Computing All Faces of the Minkowski Sum of $\mathcal{V}$-Polytopes*

K. Fukuda Ch. Weibel

June 29, 2005

Abstract

We consider the problem of listing faces of the Minkowski sum of several $\mathcal{V}$-polytopes in $\mathbb{R}^d$. An algorithm for listing all faces of dimension up to j is presented, for any given $0 \leq j \leq d-1$. It runs in time polynomial in the sizes of input and output.

1 Introduction

Minkowski sums of $\mathcal{V}$-polytopes have applications in many domains ranging from mechanical engineering [7] to algebra. Gritzmann and Sturmfels [6] studied the problem and presented polynomial algorithms with fixed number of polytopes or dimension. Fukuda [5] developed polynomial algorithms for the general case computing the vertices of the sum. Here, we use an extended notion of polynomial algorithm: we call an algorithm *polynomial* if it runs in time polynomial in both the input and the output sizes. There is no known polynomial algorithm that computes the facets of a polytope from its vertices. The aim of this paper is to introduce a polynomial algorithm for computing all faces of the Minkowski sum of polytopes. This algorithm is an extension of Fukuda's algorithm [5]. The new algorithm lists higher dimensional faces by enumerating for each vertex v the faces whose sink is v in a prescribed LP orientation. The main advantage of this method is that computed faces need not be kept in memory during the course of the algorithm. Such an algorithm is called *compact*, i.e. the memory size is bounded by a polynomial in the size of the input only, unlike the time, which is necessarily at least linear in the output size. This allows the resolution of very large problems. This work is motivated by the strong demand in polyhedral computation and algebra, especially about mixed facets.

*Supported by the Swiss National Science Foundation Project 200021-105202, "Polytopes, Matroids and Polynomial Systems".

1.1 Introduction to polytopes

A *polytope* is the convex hull of a finite number of points in $\mathbb{R}^d$ ($\mathcal{V}$-representation):

$$P = \operatorname{conv}\{v_1, \dots, v_k\} = \{x \in \mathbb{R}^d : x = \sum_{i=1}^{k} \lambda_i v_i,\ \sum_{i=1}^{k} \lambda_i = 1,\ \lambda_i \geq 0\ \forall i\}.$$

Alternatively, they can be described as the bounded intersection of a finite number of closed half-spaces ($\mathcal{H}$-representation):

$$P = \{x \in \mathbb{R}^d : Ax \leq b\},\ P \text{ bounded.}$$

The Minkowski-Weyl theorem tells us those two definitions are equivalent. In fact these two representations are in many ways dual to each other, although known algorithms to switch between those two representations take an exponential time.

For any d-vector x and any scalar β, the linear equation $\langle c, x\rangle \leq \beta$ is called *valid* if $\langle c, x\rangle \leq \beta$ holds for all $x \in P$. A subset F of a polytope P is a *face* if there is a valid inequality $\langle c, x\rangle \leq \beta$ so that $F = P \cap \{x \in \mathbb{R}^d | \langle c, x\rangle = \beta\}$.

The set of all faces of P, denoted by $\mathcal{F}(P)$, ordered by set inclusion is called the *face lattice* of P. The smallest element in the face lattice is the empty set, and the largest is the polytope itself. These two faces are the *trivial* faces.

The *vertices* are the minimal nontrivial faces, and the *facets* are the maximal nontrivial faces..

If none of the points of a $\mathcal{V}$-representation is redundant, they form the vertices of the polytope, and if none of the half-spaces of an $\mathcal{H}$-representation is redundant and the polytope is full-dimensional, they determine its *facets*.

1.2 Minkowski sums

This section defines Minkowski sums of polytopes, and reviews some results that will be useful later in the article.

The Minkowski sum $P_1 + P_2$ of two polytopes P_1 and P_2 is defined as:

$$P_1 + P_2 = \{x \in \mathbb{R}^d \mid x = x_1 + x_2, x_1 \in P_1,\ x_2 \in P_2\}$$

This definition can be easily extended to sums of three or more polytopes. The difficulty of dealing with Minkowski sums is partly due to its exponential nature. For instance, the sum of k line segments orthogonally placed in $\mathbb{R}^d$ is the k-dimensional cube, which has 2^k vertices (and only $2k$ facets. To obtain an efficient algorithm to list (a certain class of) faces of the sum, we need to understand the interactions between faces of different dimensions.

For a polytope P in $\mathbb{R}^d$ and any vector $c \in \mathbb{R}^d$, we write

$$S(P;c) \;=\; \left\{ x \in P \mid \langle c,x\rangle = \max_{y\in P}\langle c,y\rangle \right\}$$

which means that $S(P;c)$ is the face of all maximizers of the linear function defined by c. Conversely, for any face F of P, we define $\mathcal{N}(F;P)$ as the set of vectors c such that $F = S(P;c)$, which is called the (*outer*) *normal cone* of P at F. Note that the normal cones are relatively open.

The collection of the closures of normal cones $\{\overline{\mathcal{N}(F;P)} | F \in \mathcal{F}(P)\}$ forms a polyhedral cell complex called the *normal fan* of P.

Faces of a polytope and their normal cones have dual properties. In particular, if F is a i-dimensional face of P, then the normal cone $\mathcal{N}(F;P)$ is a $(d-i)$-dimensional cone. The closure of the normal cones can be ordered by inclusion, and this order is the reverse of that of the corresponding faces:

Observation 1 *Let F and G be two faces of the polytope P. Then $F \subseteq G$ if and only if $\overline{\mathcal{N}(G;P)} \subseteq \overline{\mathcal{N}(F;P)}$.*

For instance, if v is a vertex of a polytope, then its normal cone $\mathcal{N}(\{v\};P)$ has the full dimension, and the faces of the polyhedral cone $\overline{\mathcal{N}(\{v\};P)}$ are the closures of the normal cones of the faces containg v. Let us recall some basic theorems on Minkowski sums of polytopes, from Gritzmann & Sturmfels [6] and Fukuda [5].

Theorem 1 *In a sum of polytopes, every face of the result can be decomposed in a sum of faces from the different summands. This decomposition is unique.*

Proof. *Outline: Any face of the sum polytope P can be written as the set of maximizers over P of the linear function $\langle c,x\rangle$ for some c, and this set is equal to the Minkowski sum of the maximizer faces of the summand polytopes:*

$$S(P_1+\cdots+P_k;c) = S(P_1;c)+\cdots+S(P_k;c).$$

One still needs to argue that there is no other way to represent the sum. □

As a result of this, we can characterize the outer normal cones of the sum polytope:

Corollary 2 *Outer normal cones of the sum polytope $P = P_1+\cdots+P_k$ are intersections of the outer normal cones of faces of the summands:*

$$F_i = S(P_i;c) \;\; \textit{for all } i, \textit{ and}$$

$$\mathcal{N}(F_1+\cdots+F_k;P) = \bigcap_{i=1}^{k} \mathcal{N}(F_i;P_i).$$

Corollary 3 *If v is a vertex of the Minkowski sum $P = P_1+\cdots+P_k$, then it is uniquely decomposed as $v = v_1+\cdots+v_k$, where v_i is a vertex of P_i for all i. Additionnally, there is a vector c so that $\{v\} = S(P;c)$ and $\{v_i\} = S(P_i;c)$ for all i.*

For the presentation of the algorithm, it is useful to note the corresponding result for edges:

Proposition 4 *If E is an edge of the Minkowski sum $P = P_1+\cdots+P_k$, then it is uniquely decomposed as $E = E_1+\cdots+E_k$, where E_i is either a vertex or an edge of P_i for all i, and all edges are parallel. There is also a vector c so that $E = S(P;c)$ and $E_i = S(P_i;c)$ for all i.*

This means that all edges of the sum polytope incident to a vertex are parallel to an edge incident to some vertex of the decomposition.

Proposition 5 *If u and v are adjacent vertices of the sum polytope $P = P_1+\cdots+P_k$, decomposed as $u = u_1+\cdots+u_k$ and $v = v_1+\cdots+v_k$, then u_i and v_i are either equal or adjacent vertices of P_i, and all adjacent pairs are linked by parallel edges.*

Proof. *The edge E for u to v is decomposed as a sum of faces $E = E_1+\cdots+E_k$, E_i are either vertices ($u_i = v_i$) or parallel edges (u_i and v_i are adjacent).* □

Therefore, if we have a vertex v of the sum polytope $P = P_1+\cdots+P_k$ decomposed in $v = v_1+\cdots+v_k$, the edges incident to the summands v_i give us possible candidates for edges incident to v in the sum polytope P. Fukuda's algorithm [5] to list all vertices of the sum exploits this property together with reverse search, a storage-free search technique. We shall review the algorithm in the next section.

2 Face Enumeration and Complexity

2.1 Vertices

The algorithm [5] lists all vertices of the polytope by moving from one to the other following the edges in the sum. To enumerate all vertices without duplicates, it employs the reverse search scheme [1, 2].

After choosing a generic linear function, every vertex is given a unique *parent*, which would be the next vertex visited in a simplex method. All vertices receive such a parent, except for the sink vertex. This effectively defines a rooted spanning tree of the graph of the sum polytope. The algorithm enumerates all vertices by starting from the root, and tracing the tree by depth-first search.

2.2 Larger faces

Unfortunately, computing larger face from the vertices of a polytope is a difficult problem, indeed there is no known algorithm to compute even facets from vertices in a polynomial time of the input and the output. To get better results, it is necessary to have supplementary information, e.g. the incidency between vertices and facets ([3]). It is therefore desirable to find an algorithm which finds larger faces during the computation of the vertices of the Minkowski sum.

The key idea for enumerating faces of higher dimensions is to extend Fukuda's algorithm with a face enumeration procedure. For every face F of the polytope, we define a unique parent vertex $t(F)$. Every time the algorithm visits a vertex, it lists all faces whose parent is the vertex (a partial face enumeration). Clearly, such an algorithm lists all faces without repetition. Furthermore, if we can restrict the partial enumeration with additional conditions, we have an algorithm that generates only such faces. We will see that a condition such as "the dimension is at most j" (for a fixed j) can be efficiently treated.

For the description of our algorithm, it is necessary to specify two things: a parent function t and a way to enumerate all faces whose parent is a given vertex v. The parent function should be chosen in a way that it is easy to evaluate and it is convenient for the face enumeration.

The vertex enumeration algorithm already uses an edge orientation of the graph, defined by augmentation of a chosen linear function. It turns out that this orientation defines a parent function t for all faces that satisfies the desired properties above. For any face F of the polytope, we define its parent $t(F)$ to be the vertex from which no edge leaves in F, i.e. $t(F)$ is the unique sink vertex of F. The edge orientation thus gives us a natural, and as we will see, convenient for the partial face enumeration.

To find the faces containing a vertex v, we will use the properties of the outer normal cone of the vertex $\mathcal{N}(\{v\};P)$. By the duality relation between the face lattice and the normal fan of the polytope, the facets of $\mathcal{N}(\{v\};P)$ are the normal cones of the edges incident to v. The set of faces of $\mathcal{N}(\{v\};P)$ are the normal cones of all faces containing v.

Therefore, the enumeration of faces containing v can be done by enumerating all faces of its normal cone. What we need to find is a way of enumerating only those faces whose parent is v.

The edges incident to v can be partitioned into two sets: those oriented towards v, and those oriented away from v. If a face F has v as parent, all its edges containing v are oriented towards it. This means that the normal cone of F is the intersection of normal cones of edges all oriented towards v. We now need to find a way of enumerating all faces having that property.

This can easily be done in time polynomial in the input and the output sizes, by using the backtrack method presented in [4]. This method enumerates all those faces of a $\mathcal{H}$-polyhedron that are not lying on any of a prescribed set of facets. Such a set S is specified as a set of input inequalities that should be "inactive". Furthermore, this algorithm can incorporate the additional condition such as the dimension is up to j, for any fixed j. This means we can easily eliminate faces which are not needed by adding appropriate inequalities to the set S, restricting the search to the edges which are oriented towards v. (Note taht when all edges incident to v are computed, the normal cone $\mathcal{N}(\{v\};P)$ is clearly an $\mathcal{H}$-polyhedron.)

Let δ_i be the maximum degree of the graph $G(P_i)$, and let δ be the sum $\delta_1 + \cdots + \delta_k$ of the maximum degrees. We denote by $l(d,m)$ time needed to solve any linear program in d variables and m inequalities.

Theorem 6 *There is a compact polynomial algorithm to list all faces of the Minkowski sum $P = P_1 + \cdots + P_k$ for given k $\mathcal{V}$-polytopes in $\mathbb{R}^d$ whose time complexity is $O(\delta\ l(d,m)\ f)$, where f is the number of all faces of P. Moreover, it can be extended to an algorithm to list only those faces of dimension up to j, for any fixed j, in time $O(\delta\ l(d,m)\ f_{\leq j})$, where $f_{\leq j}$ is the number of faces of dimension up to j.*

Proof. *First of all, listing of all vertices can be done by Fukuda's algorithm [5] in $O(\delta\ l(d,m)\ f_0)$ time, where f_0 denotes the number of vertices of P. The face enumeration can be done by the partial face enumeration at each vertex v. By the result [4, Theorem 3.1], the partial enumeration can be done in $O(\delta\ l(d,m)\ f_v)$ time, where f_v denotes the number of faces whose parent is v. By summing this over all vertices v and the vertex enumeration complexity, we obtain the claimed complexity $O(\delta\ l(d,m)\ f)$. By using a polynomial-time LP algorithm, the resulting algorithm will be a compact polynomial algorithm. Note that the time complexity depends also on the binary encoding length of the LP in the Turing model of computation.*

The claim for the extension is straightforward, as the backtracking algorithm [4] can be terminated at any specified dimension j. □

3 Algorithm

The proof of the main theorem, Theorem 6, contains the key components of the algorithm we propose. In this section, we give a structured description of the algorithm.

The description is based on a recursive function, called on the sink node of the polytope. The sum vertex is given by its vertex decomposition in the summands. $t(v) : V \to V$ is the parent function defined on the set of vertices V.

procedure exploreSubTree($v = v_1 + \cdots + v_k$);
  for all $i = 1, \ldots, k$ **do**
    for all neighbor u_i of v_i in P_i **do**
      if $u_i - v_i$ is a valid edge **do**
        Set $u :=$ vertex adjacent to v across $u_i - v_i$;
        if $t(u) = v$ **then**
          enumerateFaces(u);
          exploreSubTree(u);
        endif
      endif
    endfor
  endfor

A direction from a vertex can be determined as being a *valid* edge if it is possible to separate it from all other possible edges by an hyperplane. The procedure $enumerateFaces(u)$ enumerates all faces of dimension at least 1 which have u as sink node.

You'll notice that a vertex adjacent to v is explored only if v is its parent. This is the core of the reverse search scheme.

4 Output size

Since the time of the algorithm will be polynomial in the output size, it is natural to ask what will be the size of the output.

The number of faces in a Minkowski sum can be exponential. Gritzmann and Sturmfels have found a maximal bound for the number of i-dimensional faces in a Minkowski sum of k polytopes in terms of the number of nonparallel edges in the input.

Theorem 7 ([6]) *Let $P_1, \ldots, P_k$ be polytopes in $\mathbb{R}^d$, and m the number of nonparallel edges of $P_1, \ldots, P_k$. Then for $i = 0, \ldots, d-1$, we have:*

$$f_i(P_1 + \cdots + P_k) \leq 2 \binom{m}{i} \sum_{h=0}^{d-i-1} \binom{m-i-1}{h}$$

We should note that the number of faces need not be even close to this limit. Indeed, as shown in [5], there is an infinite family of Minkowski sum problems for which the number of vertices (and faces) in the sum polytope is bounded by those in the summands.

This explains why it is important to design algorithms that are sensitive to the output size.

5 Conclusion

We have presented a compact polynomial algorithm for enumerating all faces of a Minkowski addition of polytopes. This is a common extension of algorithms proposed in [4, 5].

This algorithm has applications ranging from algebra ([6],[8]) to automotive construction ([7]). Also this tool allows us to significantly improve our knowledge of Minkowski additions of polytopes.

References

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

[2] D. Avis and K. Fukuda. Reverse search for enumeration. In *Discrete Applied Mathematics*, 65:21-46, 2004.

[3] K. Fukuda and V. Rosta. Combinatorial face enumeration in convex polytopes In *Combinatorial Geometry*, 4:191-198, Elsevier, 1994.

[4] K. Fukuda, Th. Liebling and F. Margot Analysis of backtrack algorithms for linsting all vertices and all faces of a convex polyhedron In *Computational Geometry*, 8:1-12, Elsevier, 1997.

[5] K. Fukuda. From the zonotope construction to the Minkowski addition of convex polytopes In *Journal of Symbolic Computation*, 38(4):1261-1272, 2004.

[6] P. Gritzmann and B. Sturmfels. Minkowski addition of polytopes: computational complexity and applications to Gröbner Bases In *SIAM J. Disc. Math.*, 6:246-269, 1993.

[7] J.-P. Petit. Spécification géometrique des produits : Methode de détérmination des tolérances. Application en conception assistée par ordinateur. *Ph.D Thesis*, Univ. de Savoie, 2004.

[8] B. Sturmfels. Gröbner bases and convex polytopes *University Lectures Series*, 8, Amer. Math. Soc., 1996.