Journal of Symbolic Computation 73 (2016) 139–152

Contents lists available at ScienceDirect

Journal of Symbolic Computation

www.elsevier.com/locate/jsc

Efficient edge-skeleton computation for polytopes defined by oracles ☆

Ioannis Z. Emiris [a], Vissarion Fisikopoulos [a,1], Bernd Gärtner [b]

[a] Department of Informatics & Telecommunications, University of Athens, Greece
[b] Department of Computer Science, ETH Zurich, Switzerland

ARTICLE INFO

Article history:
Received 12 December 2014
Accepted 4 May 2015
Available online 12 June 2015

Keywords:
General dimension
Polytope oracle
Edge-skeleton
Total polynomial-time
Linear optimization
Vertex enumeration

ABSTRACT

In general dimension, there is no known total polynomial algorithm for either convex hull or vertex enumeration, i.e. an algorithm whose complexity depends polynomially on the input and output sizes. It is thus important to identify problems and polytope representations for which total polynomial-time algorithms can be obtained. We offer the first total polynomial-time algorithm for computing the edge-skeleton—including vertex enumeration—of a polytope given by an optimization or separation oracle, where we are also given a superset of its edge directions. We also offer a space-efficient variant of our algorithm by employing reverse search. All complexity bounds refer to the (oracle) Turing machine model. There is a number of polytope classes naturally defined by oracles; for some of them neither vertex nor facet representation is obvious. We consider two main applications, where we obtain (weakly) total polynomial-time algorithms: Signed Minkowski sums of convex polytopes, where polytopes can be subtracted provided the signed sum is a convex polytope, and computation of secondary, resultant, and discriminant polytopes. Further applications include convex combinatorial optimization and convex

☆ All authors were partially supported from project "Computational Geometric Learning", which acknowledges the financial support of the Future and Emerging Technologies programme within the 7th Framework Programme for research of the European Commission, under FET-Open grant number: 255827. I.Z.E. and V.F. are partially supported by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) – Research Funding Program: THALIS – UOA (MIS 375891).

E-mail addresses: emiris@di.uoa.gr (I.Z. Emiris), vfisikop@ulb.ac.be (V. Fisikopoulos), gaertner@inf.ethz.ch (B. Gärtner).

[1] Present address: Computer Science Department, Universitè libre de Bruxelles CP 216, Boulevard du Triomphe, 1050 Brussels, Belgium.

http://dx.doi.org/10.1016/j.jsc.2015.06.001
0747-7171/

integer programming, where we offer a new approach, thus removing the complexity's exponential dependence in the dimension.

1. Introduction

Convex polytopes are fundamental geometric objects in science and engineering. Their applications are ranging from theoretical computer science to optimization and algebraic geometry. Polytopes in general dimension admit a number of alternative representations. The best known, explicit representations for a bounded polytope P are either the set of its vertices (V-representation) or a bounded intersection of halfspaces (H-representation). Switching between the two representations constitutes the convex hull and vertex enumeration problems. A linear programming problem (LP) on P consists in finding a vertex of P that maximizes the inner product with a given objective vector c. This is very easy if P is in V-representation; even if P is in H-representation, this LP can be solved in polynomial time.

In general dimension, there is no polynomial-time algorithm for either convex hull or vertex enumeration, since the output size can be exponential in the worst case by the upper bound theorem (McMullen, 1971). We therefore wish to take the output size into account and say that an algorithm runs in *total polynomial* time if its time complexity is bounded by a polynomial in the input *and* output size. There is no known total polynomial-time algorithm for either convex hull or vertex enumeration. Avis et al. (1997) identify, for each known type of convex hull algorithm, explicit families of polytopes for which the algorithms run in superpolynomial time.

However, finding the vertices of the convex hull of a given point set reduces to LP and has thus polynomial complexity in the input. The algorithm in (Avis and Fukuda, 1992) solves, in total polynomial-time, vertex enumeration for simple polytopes and convex hull for simplicial polytopes. For 0/1-polytopes a total polynomial-time algorithm for vertex enumeration is presented in (Bussieck and Lübbecke, 1998), where a 0/1-polytope is such that all vertices have coordinates 0 or 1. In this paper we establish another case where total polynomial-time algorithms exist.

An important explicit representation of a polytope is the *edge-skeleton* (or 1-skeleton), which is the graph of polytope vertices and edges, but none of the faces of dimension larger than one. For simple polytopes, the edge-skeleton determines the complete face lattice (see Joswig et al., 2002 and the references therein), but in general, this is false. The edge-skeleton is a useful and compact representation employed in different problems, e.g. in computing general-dimensional Delaunay triangulations of a given pointset: Boissonnat et al. (2009) show how the edge-skeleton suffices for point location by allowing them to recover only the needed full-dimensional simplices of the triangulation. Another application is in mixed volume computation (Malajovich, 2014).

In this paper we study the case where a polytope P is given by an implicit representation, where the only access to P is a black box subroutine (oracle) that solves the LP problem on P for a given vector c. Then, we say that P is given by an *optimization*, or *LP* oracle. Given such an oracle, the entire polytope can be reconstructed, and both V- and H-representations can be found using the Beneath–Beyond method (Emiris et al., 2013; Huggins, 2006), although not in total polynomial-time.

Another important implicit representation of P is obtained through a *separation* oracle (Section 2). Celebrated results of Khachiyan (1979) as well as Grötschel et al. (1993) show that separation and optimization oracles are polynomial-time equivalent (Proposition 2). Many important results in combinatorial optimization use the fact that the separation oracle implies the optimization oracle. In our study, we also need the other direction: Given an optimization oracle, compute a separation oracle for P.

The problem that we study is closely related to vertex enumeration. We are given an optimization oracle for a polytope P and a set of vectors that is guaranteed to contain the directions of all edges of P; edge directions are given by unit vectors. We are asked to compute the edge-skeleton of P so the vertices are also computed. This is similar to the fundamental Minkowski reconstruction problem, e.g. (Gritzmann and Hufnagel, 1999), except that, instead of information on the facets, we have information about the 1-dimensional faces (and an oracle). The problem of the reconstruction of a simple polytope by its edge-skeleton graph is studied in (Joswig et al., 2002).

The most relevant previous work is an algorithm for vertex enumeration of P, given the same input: an optimization oracle and a superset D of all edge directions (Onn and Rothblum, 2007) (cf. Proposition 3). It runs in total polynomial-time in fixed dimension. The algorithm computes the zonotope Z of D, then computes an arbitrary vector in the normal cone of each vertex of Z and calls the oracle with this vector. It outputs all vertices without further information. Computing the edges from n vertices can be done by $O(n^2)$ calls to LP.

1.1. Applications

Edge-skeleton computation given an oracle and a superset of edge directions naturally appears in many applications. In Section 4 we offer new efficient algorithms for the first two applications below.

One application is the *signed Minkowski sum* problem where, besides addition, we also allow a restricted case of *Minkowski subtraction*. Let $A - B$ be polytope C such that A can be written as a sum $A = B + C$. In other words, a signed Minkowski sum equality such as $P - Q + R - S = T$ should be interpreted as $P + R = Q + S + T$. Such sums are motivated by the fact that resultant and discriminant polytopes (to be defined later) are written as signed sums of secondary polytopes (Michiels and Cools, 2000), (Gelfand et al., 1994, Thm. 11.1.3). Also, matroid polytopes and generalized permutahedra can be written as signed Minkowski sums (Ardila et al., 2010). We assume that the summands are given by optimization oracles and the supersets of their edge directions. This is natural since we show that these supersets can be precomputed for resultant and secondary polytopes.

Minkowski sums have been studied extensively. Given r V-polytopes in $\mathbb{R}^d$, Gritzmann and Sturmfels (1993) deal with the various Minkowski sum problems that occur if they regard none, one, or both of r and d as constants. They give polynomial algorithms for fixed d regardless of the input representation. For varying d they show that no polynomial-time algorithm exists except for the case of fixed r in the binary model of computation. Fukuda (2004), extended in (Fukuda and Weibel, 2005), gives an LP-based algorithm for the Minkowski sum of polytopes in V-representation whose complexity, in the binary model of computation, is total polynomial, and depends polynomially on δ, which is the sum of the maximum vertex degree in each summand. However, we are not aware of any algorithm for signed Minkowski sums and it is not obvious how the above algorithms for Minkowski sums can be extended to the signed case.

The second application is resultant, secondary as well as discriminant polytopes. For resultant polytopes at least, the only plausible representation seems to be via optimization oracles (Emiris et al., 2013). *Resultants* are fundamental in computational algebraic geometry since they generalize determinants to nonlinear systems (Sturmfels, 1994; Gelfand et al., 1994). The Newton polytope R of the resultant, or *resultant polytope*, is the convex hull of the exponent vectors corresponding to nonzero terms. A resultant is defined for $k+1$ pointsets in $\mathbb{Z}^k$. If R lies in $\mathbb{R}^d$, the total number of input points is $d + 2k + 1$. If n is the number of vertices in R, typically $n \gg d \gg k$, so k is assumed fixed. A polynomial-time optimization oracle yields an output-sensitive algorithm for the computation of R (Emiris et al., 2013) (cf. Lemma 13).

This approach can also be used for computing the secondary and discriminant polytopes, defined in (Gelfand et al., 1994); cf. (De Loera et al., 2010) on secondary polytopes. The secondary polytope of a pointset is a fundamental object since it offers a realization of the graph of regular triangulations of the pointset. A total polynomial-time algorithm for the secondary polytope when k is fixed is given in (Masada et al., 1996). A specific application of discriminant polytopes is discussed in (Orevkov, 1999), where the author establishes a lower bound on the volume of the discriminant polytope of a multivariate polynomial, thus refuting a conjecture by E.I. Shustin on an asymptotic upper bound for the number of real hypersurfaces.

The size of all these polytopes is typically exponential in d: the number of vertices of R is $O(d^{2d^2})$ (Sturmfels, 1994), and the number of j-faces (for any j) of the secondary polytope is $O(d^{(d-1)^2})$, which is tight if d is fixed (Billera et al., 1990).

More applications of our methods exist. One is in *convex combinatorial optimization*: given $\mathcal{F} \subset 2^N$ with $N = \{1, \ldots, n\}$, a vectorial weighting $w : N \to \mathbb{Q}^d$, and a convex functional $c : \mathbb{Q}^d \to \mathbb{Q}$, find

$F \in \mathcal{F}$ of maximum value $c(w(F))$. This captures a variety of (hard) problems studied in operations research and mathematical programming, including quadratic assignment, scheduling, reliability, bargaining games, and inventory management, see (Onn and Rothblum, 2004) and the references therein. The standard linear combinatorial optimization problem is the special case with $d = 1$, $w : N \to \mathbb{Q}$, and $c : \mathbb{Q} \to \mathbb{Q} : x \mapsto x$ being the identity. As shown in (Onn and Rothblum, 2004), a convex combinatorial optimization problem can be solved in polynomial-time for fixed d, if we know the edge directions of the polytope given by the convex hull of the incidence vectors of the sets in $\mathcal{F}$.

Another application is *convex integer maximization*, where we maximize a convex function over the integer hull of a polyhedron. In (De Loera et al., 2009), the vertex enumeration algorithm of Onn and Rothblum (2007)—based on the knowledge of edge directions—is used to come up with polynomial algorithms for many interesting cases of convex integer maximization, such as multiway transportation, packing, vector partitioning and clustering. A set that contains the directions of all edges is computed via Graver bases, and the enumeration of all vertices of a projection of the integer hull suffices to find the optimal solution.

1.2. Our contribution

We present the first total polynomial-time algorithm for computing the edge-skeleton of a polytope, given an optimization oracle, and a set of directions that contains the polytope's edge directions. The polytope is assumed to have some (unknown) H-representation with an arbitrary number of inequalities, but each of known bitsize, as shall be specified below. Our algorithm also works if the polytope is given by a separation oracle. All complexity bounds refer to the (oracle) Turing machine model, thus leading to (weakly) polynomial-time algorithms when the oracle is of polynomial-time complexity. By employing the reverse search method of (Avis and Fukuda, 1992) we offer a space-efficient variant of our algorithm. It remains open whether there is also a strongly polynomial-time algorithm in the real RAM model of computation.

Our algorithm yields the first (weakly) total polynomial-time algorithms for the edge-skeleton (and vertex enumeration) of signed Minkowski sum, and resultant polytopes (for fixed k). For both polytope classes, optimization oracles are naturally and efficiently constructed, whereas it is not straightforward to obtain the more commonly employed membership or separation oracles. For signed Minkowski sum we assume that we know the supersets of edge directions for summands. This includes the important cases where the summands are V-polytopes, and secondary polytopes. For resultant polytopes, optimization oracles offer the most efficient known representation. Our results on resultant polytopes extend to secondary polytopes, as well as discriminant polytopes. Recall that a different approach in the same complexity class is known for secondary polytopes (Pfeifle and Rambau, 2003; Masada et al., 1996).

Regarding the problems of convex combinatorial optimization and convex integer programming the current approaches use the algorithm of Onn and Rothblum (2007) whose complexity has an exponential dependence on the dimension (Proposition 3). The utilization of our algorithm instead offers an alternative approach while removing the exponential dependence on the dimension.

1.3. Outline

The next section specifies our theoretical framework. Section 3 introduces polynomial-time algorithms for the edge-skeleton. Section 4 applies our results to signed Minkowski sums, as well as resultant and secondary polytopes. We conclude with open questions.

2. Well-described polytopes and oracles

This section describes our theoretical framework and relates the most relevant oracles. We start with the notation used in this paper followed by some basics from polytope theory; for a detailed presentation we refer to (Ziegler, 1995).

We denote by d the ambient space dimension and n the number of vertices of the output (bounded) polytope; k denotes dimension when it is fixed (e.g. input space for resultant polytopes);

conv(A) is the convex hull of A. Moreover, φ is an upper bound for the encoding length of every inequality defining a well-described polytope (see the next section); $\langle X\rangle$ denotes the binary encoding size of an explicitly given object X (e.g., a set of vectors). For a well-described and implicitly given polytope $P \subseteq \mathbb{R}^d$, we will define $\langle P\rangle := d + \varphi$. Let $\mathbb{O} : \mathbb{R} \to \mathbb{R}$ denote a polynomial such that the oracle conversion algorithms of Proposition 2 all run in oracle polynomial-time $\mathbb{O}(\langle P\rangle)$ for a given well-described polytope P. The polynomial $\mathbb{LP} : \mathbb{R} \to \mathbb{R}$ is such that $\mathbb{LP}(\langle A\rangle + \langle b\rangle + \langle c\rangle)$ bounds the runtime of maximizing $c^T x$ over the polyhedron $\{x \mid Ax \le b\}$.

A convex polytope $P \subseteq \mathbb{R}^d$ can be represented as the convex hull of a finite set of points, called the *V-representation* of P. In other words, $P = \text{conv}(A)$, where $A = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^d$. Another, equivalent representation of P is as the bounded intersection of a finite set of halfspaces or linear inequalities, called the *H-representation* of P. That is, $P = \{x \mid Ax \le b\}$, $A \subseteq \mathbb{R}^{m\times d}$, $x \in \mathbb{R}^d$, $b \in \mathbb{R}^m$. Given P, an inequality or a halfspace $\{a^T x \le \beta\}$ (where $a \in \mathbb{R}^d$, $\beta \in \mathbb{R}$) is called *supporting* if $a^T x \le \beta$ for all $x \in P$ and $a^T x = \beta$ for some $x \in P$. The set $\{x \in P \mid a^T x = b\}$ is a *face* of P.

Definition 1. The *polar dual polytope* of P is defined as:

$$P^* := \{c \in \mathbb{R}^d : c^T x \le 1 \text{ for all } x \in P\} \subseteq \mathbb{R}^d,$$

where we assume that the origin $\mathbf{0} \in int(P)$, the relative interior of P, i.e. $\mathbf{0}$ is not contained in any face of P of dimension $< d$.

For our results, we need to assume that the output polytope is *well-described* (Grötschel et al., 1993, Definition 6.2.2). This will be the case in all our applications.

Definition 2. A rational polytope $P \subseteq \mathbb{R}^d$ is *well-described* (with a parameter φ that we need to know explicitly) if there exists an H-representation for P in which every inequality has encoding length at most φ. The *encoding length* of a well-described polytope is $\langle P\rangle = d + \varphi$. Similarly, the *encoding length* of a set of vectors $D \subseteq \mathbb{R}^d$ is $\langle D\rangle = d + \nu$ if every vector in D has encoding length at most ν.

In defining P, the inequalities are not known themselves, and we make no assumptions about their number. The following lemma connects the encoding length of inequalities with the encoding length of vertices.

Lemma 1. (*See* Grötschel et al., 1993, Lemma 6.2.4.) *Let $P \subseteq \mathbb{R}^d$. If every inequality in an H-representation for P has encoding length at most φ, then every vertex of P has encoding length at most $4d^2\varphi$. If every vertex of P has encoding length at most ν, then every inequality of its H-representation has encoding length at most $3d^2\nu$.*

The natural model of computation when P is given by an oracle is that of an *oracle Turing machine* (Grötschel et al., 1993, Section 1.2). This is a Turing machine that can (in one step) replace any input to the oracle (to be prepared on a special oracle tape) by the output resulting from calling the oracle, where we assume that the output size is polynomially bounded in the input size. An algorithm is *oracle polynomial-time* if it can be realized by a polynomial-time oracle Turing machine. Moreover it is *total polynomial-time* if its time complexity is bounded by a polynomial in the input *and* output size.

In this paper, we consider three oracles for polytopes; they can more generally be defined for (not necessarily bounded) polyhedra, but we do not need this:

- *Optimization* ($\text{OPT}_P(c)$): Given vector $c \in \mathbb{R}^d$, either find a point $y \in P$ maximizing $c^T x$ over all $x \in P$, or assert $P = \emptyset$.
- *Violation* ($\text{VIOL}_P(c, \gamma)$): Given vector $c \in \mathbb{R}^d$ and $\gamma \in \mathbb{R}$, either find point $y \in P$ such that $c^T y > \gamma$, or assert that $c^T x \le \gamma$ for all $x \in P$.
- *Separation* ($\text{SEP}_P(y)$): Given point $y \in \mathbb{R}^d$, either certify that $y \in P$, or find a hyperplane that separates y from P; i.e. find vector $c \in \mathbb{R}^d$ such that $c^T y > c^T x$ for all $x \in P$.

The following is a main result of Grötschel et al. (1993) and the cornerstone of our method.

Proposition 2. (*See* Grötschel et al., 1993, Theorem 6.4.9.) *For a well-described polytope, any one of the three aforementioned oracles is sufficient to compute the other two in oracle polynomial-time. The runtime (polynomially) depends on the ambient dimension d and the bound φ for the maximum encoding length of an inequality in some H-representation of P.*

For applications in combinatorial optimization, an extremely important feature is that the runtime does not depend on the number of inequalities that are needed to describe P. Even if this number is exponential in d, the three oracles are polynomial-time equivalent.

Another important corollary is that linear programs can be solved in polynomial-time. Indeed, an explicitly given (bounded coefficient) system $Ax \le b$, $x \in \mathbb{R}^d$ of inequalities defines a well-described polytope P, for which the separation oracle is very easy to implement in time polynomial in $\langle P \rangle$; hence, the oracle polynomial-time algorithm for $\mathrm{OPT}_P(c)$ becomes a (proper) polynomial-time algorithm.

3. Computing the edge-skeleton

This section studies total polynomial time algorithms for the edge-skeleton of a polytope. We are given a well-described polytope $P \subseteq \mathbb{R}^d$ via an optimization oracle $\mathrm{OPT}_P(c)$ of P. Moreover, we are given a superset D of all edge directions of P; to be precise, we define

$$D(P) := \left\{ \frac{v - w}{\|v - w\|} : v \text{ and } w \text{ are adjacent vertices of } P \right\}$$

to be the set of (unit) edge directions, and we assume that for every $e \in D(P)$, the set D contains some positive multiple te, $t \in \mathbb{R}$, $t > 0$. Slightly abusing notation, we write $D \supseteq D(P)$.

The goal is to efficiently compute the edge-skeleton of P, i.e. its vertices and the edges connecting the vertices. Even if $D = D(P)$, this set does not, in general, provide enough information for the task, so we need additional information about P; here we assume an optimization oracle.

Vertex enumeration with this input has been studied in the real RAM model of computation where we count the number of arithmetic operations:

Proposition 3. (*See* Onn and Rothblum, 2007.) *Let $P \subseteq \mathbb{R}^d$ be a polytope given by $\mathrm{OPT}_P(c)$, and let $D \supseteq D(P)$ be a superset of the edge directions of P. The vertices of P are computed using $O(|D|^{d-1})$ arithmetic operations and $O(|D|^{d-1})$ calls to $\mathrm{OPT}_P(c)$.*

If P has n vertices, then $|D(P)| \le \binom{n}{2}$, and this is tight for neighborly polytopes in general position (Ziegler, 1995). This means that the bound of Proposition 3 is $O\left(n^{2d-2}\right)$, assuming that $|D| = \Theta(|D(P)|)$.

We show below that the edge-skeleton can be computed in oracle total polynomial-time for a well-described polytope, which possesses an (unknown) H-representation with encoding size φ. Thus, we show that the exponential dependence on d in Proposition 3 can be removed in the Turing machine model of computation, leading to a (weakly) total polynomial-time algorithm. It remains open whether there is also a strongly total polynomial-time algorithm with a total polynomial runtime bound in the real RAM model of computation.

The algorithm (Algorithm 1) is as follows. Using $\mathrm{OPT}_P(c)$, we find some vertex v_0 of P (this can be done even if $\mathrm{OPT}_P(c)$ does not directly return a vertex Grötschel et al., 1993, Lemma 6.51, Edmonds et al., 1982, pp. 255–256).

We maintain sets V_P, E_P of vertices and their incident edges, along with a queue $W \subseteq V_P$ of the vertices for which we have not found all incident edges yet. Initially, $W = \{v_0\}$, $V_P = E_P = \emptyset$. When we process the next vertex v from the queue, it remains to find its incident edges: equivalently, the neighbors of v. To find the neighbors, we first build a set V_{cone} of candidate vertices. We know that for every neighbor w of v, there must be an edge direction e such that $w = v + te$ for suitable $t > 0$. More precisely, w is the point corresponding to maximum t in the 1-dimensional polytope $Q(e) := P \cap \{x \mid x = v + te, t \ge 0\}$, where the latter equals the intersection of P with the ray in

Algorithm 1: Edge_Skeleton (OPT_P, D)

Input : Optim. oracle $\mathrm{OPT}_P(c)$, superset D of edge directions $D(P)$
Output: The edge-skeleton (and vertices) of P

```
Compute some vertex v_0 ∈ P;
V_P ← ∅; W ← {v_0}; E_P ← ∅;
while W ≠ ∅ do
    Choose the next element v ∈ W and remove it from W;
    V_P ← V_P ⋃ {v};
    V_cone ← ∅;
    foreach e ∈ D do
        w ← argmax{v + te ∈ P, t ≥ 0};
        if w ≠ v then
            V_cone ← V_cone ⋃ {w};
    Remove non-vertices of P from V_cone;
    foreach w ∈ V_cone do
        if w ∉ V_P then W ← W ⋃ {w};
        if {v, w} ∉ E_P then E_P ← E_P ⋃ {v, w};
return V_P, E_P;
```

direction e and apex at v. Hence, by solving $|D|$ linear programs, one for every $e \in D$, we can build a set V_{cone} that is guaranteed to contain all neighbors of v. To solve these linear programs, we need to construct optimization oracles for $Q(e)$. To do this, we first construct $\mathrm{SEP}_P(y)$ from $\mathrm{OPT}_P(c)$ in oracle polynomial-time according to Proposition 2. Thus, the construction of $\mathrm{SEP}_{Q(e)}(y)$ is elementary, and since also $Q(e)$ is well-described, we can obtain $\mathrm{OPT}_{Q(e)}(c)$ in oracle polynomial-time.

In a final step, we remove the candidates that do not yield neighboring vertices. For this, we first solve a linear program to compute a hyperplane h separating v from the candidates in V_{cone}; since V_{cone} is a finite subset of $P \setminus \{v\}$, such a hyperplane exists, and w.l.o.g. $v = \mathbf{0}$ and $h = \{x \mid x_d = 1\}$. Let C be the cone generated by the set V_{cone}. We compute the extreme points of $C \cap \{x \mid x_d = 1\}$, giving us the extremal rays of C. Note that $C \cap \{x \mid x_d = 1\}$ is called the vertex figure of P in v. Finally, we remove every point from V_{cone} that is not on an extremal ray.

The correctness of the algorithm relies on the following lemma.

Lemma 4. *Let v be a vertex of P processed during Algorithm 1, where we assume w.l.o.g. that $v = \mathbf{0}$ and the set V_{cone} of candidates is separated from v by the hyperplane $\{x \mid x_d = 1\}$.*

A point $w \in \mathbb{R}^d$ is a neighbor of v if and only if w is on some extremal ray of the cone C generated by V_{cone}. Here, an extremal ray is a ray whose intersection with the hyperplane $\{x_d = 1\}$ is an extreme point of the polytope $C \cap \{x \mid x_d = 1\}$.

Proof. Suppose that w is a neighbor of v. By construction, $w \in V_{cone}$ and $\{v, w\}$ is an edge. The thesis follows by the following well known fact about vertex figures. There is a bijection between edges of P that contain v and extreme points of $C \cap \{x \mid x_d = 1\}$ (Ziegler, 1995, Proposition 2.4). □

We now bound the *time complexity* of Algorithm 1.

Theorem 5. *Given OPT_P and a superset of edge directions D of a well-described polytope $P \subseteq \mathbb{R}^d$ with n vertices, and m edges Algorithm 1 computes the edge-skeleton of P in oracle total polynomial-time*

$$O\left(n|D|\left(\mathbb{O}(\langle P\rangle + \langle D\rangle) + \mathbb{LP}(d^3|D|(\langle P\rangle + \langle D\rangle)) + d\log n\right)\right),$$

where $\langle D\rangle$ is the binary encoding length of the vector set D.

Proof. We call $\mathrm{OPT}_P(x)$ to find the first vertex of P. Then, there are $O(n)$ iterations. In each one, we construct $O(|D|)$ oracles for polytopes $Q(e)$ of encoding length at most $\langle P\rangle + \langle D\rangle$. We also compute the (at most n) extreme points from a set of at most $|D|$ candidate points. This can be done by solving $|D|$ linear programs whose inequalities have coefficients that are in turn coordinates of vertices of the $Q(e)$'s. By Lemma 1, these coordinates have encoding lengths bounded by $4d^2(\langle P\rangle + \langle D\rangle)$, and the number of coefficients in each linear program is $O(|D|d)$. At each vertex we have to test whether the computed vertices and edges are new. In the course of the algorithm these tests are at most $O(m) = O(n|D|)$, where m the number of P edges. We can test whether a vertex (or an edge) is new in $O(d\log n)$. $\square$

3.1. Reverse search for edge-skeleton

We define a reverse search procedure based on (Avis and Fukuda, 1992) to optimize the space used by Algorithm 1. Given a vertex of P, the set of adjacent edges can be constructed as described above. Then we need to define a total order over the vertices of the polytope. Any generic vector $c \in \mathbb{R}^d$ induces such an order on the vertices, i.e. the order of a vertex u is that of $c^T u$. In other words, we can define a reverse search tree on P with root the vertex v that maximizes $c^T v$ over all the vertices of P, where c is the vector given to OPT_P for initializing P. Technically, the genericity assumption on c can be avoided by sorting the vertices w.r.t. the lexicographical ordering of their coordinates.

Reverse search also needs an *adjacency procedure* which, given a vertex v and an integer j, returns the j-th adjacent vertex of v, as well as a *local search procedure* allowing us to move from any vertex to its optimal neighbor w.r.t. the objective function. Both procedures can be implemented by computing all the adjacent vertices of a given vertex of P as described above, and then returning the best (or the j-th) w.r.t. the ordering induced by c.

The above procedures can be used by a reverse search variant of Algorithm 1 that traverses (forward and backward) the reverse search tree while keeping in memory only a constant number of P vertices and edges. On the contrary, both the original Algorithm 1 and the algorithm of Proposition 3 need to store all vertices of P whose number is exponential in d in the worst case. Note that any algorithm should use space at least $O(d|D|)$ to store the input set of edge directions. The above discussion yields the following result (encoding length of P vertices comes from Lemma 1).

Corollary 6. *Given* OPT_P *and a superset of edge directions* D*, a variant of Algorithm 1 using reverse search runs in space* $O(4d^2\langle P\rangle + \langle D\rangle)$ *(additional to the input) while keeping the same asymptotic time complexity.*

4. Applications

This section studies the performance of Algorithm 1 in certain classes of polytopes where we do not assume that we know the set of edge directions a priori. To this end, we describe methods for pre-computing a (super)set of the edge directions.

We start by describing the computation of the set of edge-directions in arbitrary polytopes using the formulation of standard polytopes. Then we study two important classes of polytopes where the number of edge directions can be efficiently precomputed and thus provide new, total polynomial-time algorithms for their representation by an edge-skeleton. In particular, we study signed Minkowski sums, and resultant and secondary polytopes. We show that these polytopes are well-described and naturally defined by optimization oracles, which provide a compact representation.

4.1. Standard polytopes

First we discuss the performance of Algorithm 1 on general polytopes. Any convex polytope $P = \{x \mid Ax \le b\}$ can be written as a linear projection of a polytope $Q = \{(x, y) \mid Ax + Iy = b, y \ge 0\}$, where $A \subseteq \mathbb{R}^{m\times d}$ and we introduce the slack variables $y \in \mathbb{R}^m$ and $P = \pi(Q)$, by the linear mapping $\pi(x, y) = x$. We can rewrite Q as the so-called *standard polytope* $\{x' \mid Bx' = b, x' \ge 0\}$. The set E of edge directions of Q is covered by the set of circuits of B (cf. Lemma 2.13

in Onn and Rothblum, 2004). Moreover, each edge direction of P is the projection of some direction in E under the mapping π (cf. Lemma 2.4 in Onn and Rothblum, 2004). However, the number of circuits of B will be exponential in d. On the other hand, for small dimensions Algorithm 1 could be an efficient choice for edge-skeleton computation or vertex enumeration.

4.2. Signed Minkowski sums

Recall that the *Minkowski sum* of (convex) polytopes $A, B \subseteq \mathbb{R}^d$ is defined as

$$A + B := \{a + b \mid a \in A, b \in B\}.$$

Following Schneider (1993) the *Minkowski subtraction* is defined as

$$A - B := \{x \in \mathbb{R}^d \mid B + x \subseteq A\}.$$

Here we consider a special case of Minkowski subtraction where B is a summand of A. Equivalently, if $A - B = C$ then $A = B + C$. A *signed Minkowski sum* combines Minkowski sums and subtractions, namely

$$P = s_1 P_1 + s_2 P_2 + \cdots + s_r P_r,\ s_i \in \{-1, 1\},$$

where all P_i are convex polytopes and so is P.

We also define the sum (or subtraction) of two optimization oracles as the Minkowski sum (or subtraction) of the resulting vertices. In particular, if $\mathrm{OPT}_P(c) = v$ and $\mathrm{OPT}_{P'}(c') = v'$ for v, v' vertices of P, P' respectively, then $\mathrm{OPT}_P(c) + \mathrm{OPT}_{P'}(c) = v + v'$ and $\mathrm{OPT}_P(c) - \mathrm{OPT}_{P'}(c) = v - v'$. An optimization oracle for the signed Minkowski sum is given by the signed sum of the optimization oracles of the summands.

Lemma 7. *If $P_1, \ldots, P_r \subset \mathbb{R}^d$ are given by optimization oracles, then we compute an optimization oracle for signed Minkowski sum $P = \sum_{i=1}^r s_i P_i$ in $O(r)$.*

Proof. Assume w.l.o.g. that $s_1 = \cdots = s_k = 1 \neq s_{k+1} = \cdots = s_r = -1$. Then, given $P = \sum_{i=1}^r s_i P_i$ and denoting $P_1 := \sum_{i=k+1}^r P_i$ and $P_2 := \sum_{i=1}^k P_i$ we have $P + P_1 = P_2$. It follows that $\mathrm{OPT}_{P+P_1}(c) = \mathrm{OPT}_{P_2}(c)$ for some vector $c \in \mathbb{R}^d$. Then $\mathrm{OPT}_P(c) + \mathrm{OPT}_{P_1}(c) = \mathrm{OPT}_{P_2}(c)$ which follows from Minkowski sum properties. Therefore, we can compute $\mathrm{OPT}_P(c) = \sum_{i=1}^r s_i \mathrm{OPT}_{P_i}(c)$ with r oracle calls to OPT_{P_i} for $i = 1, \ldots, r$. This yields a complexity of $O(r)$ for OPT_P since, by definition of oracle polynomial-time, the oracle calls in every summand are of unit cost. □

Example 1. Here we illustrate the above definitions and constructions as well as the standard reductions from (Grötschel et al., 1993). Consider the polytopes P_1, P_2, P_3, their signed Minkowski sum $P = P_1 - P_2 + P_3$, and its polar P^* as shown in Fig. 1. Observe that $P_1 = P_2 + S$, where S is a square. Assume that P_1, P_2, P_3 are given by OPT_{P_1}, OPT_{P_2}, OPT_{P_3} oracles.

Then, $\mathrm{OPT}_P(c) = \mathrm{OPT}_{P_1}(c) - \mathrm{OPT}_{P_2}(c) + \mathrm{OPT}_{P_3}(c)$ for some vector $c \in \mathbb{R}^d$. If P satisfies the requirements of Proposition 2 then, having access to $\mathrm{OPT}_P(c)$, we compute $\mathrm{SEP}_P(p)$ in oracle polynomial-time for point $p \in \mathbb{R}^d$. In particular, asking if $p \in P$ is equivalent to asking if $H := \{x \mid p^T x \leq 1\}$ is a valid inequality for P^*. The latter can be solved by computing the point c^T in P^* that maximizes the inner product with the outer normal vector of H and test if it validates H. If this happens then $\mathrm{SEP}_P(p)$ returns that $p \in P$, otherwise it returns $p \notin P$ with separating hyperplane $\{x \mid cx = 1\}$.

Let n denote the number of vertices of P. An oracle for P is provided by Lemma 7. Then, the entire polytope can be reconstructed, and both V- and H-representations can be found by Proposition 8.

Proposition 8. (*See* Emiris et al., 2013.) *Given OPT_P for $P \subseteq \mathbb{R}^d$, its V- and H-representations as well as a triangulation T of P can be computed in*

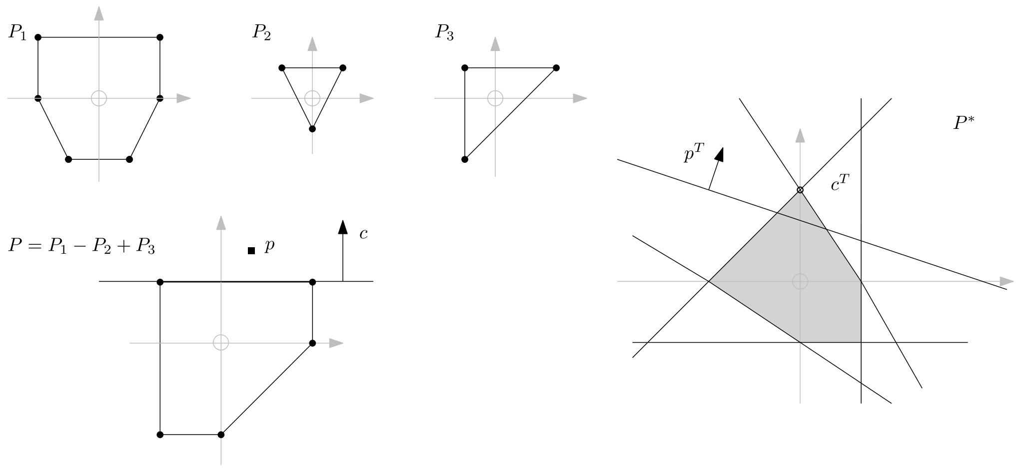

Fig. 1. Signed Minkowski sum oracles.

$$O(d^5 n|T|^2) \text{ arithmetic operations, and } O(n+f) \text{ calls to } \mathrm{OPT}_P,$$

where n and f are the number of vertices and facets of P, respectively, and $|T|$ the number of d-dimensional simplices of T.

Corollary 9. *Given optimization oracles for $P_1, \ldots, P_r \subseteq \mathbb{R}^d$, we construct the V- and H-representations, and a triangulation T of the signed Minkowski sum $P = P_1 + s_2 P_2 + \cdots + s_r P_r$, $s_i \in \{-1, 1\}$ in output sensitive complexity, namely $O(d^5 n|T|^2 + (n+f)r)$, where n, f are the number of vertices and facets in P and $|T|$ the number of full-dimensional simplices of T.*

In the above complexity the number of d-dimensional simplices of the computed triangulation T can be exponential in d which is essentially given by the Upper Bound Theorem for spheres, i.e. $|T| = O(n^{(d+1)/2})$ (Stanley, 2004). This stresses the need for total polynomial-time algorithms for the edge-skeleton of P. Note that it is not assumed that the polytopes are well-described. But we assume the input contains a superset of all edges for each P_i. In one of the most important cases where we are given the vertices of all summands P_i, we can compute all edges in each P_i by solving a linear program for each pair of vertices. Each such pair defines a candidate edge. Hence, the overall computation of the edges of P_i's is polynomial.

Corollary 10. *Given optimization oracles for well-described $P_1, \ldots, P_r \subseteq \mathbb{R}^d$, and supersets of their edge directions $D_1, \ldots, D_r$, the edge-skeleton of the signed Minkowski sum P can be computed in oracle total polynomial-time by Algorithm 1.*

Proof. To be able to apply Algorithm 1, first we should show that P is well-described. Let $\langle P_{max} \rangle$ be the maximum encoding length of summands $P_1, \ldots, P_r$. Then by Lemma 1, the encoding length of the coordinates of summand vertices is $4d^2 \langle P_{max} \rangle$. Thus, $4d^2 \langle P_{max} \rangle + \langle r \rangle$ is the encoding length of the coordinates of P vertices. Finally, $\langle P \rangle = d + 12d^4 \langle P_{max} \rangle + 3d^2 \langle r \rangle$ by Lemma 1. Now OPT_P is computed by Lemma 7 in $O(r)$. The superset of the edge directions of P is $D = \bigcup_{s_i > 0} D_i$, because $D(P_1 - P_2) \subset D(P_1)$ since $P_1 - P_2 = P_3 \Leftrightarrow P_1 = P_2 + P_3$. $\square$

Our algorithm assumes that, in the Minkowski subtraction $A - B$, B is a summand of A and does not verify this assumption.

4.3. Secondary and resultant polytopes

The *secondary polytope* Σ of a set of d points $A = \{p_1, \dots, p_d\} \subset \mathbb{Z}^k$ is a fundamental object since it expresses the triangulations of conv(A) via a polytope representation. For any triangulation T of conv(A), define vector $\phi_T \in \mathbb{R}^d$ with i-coordinate

$$\phi_T(i) = \sum_{\sigma \in T \,\mid\, p_i \in \mathrm{vtx}(\sigma)} \mathrm{vol}(\sigma), \tag{1}$$

summing over all simplices σ of T having p_i as a vertex, where vtx(σ) is the vertex set of simplex σ, and $i \in \{1, \dots, d\}$. Now the secondary polytope $\Sigma(\mathcal{A})$, or just Σ, is defined as the convex hull of ϕ_T for all triangulations T. A famous theorem of Gelfand et al. (1994), which is also the central result in (De Loera et al., 2010), states that there is a bijection between the vertices of Σ and the regular triangulations of conv(A). This extends to a bijection between the face poset of Σ and the poset of regular subdivisions of conv(A). Moreover, Σ, although in ambient space $\mathbb{R}^d$, has actual dimension $\dim(\Sigma) = d - k - 1$.

Let us now consider the Newton polytope of resultants, or *resultant polytopes*, for which optimization oracles provide today the only plausible approach for their computation (Emiris et al., 2013).

Let us consider sets $A_0, \dots, A_k \subset \mathbb{Z}^k$. In the algebraic setting, these are the supports of $k + 1$ polynomials in k variables. Let the Cayley set be defined by

$$A := \bigcup_{j=0}^{k} (A_j \times \{e_j\}) \subset \mathbb{Z}^{2k},$$

where $e_0, \dots, e_k$ form an affine basis of $\mathbb{Z}^k$. Clearly, each point in A corresponds to a unique point in some A_i. The (regular) triangulations of A are in bijective correspondence with the (regular) fine mixed subdivisions of the Minkowski sum $A_0 + \cdots + A_k$ (Gelfand et al., 1994). Mixed subdivisions are those where all cells are Minkowski sums of convex hulls of subsets of the A_i. A mixed subdivision is fine if, for every cell, the sum of its summands' dimensions equals the dimension of the cell.

Let $d := \sum_{j=0}^{k} |A_j|$, then given triangulation T of conv(A), define vector $\rho_T \in \mathbb{R}^d$ with i-coordinate

$$\rho_T(i) := \sum_{i\text{-mixed } \sigma \in T} \mathrm{vol}(\sigma), \tag{2}$$

where $i \in \{1, \dots, d\}$. A simplex σ is called *i-mixed* if it contains $p_i \in A_\ell$ for some $\ell \in \{1, \dots, k\}$ and exactly 2 points from each A_j, where j ranges over $\{0, 1, \dots, k\} - \{\ell\}$. The *resultant polytope* R is defined as the convex hull of ρ_T for all triangulations T. Similarly with the secondary polytope, it is in ambient space $\mathbb{R}^d$ but has dimension $\dim(R) = d - 2k - 1$ (Gelfand et al., 1994). There is a surjection, i.e. many to one relation, from the regular triangulations of conv(A) to the vertices of R.

Example 2. Let $A_0 = \{\{0\}, \{2\}\}$, $A_1 = \{\{0\}, \{1\}, \{2\}\}$, then the Cayley set will be $A = \{\{0, 0\}, \{2, 0\}, \{0, 1\}, \{1, 1\}, \{2, 1\}\}$. The 5 vertices of the secondary polytope $\Sigma(A)$ are computed using equation (1):

$$\phi(T_1) = (2, 4, 2, 0, 4),$$

$$\phi(T_2) = (4, 2, 4, 0, 2),$$

$$\phi(T_3) = (4, 2, 3, 2, 1),$$

$$\phi(T_4) = (3, 3, 1, 4, 1),$$

$$\phi(T_5) = (2, 4, 1, 2, 3),$$

and the 3 vertices of the resultant polytope $N(R)$ are computed using equation (2):

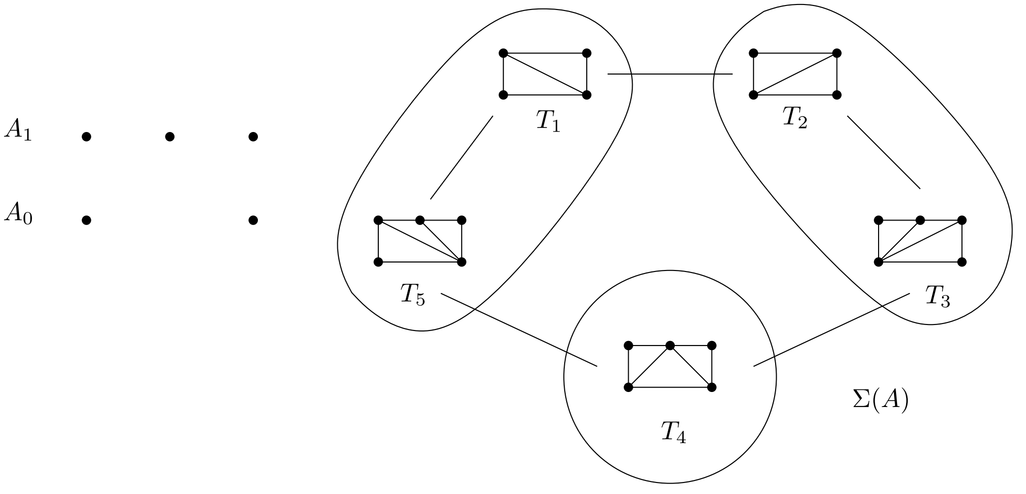

Fig. 2. Secondary and resultant polytopes.

$$\rho(T_1) = (0, 2, 0, 0, 2),$$
$$\rho(T_2) = (2, 0, 2, 0, 0),$$
$$\rho(T_3) = (2, 0, 2, 0, 0),$$
$$\rho(T_4) = (1, 1, 0, 2, 0),$$
$$\rho(T_5) = (0, 2, 0, 0, 2).$$

Note that there are two pairs of triangulations that yield one resultant vertex each. Fig. 2 illustrates this example.

We consider k fixed because in practice it holds $k \ll d \ll n$, where n stands for the number of polytope vertices. Note that R is computed as a full-dimensional polytope in a space of its intrinsic dimension (Emiris et al., 2013) and this approach extends to Σ.

Computing the V-representation of Σ and R by the algorithm in (Emiris et al., 2013) is not total polynomial. In fact, the complexity depends on the number of polytope vertices and facets, but also on the number of simplices in a triangulation of the polytope (see Proposition 8). However, we show that Algorithm 1 computes Σ and R in oracle total polynomial-time.

Lemma 11. *Both Σ and R are well-described polytopes.*

Proof. For the case of Σ, given $A \in \mathbb{Z}^k$, let $\langle A \rangle$ be its encoding length and $\alpha := \text{vol}(\text{conv}(A))$. It is clear that $\alpha = O(\langle A \rangle^k)$ and thus $\langle \alpha \rangle = O(k\langle A \rangle)$. For each triangulation T each coordinate of ϕ_T is upper bounded by α, since the sum of the volumes of its adjacent simplices cannot exceed vol(conv(A)). This bound is tight for the points $a \in A$ of a regular triangulation T where the simplices containing a partition conv(A). It follows that the encoding length of Σ vertices is $\langle \alpha \rangle$ and thus $\langle \Sigma \rangle = 4n^2 \langle \alpha \rangle + d = O(dn^2 \langle A \rangle)$ by Lemma 1. Similarly, we bound the encoding length of ρ_T which yields that R is also a well-described polytope. □

In the sequel, we characterize the set of edge directions of Σ and R. The edge directions of both Σ, R can be computed by enumerating circuits of A. More specifically, circuit enumeration suffices to compute the *edge vectors*, i.e. both directions and lengths of the edges.

We first give some fundamental definitions from combinatorial geometry. For a detailed presentation we recommend (De Loera et al., 2010). A *circuit* $C \subseteq A$ is a minimum affinely dependent subset of A. It holds that $\text{conv}(C)$ has exactly two triangulations C_+, C_-. The operation of switching from one triangulation to another is called *flip*. Triangulation T of A, which equals C_+ when restricted on circuit C, is supported on C if, by flipping C_+ to C_-, we obtain another triangulation T' of A. The dimension of a circuit is the dimension of its convex hull. If A is in *generic position*, then all circuits C are full dimensional. Then all the edges of Σ correspond to full dimensional circuits. If A is *not* in generic position, some edges may correspond to lower-dimensional circuits.

In the case of R, where $A = \bigcup_{j=0}^{k} A_j$, a circuit C is called *cubical* if and only if $|C \cap A_j| \in \{0, 2\}$, $j = 0, \ldots, k$. If A is in *generic position*, all the edges of R correspond to full dimensional cubical circuits (Sturmfels, 1994).

Lemma 12. *Given $A \in \mathbb{Z}^k$ in generic position, we compute the set of edge directions of Σ in $O(d^{k+2})$. Given $A \in \mathbb{Z}^{2k}$ in generic position the set of edge directions of R is computed in $O(d^{2k+2})$. In both cases, genericity of A is checked within the respective time complexity.*

Proof. For Σ, we enumerate all $\binom{|A|}{k+2}$ circuits in $O(d^{k+2})$, obtaining the set of all edge vectors. Genericity of A is established by checking whether all $\binom{|A|}{k}$ subsets, $k \in \{1, \ldots, k+1\}$, are independent. This is in $O(d^{k+1})$ for $k = O(1)$.

In the case of R, where $A = \bigcup_{j=0}^{k} A_j$, a flip on T is cubical if and only if it is supported on a cubical circuit C. In generic position, $|C| = 2k + 2$. For those supporting cubical flips, $|C \cap A_j| = 2$, $j = 0, \ldots, k$. Every edge d_C of R is supported on cubical flip C, where $d_C(a)$ equals $\rho_{C_+}(a) - \rho_{C_-}(a)$, if $a \in C$, and 0 otherwise (Sturmfels, 1994). Given A, all such circuits are enumerated in $\binom{|A|}{2k+2} = O(d^{2k+2})$; a better bound is $O(t^{2k+2})$ if t bounds $|A_j|$, $j = 0, \ldots, k$. □

Lemma 13. (*See* Emiris et al., 2013.) *For $k + 1$ pointsets in $\mathbb{Z}^k$ of total cardinality d, optimization over R takes polynomial-time, when k is fixed.*

Corollary 14. *In total polynomial-time, we compute the edge-skeleton of $\Sigma \subset \mathbb{R}^d$, given $A \in \mathbb{Z}^k$ in generic position, and the edge-skeleton of R, given $A \in \mathbb{Z}^{2k}$ in generic position.*

Proof. Since by Lemma 11 Σ, R are well-bounded, optimization oracles are available by Lemma 13 and the set of edge directions by Lemma 12, the edge-skeletons of Σ, R can be computed by Algorithm 1 in oracle total polynomial-time. Moreover, since the optimization oracle is polynomial-time this yields a (proper) total polynomial-time algorithm for Σ, R. □

Following Lemma 12, for Σ, R we also obtain their edge lengths. This can lead to a more efficient edge-skeleton algorithm on the real RAM.

Remark 1. Our results readily extend to the Newton polytope of discriminants, or discriminant polytopes. This follows from the fact that these polytopes can be written as signed Minkowski sums of secondary polytopes (Gelfand et al., 1994).

5. Concluding remarks

We have presented the first total polynomial-time algorithm for computing the edge-skeleton of a polytope, given an optimization oracle, and a set of directions that contains the polytope's edge directions. Our algorithm yields the first (weakly) total polynomial-time algorithms for the edge-skeleton (and vertex enumeration) of signed Minkowski sum, and resultant polytopes.

An open question is a *strongly* total polynomial-time algorithm for the edge-skeleton problem. Another is to solve the edge-skeleton problem without edge directions; characterizations of edge directions for polytopes in H-representation are studied in (Onn et al., 2005). It is also interesting to investigate new classes of convex combinatorial optimization problems where our algorithm runs in polynomial time.

Acknowledgments

The authors thank K. Fukuda, C. Müller, S. Stich for discussions and bibliographic suggestions.

References

Ardila, F., Benedetti, C., Doker, J., 2010. Matroid polytopes and their volumes. Discrete Comput. Geom. 43 (4), 841–854.
Avis, D., Bremner, D., Seidel, R., 1997. How good are convex hull algorithms? Comput. Geom. 7, 265–301.
Avis, D., Fukuda, K., 1992. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. Discrete Comput. Geom. 8, 295–313.
Billera, L., Filliman, P., Sturmfels, B., 1990. Constructions and complexity of secondary polytopes. Adv. Math. 83 (2), 155–179.
Boissonnat, J.-D., Devillers, O., Hornus, S., 2009. Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. In: ACM Symp. on Comp. Geometry, pp. 208–216.
Bussieck, M., Lübbecke, M., 1998. The vertex set of a 0/1-polytope is strongly p-enumerable. Comput. Geom. 11 (2), 103–109.
De Loera, J., Hemmecke, R., Onn, S., Rothblum, U., Weismantel, R., 2009. Convex integer maximization via Graver bases. J. Pure Appl. Algebra 213 (8), 1569–1577.
De Loera, J., Rambau, J., Santos, F., 2010. Triangulations: Structures for Algorithms and Applications. Algorithms and Computation in Mathematics, vol. 25. Springer.
Edmonds, J., Pulleyblank, W., Lovász, L., 1982. Brick decompositions and the matching rank of graphs. Combinatorica 2 (3), 247–274.
Emiris, I., Fisikopoulos, V., Konaxis, C., Peñaranda, L., 2013. An oracle-based, output-sensitive algorithm for projections of resultant polytopes. Int. J. Comput. Geom. Appl. 23, 397–423.
Fukuda, K., 2004. From the zonotope construction to the Minkowski addition of convex polytopes. J. Symb. Comput. 38.
Fukuda, K., Weibel, C., 2005. Computing all faces of the Minkowski sum of V-polytopes. In: Canad. Conf. Comp. Geom., pp. 253–256.
Gelfand, I., Kapranov, M., Zelevinsky, A., 1994. Discriminants, Resultants and Multidimensional Determinants. Birkhäuser, Boston.
Gritzmann, P., Hufnagel, A., 1999. On the algorithmic complexity of Minkowski's reconstruction problem. J. Lond. Math. Soc. 2, 5–9.
Gritzmann, P., Sturmfels, B., 1993. Minkowski addition of polytopes: computational complexity and applications to Gröbner bases. SIAM J. Discrete Math. 6 (2), 246–269.
Grötschel, M., Lovász, L., Schrijver, A., 1993. Geometric Algorithms and Combinatorial Optimization, 2nd edition. Algorithms and Combinatorics, vol. 2. Springer.
Huggins, P., 2006. ib4e: a software framework for parametrizing specialized LP problems. In: Iglesias, A., Takayama, N. (Eds.), Mathematical Software – ICMS. In: Lect. Notes Comput. Sci., vol. 4151. Springer, pp. 245–247.
Joswig, M., Kaibel, V., Körner, F., 2002. On the k-systems of a simple polytope. Isr. J. Math. 129 (1), 109–117.
Khachiyan, L., 1979. A polynomial algorithm in linear programming. Sov. Math. Dokl. 20 (1), 191–194.
Malajovich, G., 2014. Computing mixed volume and all mixed cells in quermassintegral time. ArXiv e-prints.
Masada, T., Imai, H., Imai, K., 1996. Enumeration of regular triangulations. In: ACM Symp. on Comp. Geometry. SoCG '96, pp. 224–233.
McMullen, P., 1971. The maximum numbers of faces of a convex polytope. Mathematika 17, 179–184.
Michiels, T., Cools, R., 2000. Decomposing the secondary Cayley polytope. Discrete Comput. Geom. 23, 367–380.
Onn, S., Rothblum, U., 2004. Convex combinatorial optimization. Discrete Comput. Geom. 32 (4), 549–566.
Onn, S., Rothblum, U., 2007. The use of edge-directions and linear programming to enumerate vertices. J. Comb. Optim. 14, 153–164.
Onn, S., Rothblum, U., Tangir, Y., 2005. Edge-directions of standard polyhedra with applications to network flows. J. Glob. Optim. 33 (1), 109–122.
Orevkov, S., 1999. The volume of the Newton polytope of a discriminant. Russ. Math. Surv. 54 (5), 1033–1034.
Pfeifle, J., Rambau, J., 2003. Computing triangulations using oriented matroids. In: Joswig, M., Takayama, N. (Eds.), Algebra, Geometry and Software Systems. Springer, Berlin, Heidelberg, pp. 49–75.
Schneider, R., 1993. Convex Bodies: The Brunn–Minkowski Theory. Cambridge University Press.
Stanley, R., 2004. Combinatorics and Commutative Algebra. Birkhäuser, Boston.
Sturmfels, B., 1994. On the Newton polytope of the resultant. J. Algebr. Comb. 3, 207–236.
Ziegler, G., 1995. Lectures on Polytopes. Springer.