876	0.241	0.480	0.186	0.132	0.094	0.861
7	1386	0.616	1.42	0.422	0.186	0.121	2.20
8	2064	1.69	4.66	1.16	0.882	0.178	8.44
9	2934	3.88	15.1	3.35	1.18	0.284	37.9
10	4020	9.58	46.5	8.90	0.950	0.395	139
11	5346	25.1	171	28.8	1.24	0.541	1101
12	6936	50.7	535	76.3	28.4	0.765	(a)
13	8814	113	1343	267	6.23	1.038	
14	11004	302	4486	1001	17.17	1.362	
15	13530	1031	15077	4274	73.4	1.819	
16	16416	3817	41635	(b)	4825	2.390	
17	19686	14614	123326		664	2.977	
18	23364	58142	(b)		1337	3.871	

(a) Virtual memory usage exceeded available 120 megabytes.

(b) Experiment in progress.

Table 2: Timing results for truncated cubes. All timings are on an SGI Indigo XS24 R4000 with 48M of memory and 72M swap, running IRIX 5.2.

d	maxindex	scaled	maxcutoff	minindex
2	42	40	84	38
3	441	346	545	218
4	2953	3321	1576	647
5	15204	25825	4429	1515
6	54532	155263	12347	3049
7	169188	566793	18537	5495
8	551773	1968927	128807	9148
9	1366733	6966913	165586	14358
10	3153737	19860935	102658	21518
11	8433970	67547177	139569	31063
12	17531284	199965060	6698317	43472
13	35612382	476434767	657547	59268
14	89069240	1509324332	3189971	79018
15	224617847	3731767397	20328100	103333
16	656832668	8497874780	903479053	132868
17	2267294806	23782814756	123175667	168322
18	8296480578	in progress	321679986	210438

Table 3: Set intersection operations performed by `cdd` for truncated cubes

d	maxindex	scaled	maxcutoff	minindex
4	40	52	(a)	(a)
5	82	152	63	70
6	148	401	92	59
7	214	653	109	137
8	392	1036	157	120
9	691	1433	180	194
10	1290	2063	240	266
11	2389	4090	272	335
12	4588	8189	720	402
13	8887	16288	468	467
14	17286	31487	831	528
15	33885	60186	2001	587
16	66984	113585	7776	740
17	132883	212384	2917	793
18	264282	(b)	3471	941

(a) No ray counts printed.

(b) Experiment in progress.

Table 4: Measured maximum intermediate sizes (number of extreme rays) for `cdd` on truncated cubes.

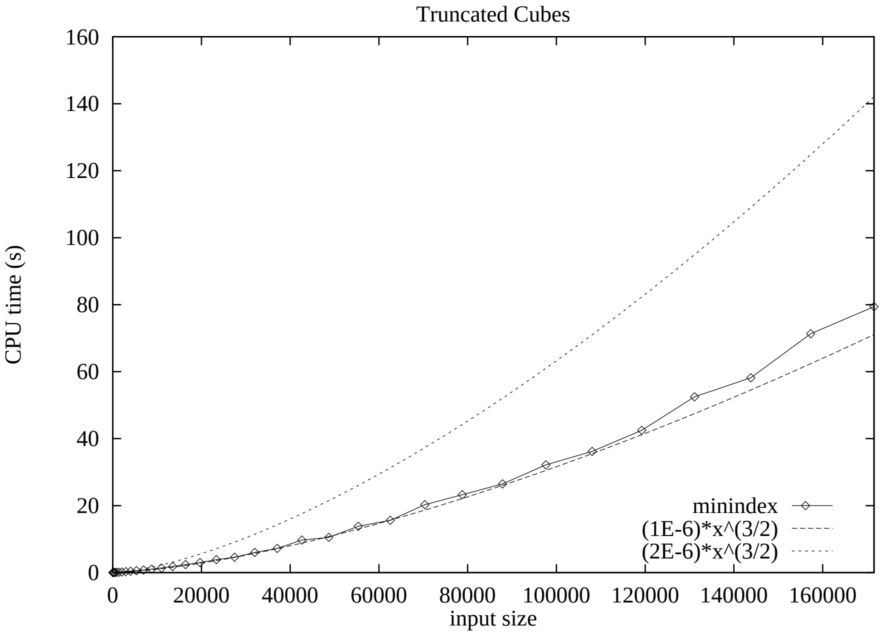

Figure 4: CPU time for truncated cubes, minindex insertion order.

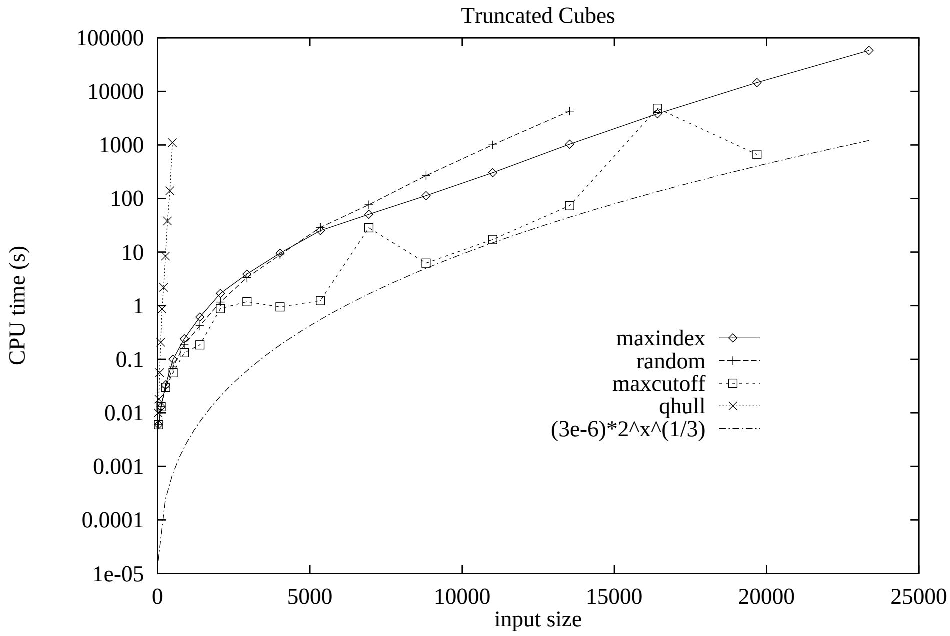

Figure 5: Truncated cubes, `qhull` and `cdd`

What	Host	Internet Address	Directory	Files
`cdd`	ftp.epfl.ch	128.178.139.3	incoming/dma	cdd-*.tar.gz
`qhull`	geom.umn.edu	128.101.25.35	pub/software	qhull.tar.Z
`qrs`	mutt.cs.mcgill.ca	132.206.3.13	pub/C	qrs.c
products of simplices	mutt.cs.mcgill.ca	132.206.3.13	pub/C/ext/prod	t2t2.ver, t4t4.ver ...
truncated cubes	mutt.cs.mcgill.ca	132.206.3.13	pub/C/ine/trunc	trunc*.ine

Table 5: Availability of Software and Data by anonymous ftp.

Availability of Software and Data

All of the software and data files described in this paper are available by anonymous ftp. See Table 5 for details.

References

[1] 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.

[2] B. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hull. Technical Report GCG53, The Geometry Center, University of Minnesota, July 1993.

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

[4] G. Ceder, G.D. 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.

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

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

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

[8] 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.

[9] K. Fukuda. cdd Reference manual, version 0.52b. EPFL, Lausanne, Switzerland.

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

[11] B. Grünbaum. *Convex Polytopes*. Wiley, New York, NY, 1967.

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

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

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

[15] L'Ecuyer and Tezuka. Structural properties for two classes of combined random number generators. *Mathematics of Computation*, 57, 1991.

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

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

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

[19] R. Seidel. Constructing higher-dimensional convex hulls at logarithmic cost per face. In *Proc. 18th Annu. ACM Sympos. Theory Comput.*, pages 404–413, 1986.

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