From the Zonotope Contstruction to the Minkowski Addition of Convex Polytopes

Komei Fukuda*
School of Computer Science, McGill University, Montreal, Canada
fukuda@cs.mcgill.ca

January 11, 2003

Abstract

A zonotope is the Minkowski addition of line segments in R^d. The zonotope construction problem is to list all extreme points of a zonotope given by its line segments. By duality, it is equivalent to the arrangement construction problem that is to generate all regions of an arrangement of hyperplanes.

By replacing line segments with convex V-polytopes, we obtain a natural generalization of the zonotope construction problem: the construction of the Minkowski addition of k polytopes. Gritzmann and Sturmfels studied this general problem in various aspects and presented polynomial algorithms for the problem when one of the parameters k or d is fixed. The main objective of the present work is to introduce an efficient algorithm for variable d and k. Here we call an algorithm *efficient* or *polynomial* if it runs in time bounded by a polynomial function of both the input size and the output size. The algorithm is a natural extension of a known algorithm for the zonotope construction, based on linear programming and reverse search. It is compact, highly parallelizable and very easy to implement.

This work has been motivated by the use of polyhedral computation for optimal tolerance determination in mechanical engineering.

1 Introduction

Geometric computation in general dimension requires a special attention due to socalled "curse of dimension." For example, the number of facets of the convex hull of a set of k points in Euclidean d-space can be of order $k^{\lfloor d/2 \rfloor}$ even when d is considered fixed. Therefore merely outputting the facet representation of the convex hull might take considerable time for dimension as small as six and is unlikely to be practically done for dimension higher than, say ten, for the class of input attaining the worst output order. The situation is even worse for the arrangement construction problem of outputting all regions (cells) of a central arrangement of k hyerplanes in $\mathbb{R}^d$, since the largest outsize is of order $O(k^{\lfloor d-1 \rfloor})$. Fortunately, many real-world problems are often very different from the worst output cases and one can solve many practical instances of the convex hull problem in high dimensions well above ten and sometimes in dimension over sixty. Looking

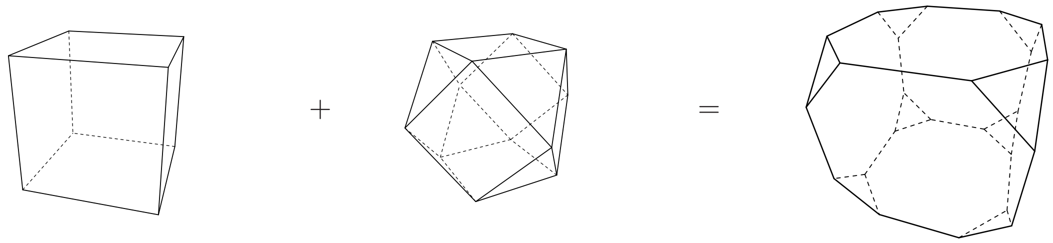
*The paper was written while the author was visiting Institute for Operations Research, ETHZ, CH-8092 Zurich, Switzerland. The research is partially supported by an NSERC grant, Canada.

into various instances arising in practice, one can observe that most of real-world problems have special structures that make the problems very different from the worst-case-output instances.

One way to go around the curse of dimension is to look for output sensitive algorithms. More formally we call an algorithm *polynomial* or *efficient* if it runs in time polynomial in both the input size and the output size. A polynomial algorithm might be more practical than an worst-case optimal algorithm, i.e. an algorithm that runs in time linear in the size of the worst case output for any given input instance size. The benefit of designing an polynomial algorithm is well justified by fhe fact that the range of output sizes often varies from a constant to an exponential function in the input size. While an worst-case optimal algorithm can be a polynomial algorithm at the same time, such an algorithm seems to be extremely rare. This is not surprising since a worst-case optimal algorithm aims at being optimal for the worst-case instances, a polynomial algorithm must deal with all possible instances efficiently including those with small output sizes.

The main objective of the present paper is to introduce an polynomial algorithm for the Minkowski addition of k convex polytopes in $\mathbb{R}^d$. Our algorithm is also a *compact* algorithm, i.e. algorithm whose space complexity is polynomially bounded by the input size only. The compactness is obviously important because the size of output is usually enormously large. Gritzmann and Sturmfels [6] studied this problem to design worst-case optimal algorithms with obtained many fundamental results with interesting applications to the Gröbner basis computation. As we explained, our goal of finding a polynomial algorithm for variable dimension and variable k is considerably different. In fact, our new algorithm is totally different from any of the algorithms proposed in [6], although all algorithms rely on linear programming (LP).

Let us define the problem formally. The *Minkowski addition* or *sum*, denoted by $P + Q$, of subsets P, Q of $\mathbb{R}^d$ is defined as the set $\{x + y \mid x \in P \text{ and } y \in Q\}$. P and Q are called *Minkowski summands* of $P + Q$. Let P_1, P_2, ..., P_k be (convex) $\mathcal{V}$*-polytopes* i.e. polytopes given by their sets of extreme points V_1, V_2, ..., V_k. Thus P_i is the convex hull $conv(V_i)$ of V_i for each i. The (polyhedral) *Minkowski addition problem* is to compute the set V of all extreme points of the Minkowski addition $P_1 + P_2 + \cdots + P_k$, when V_1, V_2, ..., V_k are given. The following is an example of the Minkowski addition of 2 polytopes in $\mathbb{R}^3$.



It is a natural generalization of the zonotope construction problem of generating all extreme points of the Minkowski sum of k line segments in $\mathbb{R}^d$. By polarity, the zonotope construction problem is equivalent to the arrangement construction problem mentioned above with the same parameters. As shown in [6], the worst case output has size $O(k^{d-1}n^{2(d-1)})$ where n is the largest number of extreme points in P_i's. On the other hand, there is a class of nontrivial problems whose output size is $O(k\,n)$, as shown in Proposition 2.6. These already provide us with a good reason to look for a polynomial algorithm rather than a worst-case optimal algorithm.

We will show that the reverse search algorithms presented in [2, 3] can be extended naturally to the Minkowski addition problem, once the original algorithm is dualized for the zonotope construction problem and the basic combinatorial properties of the Minkowski addition are observed. The resulting algorithm is a compact polynomial algorithm which can be highly parallelizable. In addition, the algorithm is fairly easy to implement with an external linear programming solver. We should also note that the algorithm can be further extended to unbounded cases where each P_i is given by its vertices and extreme rays.

Gritzmann and Sturmfels [6] presented applications of the Minkowski addition problem to the Gröbner basis computation. In particular, it was shown that the Minkowski addition computation can be used to determine whether a given set of polynomials is a Gröbner basis with respect to some monomial order, and if the answer is not, it can be used to find a monomial order which is optimal in some natural measure associated with the Hilbert function of an polynomial ideal. Recently we have encountered a new application in tolerance analysis and synthesis in mechanical engineering [5]. They use six dimensional mathematical models representing mechanical parts, joints and their displacements in order to analyze and design optimal tolerances. The dimension six comes from three plus three, each representing the 3D location and 3D rotation. One tolerance analysis requires to check whether $A \subseteq B$ where both A and B are six dimensional polytopes, each given as a Minkowski addition of polytopes.

We shall not discuss a variant of the Minkowski addition problem where input and output polytopes are $\mathcal{H}$-polytopes, i.e. those given by systems of inequalities. In fact, the existence of a polynomial algorithm (for variable d and k) for the Minkowski $\mathcal{H}$-addition problem is open. For the special case of the zonotope $\mathcal{H}$-construction, Seymour's polynomial (and non-compact) algorithm [7] exists but it is not clear the algorithm can be extended to the general case.

2 Combinatorial Properties of the Minkowski Addition

In this section, we review some of the basic properties of the Minkowski sum of k polytopes given by Gritzmann and Sturmfels [6]. We shall follow thier notations whenever possible. Also, we present some new results on the Minkowski sum that are useful to see the importance of polynomial algorithms to compute the Minkowski sum.

Faces, Minkowski Decomposition and Adjacency

A *convex polytope* or simply *polytope* is the convex hull of a finite set of points in $\mathbb{R}^d$. For a polytope P and for any vector $c \in \mathbb{R}^d$, the set of maximizers x of the inner product $c^T x$ over P is denoted by $S(P; c)$. Thus each nonempty face of P is $S(P; c)$ for some c. We denote by $\mathcal{F}(P)$ the set of faces of P, by $\mathcal{F}_i(P)$ the set of i-dimensional faces, and by $f_i(P)$ the number of i-dimensional faces, for $i = -1, 0, \dots, d$, For each nonempty face F, the relatively open polyhedral cone of outer normals of P at F is denoted by $\mathcal{N}(F; P)$. Thus, $c \in \mathcal{N}(F; P)$ if and only if $F = S(P; c)$. The *normal fan* $\mathcal{N}(P)$ of P is the cell complex $\{\mathcal{N}(F; P) | F \in \mathcal{F}(P)\}$ whose body is $\mathbb{R}^d$. If F is i-dimensional ($i = 0, 1, \dots, d$), the normal cone $\mathcal{N}(F; P)$ is $(d-i)$-dimensional. Thus the extreme points of P are in one-to-one correspondence with the full dimensional faces (which we call the *regions* or *cells*) of the complex.

Proposition 2.1 *Let P_1, P_2, ..., P_k be polytopes in $\mathbb{R}^d$ and let $P = P_1 + P_2 + \cdots + P_k$. Then a nonempty subset F of P is a face of P if and only if $F = F_1 + F_2 + \cdots + F_k$ for some face F_i of*

P_i *such that there exists* $c \in \mathbb{R}^d$ *(not depending on* i*) with* $F_i = S(P_i; c)$ *for all* i*. Furthermore, the decomposition* $F = F_1 + F_2 + \cdots + F_k$ *of any nonempty face* F *is unique.*

Proof. The equivalence follows directly from the obvious relation [6, Lemma 2.1.4]

$$S(P_1 + P_2 + \cdots + P_k; c) = S(P_1; c) + S(P_2; c) + \cdots + S(P_k; c) \text{ for any } c \in \mathbb{R}^d.$$

For the uniqueness, let F be a nonempty face with $F = S(P; c)$ for some c and let $F = F_1 + F_2 + \cdots + F_k$ be any decomposition. First, note that $F_i \subseteq S(P_i c)$ for all i, because the value $c^T x$ for any $x \in F$ is the sum of the maximum values $c^T x_i$ subject to $x_i \in P_i$ for $i = 1, \dots, k$, and thus if $x \in F$ and $x = x_1 + x_2 + \cdots + x_k$ for $x_i \in F_i$, then $x_i \in S(P_i, c)$. Now suppose there exists F_i properly contained in $S(P_i; c)$. Let v be an extreme point of $S(P_i; c)$ not in F_i. Then there is a linear function $w^T x$ such that $w^T v$ is strictly greater than any value attained by $x \in F_i$. Now let x^* be any point attaining the maximum of $w^T x$ over the polytope $F_1 + F_2 + \cdots F_{i-1} + F_{i+1} + \cdots + F_k$. Clearly $x^* + v \in F$ but this point cannot be in $F_1 + F_2 + \cdots + F_k$, a contradiction. This proves the uniqueness. ∎

We refer the unique decomposition $F = F_1 + F_2 + \cdots + F_k$ of a nonempty face F as the *Minkowski decomposition*. Here, the dimension of F is at least as large as the dimension of each F_i. Thus we have the following.

Corollary 2.2 *Let* P_1*,* P_2*, ...,* P_k *be polytopes in* $\mathbb{R}^d$ *and let* $P = P_1 + P_2 + \cdots + P_k$*. A vector* $v \in P$ *is an extreme point of* P *if and only if* $v = v_1 + v_2 + \cdots + v_k$ *for some extreme point* v_i *of* P_i *and there exists* $c \in \mathbb{R}^d$ *with* $\{v_i\} = S(P_i; c)$ *for all* i*.*

For our algorithm to be presented in the next section, it is important to characterize the adjacency of extreme points in P.

Corollary 2.3 *Let* P_1*,* P_2*, ...,* P_k *be polytopes in* $\mathbb{R}^d$ *and let* $P = P_1 + P_2 + \cdots + P_k$*. A subset* E *of* P *is an edge of* P *if and only if* $E = E_1 + E_2 + \cdots + E_k$ *for some face* E_i *of* P_i *such that* $dim(E_i) = 0$ *or* 1 *for each* i *and all faces* E_i *of dimension 1 are parallel, and there exists* $c \in \mathbb{R}^d$ *with* $E_i = S(P_i; c)$ *for all* i*.*

The following variation of the above is useful for the algorithm to be presented. The essential meaning is that the adjacency of extreme points is inherited from those of Minkowski summands.

Proposition 2.4 *Let* P_1*,* P_2*, ...,* P_k *be polytopes in* $\mathbb{R}^d$ *and let* $P = P_1 + P_2 + \cdots + P_k$*. Let* u *and* v *be adjacent extreme points of* P *with the Minkowski decompositions:* $u = u_1 + u_2 + \cdots + u_k$ *and* $v = v_1 + v_2 + \cdots + v_k$*. Then* u_i *and* v_i *are either equal or adjacent in* P_i *for each* i*.*

Proof. Let u and v be adjacent extreme points. It is sufficient to show that $[u, v] = [u_1, v_1] + [u_2, v_2] + \cdots + [u_k, v_k]$ and each $[u_i, v_i]$ is a face of P_i. Let $c \in \mathbb{R}^d$ be such that $[u, v] = S(P; c)$. Because $[u, v] = S(P_1; c) + S(P_2; c) + \cdots + S(P_k; c)$ and by the uniqueness of decomposition of u and v, both u_j and v_j are in $S(P_j, c)$, for all j. This implies that $[u_j, v_j] \subseteq S(P_j, c)$, for all j. On the other hand, one can easily see that in general $[u, v] \subseteq [u_1, v_1] + [u_2, v_2] + \cdots + [u_k, v_k]$. The last two relations give $[u_j, v_j] = S(P_j, c)$ for all j. ∎

This proposition immediately provides a polynomial algorithm for listing all neighbors of a given extreme point using linear programming.

Face Complexity

One of the basic questions on the Minkowski addition of polytopes is its complexity in terms of the number of faces, in particular the number of extreme points in the present paper.

Gritzmann and Sturmfels obtained the largest number of i-faces in the Miknowski addition of $k$polytopes in terms of the number of non-parallel edges in the input polytopes.

Theorem 2.5 ([6]) *Let P_1, P_2, ..., P_k be polytopes in $\mathbb{R}^d$ and let m be the number of non-parallel edges of P_1, P_2, ..., P_k. Then $f_i(P_1 + P_2 + \cdots + P_k)$ attains its maximum when each P_j is a generic zonotope, and thus we have:*

$$f_i(P_1 + P_2 + \cdots + P_k) \leq 2\binom{m}{i}\sum_{h=0}^{d-i-1}\binom{m-i-1}{h}, \text{ for } i = 0, 1, \ldots, d-1.$$

In the theorem above, the number f_0 of 0-faces (extreme points) is the most important for our purpose:

$$f_0(P_1 + P_2 + \cdots + P_k) \leq 2\sum_{h=0}^{d-1}\binom{m-1}{h}. \tag{2.1}$$

Thus, for fixed d, the bound is in $O(m^{d-1})$. This bound itself does not immediately show the fact that the output size of the Minkowski addition problem may not be polynomially bounded by the input size, because the input size can be exponentially large in m, e.g. when P_j's are possibly rotated d-hypercubes. The simplest way to see it is by setting P_j's to be line segments in general position (i.e. $m = k$). Then this upper bound is exactly the number of extreme points of a generic zonotope generated by k intervals (i.e. k pairs of points) in $\mathbb{R}^d$, and for fixed d it is in $\Theta(k^{d-1})$. This indicates that the output size cannot be polynomially bounded by the input size. In contrast, there is an infinite family of the Minkowski addition problems whose output size is small and linearly bounded by the size of input.

Proposition 2.6 *For each $k \geq 2$ and $d \geq 2$, there is an infinite family of the Minkowski addtion problems for which $f_0(P_1 + P_2 + \cdots + P_k) \leq f_0(P_1) + f_0(P_2) + \cdots + f_0(P_k)$.*

Proof. Suppose $k \geq 2$ and $d \geq 2$. First pick up any d-polytope, say Q, with at least k extreme points, and select k extreme points. For each jth selected extreme point v^j, make a new polytope P_j from Q by truncating only v^j with one or more hyperplanes. Now we claim that the number $f_0(P_1 + P_2 + \cdots + P_k) \leq f_0(P_1) + f_0(P_2) + \cdots + f_0(P_k)$. See Figure 1 for an example for $k = 2$, $d = 3$ and Q is a 3-cube. To see this, let v be an extreme point of P_j for some fixed j. There are three cases. The first case is when v is an unselected one, i.e. an extreme point of Q not selected. In this case, it can be an Minkowski summand of an extreme point of P in a unique way, since any linear function maximized exactly at v over P_j is maximized exactly at v over other P_i's. The second case is when v is a newly created vertex by the truncation of v^j. Since it is obtained by the trunction of v^j, any linear function maximized exactly at v over P_j is maximized exactly at v^j over other other P_i's. The last case is when $v = v^i$ for some $i \neq j$. This case is essentially the same as the second case where v contributes uniquely to a new extreme point with each truncation vertex of P_i. By Corollary 2.2, every extreme point of P_j contributes at most once to $f_0(P_1 + P_2 + \cdots + P_k)$. This complete the proof. ∎

Proposition 2.6 shows that a polynomial algorithm can be much more efficient than any worst-case optimal algorithm that runs in $\Theta(m^{d-1})$ for fixed d, such as one given in [6]. Note that $m \geq d/2(f_0(P_1) + f_0(P_2) + \cdots + f_0(P_k))$ when there are no parallel edges in the input.

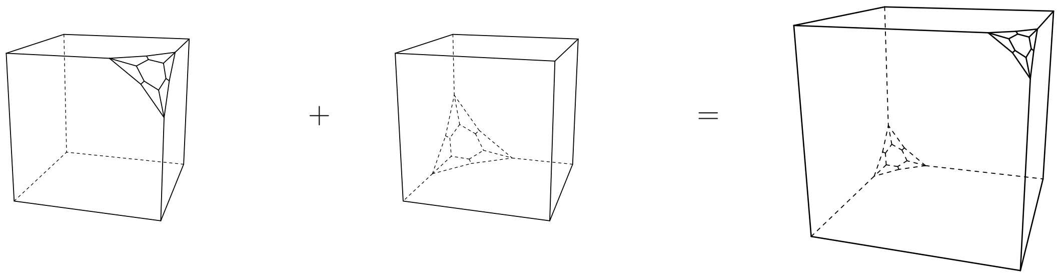


Figure 1: A construction of "easy" Minkowski addition problems

3 Extension of a Zonotope Construction Algorithm

We assume in this section that P_1, P_2, ..., P_k are polytopes in $\mathbb{R}^d$ given by the sets V_1, V_2, ..., V_k of extreme points. We also assume that the graph $G(P_j)$ of P_j is also given by the adjacency list $(a_j(v,i) : i = 1, \dots, \delta_j)$ of vertices adjacent to vertex $v \in V_j$ in graph $G(P_j)$, where δ_j is the maximum degree of $G(P_j)$ for each $j = 1, \dots, k$. If the degree $deg_j(v)$ of v is less than δ_j in $G(P_j)$, we assume that $a_j(v,i) = null$ for all $i > deg_j(v)$. Finally we define $\delta = \delta_1 + \delta_2 + \cdots + \delta_k$, an upper bound of the maximum degree of $G(P)$, due to Proposition 2.4. Note that for a given set V_j, one can compute the adjacency list in polynomial time using linear programming.

Recall that the Minkowski addition problem is to compute the set V of extreme points of $P = P_1 + P_2 + \cdots + P_k$. We shall present a compact polynomial algorithm for the Minkowski addition problem.

The key idea in our algorithm design

The main algorithmic idea is quite simple. Just like for the vertex enumeration for convex polyhedra using reverse search [1], it traces a directed spanning tree T of the graph $G(P)$ of P rooted at an initial extreme point v^*. The diffence from the vertex enumeration algorithm is that the polytope P is not given by a system of inequalities (i.e. not an H-polytope) in the present setting but as a Minkowski-addition of V-polytopes. Thus we need to introduce a new way of defining a directed spanning tree that is easy to trace. We shall use the following simple geometric property of normal fans.

Proposition 3.1 *Let v and v' be two distinct extreme points of P, and let $c \in \mathcal{N}(v;P)$ and $c' \in \mathcal{N}(v';P)$. Then there exists an extreme point v'' adjacent to v such that $\mathcal{N}(v'';P)$ contains a point of form $(1-\theta)c + \theta c'$ for some $0 \le \theta \le 1$.*

Proof. Since $v \neq v'$, their outer normal cones are two distinct full dimensional cones in the normal fan $\mathcal{N}(P)$. This means that the parameterized point $t(\theta) := c + \theta(c' - c)$ $(0 \le \theta \le 1)$ in the line segment $[c, c']$ must leave at least one of the bounding halfspaces of the first cone $\mathcal{N}(v;P)$ as θ increases from 0 to 1. Since the bounding halfspaces of $\mathcal{N}(v;P)$ are in one-to-one correspondence with the edges of G incident to v, any one of the halfspaces violated first corresponds to a vertex v'' adjacent to v claimed by the proposition. ∎

Let us fix v^* as an initial extreme point of P. Finding one extreme point of P is easy. Just select any generic $c \in \mathbb{R}^d$, and find the unique maximizer extreme point v^i of $c^T x$ over P_i, for each i. The point $v = v^1 + v^2 + \cdots + v^k$ is an extreme point of P.

Now we construct a directed spanning tree of $G(P)$ rooted at v^* as follows. Let $v \in V$ be any vertex different from v^*. We assume for the moment that there is some canonical way to select an interior point of the normal cone of P at any given vertex, as we shall give one method to determine such a point later. Let c and c^* be the canonical vector of $\mathcal{N}(v; P)$ and $\mathcal{N}(v^*; P)$, respectively. By Proposition 3.1, by setting $v' = v^*$, we know that there is a vertex v'' adjacent to v such that $\mathcal{N}(v''; P)$ meets the segment $[c, c^*]$. In general there might be several such vetices v'' (degeneracy). We break ties by the standard symbolic perturbation of c as $c+(\epsilon^1, \epsilon^2, \ldots, \epsilon^d)^T$ for sufficiently small $\epsilon > 0$. Define the mapping $f : V \setminus \{v^*\} \to V$ as $f(v) = v''$. This mapping, called a *local search function* in reverse search, determines the directed spanning tree $T(f) = (V, E(f)$ rooted at v^*, where $E(f)$ is the set of directed edges $\{(v, f(v)) | v \in V \setminus \{v^*\}\}$.

Proposition 3.2 *The digraph $T(f)$ is a spanning tree of $G(P)$ (as undirected graph) and v^* is a unique sink node of $T(f)$.*

Proof. By the construction, v^* is a unique sink node of $T(f)$. It is sufficient to show that $T(f)$ has no directed cycle. For this, take any edge $(v, v'' = f(v)) \in E(f)$. Let c, c^* be the canonical vector for v, v^*, respectively. Without loss of generality, we assume nondegeneracy, since one can replace c with the perturbed vector $c + (\epsilon^1, \epsilon^2, \ldots, \epsilon^d)^T$. Since c is an interior point of $\mathcal{N}(v; P)$,

$$c^T(v - v'') > 0. \tag{3.2}$$

Again, by the construction and because the canonical points are selected as interior points of the associated normal cones, there exists $0 < \theta < 1$ such that $\hat{c} := (1 - \theta)c + \theta c^* \in \mathcal{N}(v''; P)$. This implies $\hat{c}^T(v'' - v) > 0$, that is,

$$\begin{aligned} 0 &< ((1-\theta)c + \theta c^*)^T(v'' - v) \\ &= (1-\theta)c^T(v'' - v) + \theta(c^*)^T(v'' - v) \\ &< \theta(c^*)^T(v'' - v) \qquad (\text{ by } 3.2) . \end{aligned}$$

This implies that the vertex v'' attains a strictly higher inner product with c^* than v. Therefore, there is no directed cycle in $T(f)$. ■

The critical computation in our algorithm is solving the linear programming. We denote by $LP(d, m)$ the time necessary to solve a linear programming in d variables and m inequalities. Here we ignore the dependency on the binary size of input for simplicity, and we assume that the time necessary to solve an LP depends only on the two critical parameters. We assume that the constant magnification of the problem does not change the complexity: $O(LP(d, m)) = O(LP(cd, m)) = O(LP(d, cm))$ for any constant c, which is true with any polynomial algorithms, and is practically true with the simplex method. One can easily replace $LP(d, m)$ by $LP(d, m, L)$ where L is the binary size of an input LP for a more precise analysis. Now we can state the complexity of our algorithm.

Theorem 3.3 *There is a compact polynomial algorithm for the Minkowski addition of k polytopes that runs in time $O(\delta\, LP(d, \delta) f_0(P))$ and space linear in the input size.*

The algorithm

The sequel of the section is devoted to present the technical details of a reverse search algorithm that traces $T(f)$ starting from its root vertex v^* against the orientation of edges. We shall prove Theorem 3.3 at the end.

As usual, our reverse search algorithm requires, in addition to the local search function f, an adjacency oracle function that implicitly determines the graph $G(P)$.

Let v be any vertex of P with the Minkowski decomposition $v = v_1 + v_2 + \cdots + v_k$ (see, Corollary 2.2). Let

$$\Delta = \{(j,i) : j = 1,\dots,k \text{ and } i = 1,\dots,\delta_j\}. \tag{3.3}$$

Recall that for any $(j,i) \in \Delta$, $a_j(v_j,i)$ is the ith vertex adjacent to v_j whenever it is not *null*. We shall call a pair (j,i) *valid* for v if $a_j(v_j,i) \neq \textit{null}$, and *invalid* otherwise. Let us define the associated edge vectors $e_j(v_j,i)$ by

$$e_j(v_j,i) = \begin{cases} a_j(v_j,i) - v_j & (j,i) \text{ is valid for } v \\ \textit{null} & \text{otherwise.} \end{cases} \tag{3.4}$$

Proposition 2.4 shows that all edges of P incident to v are coming from the edges incident to v_j's, or more precisely, each edge of P incident to v is parallel to some $e_j(v_j,i)$. This immediately implies that δ is an obvious upper bound of the degree of v. For each $(s,r) \in \Delta$, let us group the same (parallel) directions together as

$$\Delta(v,s,r) = \{(j,i) \in \Delta : e_j(v_j,i) \parallel e_s(v_s,r)\}. \tag{3.5}$$

Consider it as the empty set if (s,r) is invalid. Now, for any given pair $(s,r) \in \Delta$, checking whether $e_s(v_s,r)$ determines an edge direction of P is easily reducible to an LP (or more precisely, a linear feasibility problem):

$$\begin{aligned} e_s(v_s,r)^T\lambda &< 0, \\ e_j(v_j,i)^T\lambda &\geq 0 \quad \text{for all valid } (j,i) \notin \Delta(v,s,r). \end{aligned} \tag{3.6}$$

More precisely, the system (3.6) has a solution λ if and only if the direction $e_s(v_s,r)$ determines an edge of P incident to v. If it has a feasible solution, then by Proposition 2.4, the vertex $\hat{v}$ adjacent to v along this direction is given by

$$\hat{v} = \hat{v}_1 + \hat{v}_2 + \cdots + \hat{v}_k$$
$$\hat{v}_j = \begin{cases} a_j(v_j,i) & \text{if there exists } i \text{ such that } (j,i) \in \Delta(v,s,r) \\ v_j & \text{otherwise.} \end{cases}$$

Let us denote by $\Delta(v)$ as the set of all pairs $(s,r) \in \Delta$ such that $e_s(v_s,r)$ determines an edge of P and (s,r) is a member of $\Delta(v,s,r)$ with the smallest first index. This set represents a duplication-free index set of all edge directions at v.

Now we are ready to define our adjacency oracle as a function $Adj : V \times \Delta \to V \cup \{\textit{null}\}$ such that

$$Adj(v,(s,r)) = \begin{cases} \hat{v} & \text{if } (s,r) \in \Delta(v) \\ \textit{null} & \text{otherwise.} \end{cases} \tag{3.7}$$

Lemma 3.4 *One can evaluate the adjacency oracle $Adj(v, (s, r))$ in time $LP(d, \delta)$.*

Proof. The essential part of the evaluation is solving the system (3.6). Since $\delta = |\Delta|$, the system has d variables and at most δ inequalities and the claim follows. ∎

Lemma 3.5 *There is an implementation of the local search function $f(v)$ with evaluation time $O(LP(d, \delta))$, for each $v \in V \setminus \{v^*\}$ with the Minkowski decomposition $v = v_1 + v_2 + \cdots + v_k$.*

Proof. The implementation of f essentially depends on how we define the canonical vector of the normal cone $\mathcal{N}(v; P)$. Like in the adjacency oracle implementation, we use an LP formulation. Since the set of directions $e_j(v, i)$ for valid $(j, i) \in \Delta$ include all edge directions at v, the normal cone $\mathcal{N}(v; P)$ is the set of solutions λ to the system

$$e_j(v_j, i)^T \lambda \leq 0 \quad \text{for all valid } (j, i) \in \Delta.$$

Since we need an interior point of the cone, we formulate the following LP:

$$(3.8) \qquad \begin{array}{lll} \max & \lambda_0 & \\ \text{subject to} & & \\ e_j(v_j, i)^T \lambda + & \lambda_0 \leq 0 & \text{for all valid } (j, i) \in \Delta \\ & \lambda_0 \leq K. & \end{array}$$

Here K is any positive constant. Since v is a vertex of P, this LP has an optimal solution. We still need to define a unique optimal solution. For this, we use a very pragmatic definition: fix one deterministic algorithm and define the canonical vector as the unique solution returned by the algorithm. Since the number of variables is $d+1$ and the number of inequalities is at most $\delta+1$, the assumptions on LP implies the time complexity $O(LP(d, \delta))$ to compute the canonical vector. Note that for practical purposes, we should probably add bounding inequalities for λ to the LP (3.8) such as $-1 \leq \lambda_i \leq 1$ for all i to make sure that the optimal solution stays in a reasonable range. This does not change the complexity.

An execution of f requires to compute the canonical vectors c and c^*. Once they are computed, the remaining part is to determine the first bounding hyperplane of the normal cone $\mathcal{N}(v; P)$ hit by the oriented line $t(\theta) := c + \theta(c^* - c)$ (as θ increases from 0 to 1). This amounts to solving at most δ one-variable equations, and is dominated by the canonical vector computation. ∎

In Figure 2 , we present the resulting reverse search algorithm, where we assume that the δ index pairs (j, i) in Δ are ordered as $(1, 1) < (1, 2) < \cdots < (1, \delta_1) < (2, 1) < \cdots < (k, \delta_k)$. Finally, we are ready to prove the main theorem, Theorem 3.3.

Proof. We use the general complexity result [2, Corollary 2.3] which says the time complexity of the reverse search in Figure 2 is $O(\delta(t(Adj)+t(f))|V|)$ where $t(\cdot)$ denotes the time to evaluate the function $\cdot$. By Lemma 3.4 and Lemma 3.5, both $t(Adj)$ and $t(f)$ can be replaced by $LP(d, \delta)$. Since $f_0(P) = |V|$, the claimed time complexity follows. The space complexity is dominated by those of the functions f and Adj which are clearly linear in the input size. ∎

4 Concluding Remarks

The new algorithm presented here is a natural extension of the arrangement construction algorithm given in [3]. The best way to see is through the dualization of the latter for the zonotope

```
        procedure MinkowskiAddition(Adj,(δ_1, ..., δ_k), v*,f);
            v := v*; (j, i) := (1, 0); (* (j, i): neighbor counter *)
            output v;
            repeat
                while (j, i) < (k, δ_k) do
                    increment (j, i) by one;
(r1)                next := Adj(v, (j, i));
                    if next ≠ 0 then
(r2)                    if f(next) = v then   (* reverse traverse *)
                            v := next;(j, i) := (1, 0);
                            output v
                        endif
                    endif
                endwhile;
                if v ≠ v* then (* forward traverse *)
(f1)                u := v;    v := f(v);
(f2)                restore (j, i) such that Adj(v, (j, i)) = u
                endif
            until v = v* and (j, i) = (k, δ_k).
```

Figure 2: Reverse Search Algorithm

construction problem. It is easy to relate an interior point of a cell of an arrangement of hyperplanes to an interior point of the outer normal cone of the dual zonotope at the vertex corresponding the cell. It might be interesting to think whether there is a natural way to dualize the new algorithm. What could be a problem dual to the Minkowski addition of convex polytopes? What can be a notion extending that of arrangements of hyperplanes?

We presented the algorithm for the polytopal case where input polyhedra are bounded. As long as input polyhedra are pointed (i.e., having at least one extreme point), the essentially the same algorithm works. The only change is an extra treatment of extreme rays in addition to extreme points. Since extreme points can be considered as extreme points at infinity, the treatment is merely cosmetic.

As we see in Proposition 2.6, the output size of Minkowski-addition problems can be very small. This is in strong contrast with the worst-case output result, Theorem 2.5 due to Gritzmann and Sturmfels. It is interesting to study further refinements of the worst output sizes in relation to the input size. For example, if input polytopes are all full dimensional, can the output size be exponential in the input size?

As we already remarked, there is no known efficient algorithm for the Minkowski-addition of $\mathcal{H}$-polytopes that is to list all facets of the Minkowski-addition of polytopes given by facets. Here again, the efficiency is measured as a polynomial function of both the input and output sizes. A variant of the problem is to list all mixed facets only, where a facet is *mixed* if it is the Minkowski addition of some edges of input polytopes. This problem has important applications in generating all solutions to a system of polynomial equations, see [8].

There is a parallel implementation of the algorithm [3] available at [4]. We plan to extend it to the Minkowski-addition problem using the algorithm presented here.

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] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65:21–46, 1996.

[3] J.A. Ferrez, K. Fukuda, and Th. M. Liebling. Solving the fixed rank convex quadratic maximization in binary variables by a parallel zonotope construction algorithm. Preprint, Swiss Federal Institute of Technology, Lausanne, 2002. An earlier version of this paper was titled *Cuts, zonotopes and arrangements*.

[4] K. Fukuda and J.A. Ferrez. Implementations of LP-based reverse search algorithms for the zonotope construction and the fixed-rank convex quadratic maximization in binary variables using the zram and the cddlib libraries, 2002. http://www.cs.mcgill.ca/˜fukuda/download/mink/RS_TOPE020713.tar.gz.

[5] M. Giordano, B. Kataya, and E. Pairel. Tolerance analysis and synthesis by means of clearance and deviation spaces. In *7th CIRP International Seminar on Computer-Aided Tolerancing*. ENS de Cachan, France, April 2001.

[6] P. Gritzmann and B. Sturmfels. Minkowski addition of polytopes: computational complexity and applications to Gröbner bases. *SIAM J. Dics. Math.*, 6:246–269, 1993.

[7] P.D. Seymour. A note on hyperplane generation. *J. Combin. Theory, Series B*, 61:88–91, 1994.

[8] B. Sturmfels. Polynomial equations and convex polytopes. *Amer. Math. Monthly*, 105(10):907–922, 1998.