Project number IST-25582

CGL

Computational Geometric Learning

Efficient edge skeleton computation for polytopes defined by oracles

STREP

Information Society Technologies

Period covered:	November 1, 2012 – October 31, 2013
Date of preparation:	October 25, 2013
Date of revision:	October 25, 2013
Start date of project:	November 1, 2010
Duration:	3 years
Project coordinator name:	Joachim Giesen (FSU)
Project coordinator organisation:	Friedrich-Schiller-Universität Jena Jena, Germany

Efficient edge skeleton computation for polytopes defined by oracles

Ioannis Z. Emiris* Vissarion Fisikopoulos* Bernd Gärtner†

October 25, 2013

Abstract

In general dimension, there is no known total polynomial algorithm for either convex hull or vertex enumeration, i.e. whose complexity depends polynomially in the input and output sizes. It is thus important to identify polytope constructions 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. All complexity bounds refer to the (oracle) Turing machine model. We also offer a workspace efficient variant of our algorithm by employing reverse search. 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 integer programming, where we offer an alternative approach removing the exponential dependence on the dimension in the complexity.

Keywords: general dimension, polytope oracle, edge-skeleton, linear optimization, vertex enumeration, Minkowski sum, secondary polytope

1 Introduction

Convex polytopes in general dimension admit a number of alternative representations. The best known, explicit representations for a polytope P is either as the set of its vertices (V-representation) or as a bounded intersection of halfspaces (H-representation). Switching between the two representations constitutes the convex hull and vertex enumeration problems. Given an H-representation, 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.

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. We therefore want 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. 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 [AF92] solves, in total polynomial-time, vertex

*Department of Informatics and Telecommunications, National and Kapodistrian University of Athens, Greece. {emiris,vissarion}@di.uoa.gr

†Institut für Theoretische Informatik, Swiss Federal Institute of Techology (ETHZ), Zurich, Switzerland, {gaertner@inf.ethz.ch}

enumeration for simple polytopes and convex hull for simplicial polytopes. In this paper we establish another case where total polynomial-time algorithms exist.

An important explicit representation is the *edge-skeleton* (or 1-skeleton) of P, which is the graph of polytope vertices and edges, but none of the faces of dimension ≥ 2. For simple polytopes, the skeleton determines the complete face lattice (see [JKK02] 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: In [BDH09] the authors show how the edge-skeleton suffices for point location by allowing them to recover only the needed simplices.

In this paper we study the case where a polytope 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; see e.g. [EFKP12, Hug06], although not in total polynomial-time.

Proposition 1. [EFKP12] *Given OPT_P for $P \subseteq \mathbb{R}^n$ the V- and H-representation as well as a triangulation of T of P can be computed in*

$$O(n^5 m s^2) \text{ arithmetic operations } + O(m+f) \text{ calls to } OPT_P,$$

where m, f are the number of vertices and facets of P and s the number of n-dimensional simplices of T.

Another important implicit representation of P is obtained through a *separation* oracle (Section 2). Celebrated results of Khachiyan [Kha79] as well as Grötschel, Lovász and Schrijver [GLS93] show that separation and optimization oracles are polynomial-time equivalent (Proposition 3). Many important results in combinatorial optimization use the fact that separation implies optimization. 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 a special case of 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. Since the vertices are computed along with the skeleton, our problem subsumes vertex enumeration for polytopes for which we know the edge directions. This resembles the fundamental Minkowski reconstruction problem, e.g. [GH99], except that, instead of information on the facets, we know the 1-dimensional faces (and an oracle). The problem of the reconstruction of a simple polytope by its edge-skeleton graph is studied in [JKK02].

The most relevant previous work is an algorithm for vertex enumeration of P, given by an optimization oracle and a superset D of all edge directions, proposed in [OR07] (Proposition 4). 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 m vertices can be done by $O(m^2)$ calls to LP.

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 difference*. Let $A-B$ be polytope C such that A can be written as a sum $A = B+C$. That is, 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

[MC00], [GKZ94, Thm 11.1.3]. Also, matroid polytopes and generalized permutahedra can be written as signed Minkowski sums [ABD10].

Minkowski sums have been studied extensively. Given r V-polytopes in $\mathbb{R}^n$, Gritzmann et al. [GS93] deal with the various Minkowski sum problems that occur if they regard none, one, or both of r and n as constants. They give polynomial algorithms for fixed n regardless of the input representation. For varying n they show that no polynomial-time algorithm exists except from the case of fixed r in the binary model of computation. Fukuda [Fuk04] (extended in [FW05]) 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 in 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 today seems to be via optimization oracles [EFKP12]. *Resultants* are fundamental in computational algebraic geometry since they generalize determinants to nonlinear systems [Stu94, GKZ94]. 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 $d+1$ pointsets in $\mathbb{Z}^d$. If R lies in $\mathbb{R}^n$, the total number of input points is $n+2d+1$. If m is the number of vertices in R, typically $m \gg n \gg d$, so d is assumed fixed. A polynomial-time optimization oracle yields an output-sensitive algorithm for R [EFKP12] (Lemma 13).

This approach can also be used for computing the secondary and discriminant polytopes, defined in [GKZ94]; cf [DLRS10] on secondary polytopes. The secondary polytope of a pointset is a fundamental object since it offers a polytope realization of the graph of regular triangulations of the pointset. A total polynomial-time algorithm for the secondary polytope when d is fixed is given in [MII96]. A specific application of discriminant polytopes is discussed in [Ore99], 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 n: the number of vertices of R is $O(n^{2n^2})$ [Stu94], and the number of k-faces (for any k) of the secondary polytope is $O(n^{(n-1)^2})$, which is tight if n is fixed [BFS90].

More applications of our methods exist. One is in *convex combinatorial optimization*: given $\mathcal{F} \subset 2^N$ with $N = \{1, \ldots, m\}$, a vectorial weighting $w : N \to \mathbb{Q}^n$, and a convex functional $c : \mathbb{Q}^n \to \mathbb{Q}$, 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 [OR04] and references therein. The standard linear combinatorial optimization problem is the special case with $n = 1, w : N \to \mathbb{Q}$, and $c : \mathbb{Q} \to \mathbb{Q} : x \mapsto x$ being the identity. As shown in [OR04], a convex combinatorial optimization problem can be solved in polynomial-time for fixed n, 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 programming*, where we maximize a convex function over the integer hull of a polyhedron. In [LHO$^+$09], the vertex enumeration algorithm of [OR07]—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.

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 unspecified 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 [AF92] we offer a workspace 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 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 resultant polytopes, optimization oracles offer the most efficient representation. Our results on resultant polytopes extend to secondary polytopes, for which a different approach in the same complexity class is known, as well as discriminant polytopes.

Regarding the problems of convex combinatorial optimization and convex integer programming the current approaches use the algorithm of [OR07] whose complexity has an exponential dependence on the dimension (Proposition 4). The utilization of our algorithm instead offers an alternative approach while removing the exponential dependence on the dimension.

Notation. n denotes the ambient space dimension and m the number of vertices of the output (bounded) polytope; d denotes dimension when it is fixed (e.g. input space for resultant polytopes); $\text{conv}(A)$ 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 $P \subseteq \mathbb{R}^n$, we will define $\langle P \rangle := n + \varphi$. Let $\mathbb{O} : \mathbb{R} \to \mathbb{R}$ denote a polynomial such that the oracle conversion algorithms of Proposition 3 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 $\{x \mid Ax \leq b\}$.

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, 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 some basics from polytope theory; for a detailed presentation we refer to [Zie95].

A convex polytope $P \subseteq \mathbb{R}^n$ can be represented as: the convex hull of a finite set of points, called the *V-representation* of P, or the bounded intersection of a finite set of halfspaces or linear inequalities, called the *H-representation* of P. Given P, an inequality or a halfspace $\{a^T x \leq b\}$ is called *supporting* if it holds for all $x \in P$. The set $\{x \in P \mid a^T x = b\}$ is a *face* of P. Faces of dimension $0, 1, n-2, n-1$ are called *vertices*, *edges*, *ridges* and *facets* respectively.

Definition 1. *The polar dual polytope of P in dual space $(\mathbb{R}^n)^*$, is defined as:*

$$P^* := \{c \in \mathbb{R}^n : c^T x \leq 1, \text{ for all } x \in P\} \subseteq (\mathbb{R}^n)^*,$$

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 $< n$.

For our results, we need to assume that the output polytope is *well-described* [GLS93, Definition 6.2.2]. This will be the case in all our applications.

Definition 2. *A rational polytope $P \subseteq \mathbb{R}^n$ 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 $\langle P \rangle = n + \varphi$. Similarly, the* encoding length *of a set of vectors $D \subseteq \mathbb{R}^n$ is $\langle D \rangle = n + \nu$ 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 2. [GLS93, Lemma 6.2.4] *Let $P \subseteq \mathbb{R}^n$. 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 $4n^2\varphi$. If every vertex of P has encoding length at most ν, then every inequality of its H-representation has encoding length at most $3n^2\nu$.*

The natural model of computation when P is given by an oracle is that of an *oracle Turing machine* [GLS93, 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 ($\mathrm{OPT}_P(c)$): Given vector $c \in \mathbb{R}^n$, either find a point $y \in P$ maximizing $c^T x$ over all $x \in P$, or assert $P = \emptyset$.

Violation ($\mathrm{VIOL}_P(c, \gamma)$): Given vector $c \in \mathbb{R}^n$ and $\gamma \in \mathbb{R}$, either find point $y \in P$ such that $c^T y > \gamma$, or assert that $c^T x \leq \gamma$ for all $x \in P$.

Separation ($\mathrm{SEP}_P(y)$): Given point $y \in \mathbb{R}^n$, either certify that $y \in P$, or find a hyperplane that separates y from P; i.e. find vector $c \in \mathbb{R}^n$ such that $c^T y > c^T x$ for all $x \in P$.

The following is a main result of [GLS93] and the cornerstone of our method.

Proposition 3. [GLS93, Thm 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 n 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 n, 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 $Ax \leq b, x \in \mathbb{R}^n$ 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. We are given a well-described polytope $P \subseteq \mathbb{R}^n$ 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 4. [OR07] *Let $P \subseteq \mathbb{R}^n$ be a polytope given by $OPT_P(c)$, and let $D \supseteq D(P)$ be a superset of the edge directions of P. The vertices of P are computed using $O(|D|^{n-1})$ arithmetic operations and $O(|D|^{n-1})$ calls to $OPT_P(c)$.*

If P has m vertices, then $|D(P)| \leq \binom{m}{2}$, and this is tight for neighborly polytopes in general position. This means that the bound of Proposition 4 is $O\left(m^{2n-2}\right)$, 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 n in Proposition 4 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 [GLS93, Lemma 6.51], [EPL82, 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 maximizes t over the 1-dimensional polytope $Q(e) := P \cap \{x \mid x = v + te, t \geq 0\}$, the intersection of P with the ray in 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 the above linear programs, we need to construct optimization oracles for the $Q(e)$. To do this, we first construct $\mathrm{SEP}_P(y)$ from $\mathrm{OPT}_P(c)$ in oracle polynomial-time according to Proposition 3. From this, the construction of $\mathrm{SEP}_{Q(e)}(y)$ is elementary, and since also $Q(e)$ is well-described, we can in turn obtain $\mathrm{OPT}_{Q(e)}(c)$ in oracle polynomial-time.

In a final step, we remove the candidates that do not yield 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. $v = 0$ and $h = \{x \mid x_n = 1\}$. Let C be the cone generated by the set V_{cone}. We compute the extreme points of $C \cap \{x_n = 1\}$,

Algorithm 1: Edge_Skeleton (OPT_P, D)

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

Compute some vertex $v_0 \in P$;
$V_P \leftarrow \emptyset$; $W \leftarrow \{v_0\}$; $E_P \leftarrow \emptyset$;
while $W \neq \emptyset$ **do**
  Choose the next element $v \in W$ and remove it from W;
  $V_P \leftarrow V_P \bigcup \{v\}$;
  $V_{cone} \leftarrow \emptyset$;
  foreach $e \in E$ **do**
    $w = \mathrm{argmax}\{v + te \in P, t \geq 0\}$;
    if $w \neq v$ **then**
      $V_{cone} \leftarrow V_{cone} \bigcup \{v'\}$;
  Remove non-vertices of P from V_{cone};
  foreach $w \in V_{cone}$ **do**
    insert w into W;
    $E_P \leftarrow E_P \bigcup \{v, w\}$;
return V_P, E_P;

giving us the extremal rays of C. 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 5. *Let v be a vertex of P processed during Algorithm 1, where we assume w.l.o.g. that $v = 0$ and the set V_{cone} of candidates is separated from v by the hyperplane $\{x \mid x_n = 1\}$.*

A point $p \in \mathbb{R}^n$ is a neighbor of v if and only if p 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 $\{x_n = 1\}$ is an extreme point of the polytope $C \cap \{x \mid x_n = 1\}$.

Proof. Suppose w is a neighbor of v. By construction, $w \in V_{cone}$. Moreover, since $\{v, w\}$ is an edge, there is a hyperplane $h = \{a^T x = 0\}$ (recall that $v = 0$) such that $a^T x = 0$ for all $x \in \mathrm{conv}(\{v, w\})$ and $a^T x > 0$ for all $p \in P \setminus \mathrm{conv}(\{v, w\})$. For each $q \in V_{cone}$, let $c(q) = \frac{1}{q_n} q \in C \cap \{x \mid x_n = 1\}$. We have $q_n > 0$ by construction. Furthermore, $a^T c(w) = 0$ while $a^T c(q) > 0$ for $q \in V_{cone}$, unless $q \in \mathrm{conv}(\{v, w\})$. In the latter case, $c(q) = c(w)$. Hence, $c(w)$ is the only point y in $C \cap \{x_n = 1\}$ such that $a^T y = 0$, and this implies that $c(w)$ is an extreme point of $C \cap \{x_n = 1\}$. So w is on some extremal ray of C.

For the other direction, suppose that $p \in V_{cone}$ is on the extremal ray $\{tp \mid t \in \mathbb{R}\}$. So $c(p)$ is an extreme point of $C \cap \{x \mid x_n = 1\}$. This means, there exists a vertical hyperplane $h = \{a^T x = b\}$ with $a_n = 0$ such that $a^T c(p) = b$ and $a^T(c(q)) > b$ for all $q \in V_{cone}$ satisfying $c(q) \neq c(p)$. Now define the hyperplane $\overline{h} = \{\overline{a}^T x = 0\}$ with $\overline{a} = (a_1, \ldots, a_{n-1}, -b)$. It follows that $\overline{a}^T q \geq 0$ for all $q \in V_{cone}$, so the positive halfspace of $\overline{h}$ contains C and thus also P since $P \subseteq C$. We claim $\overline{h} \cap P = \mathrm{conv}(\{v, p\})$, which proves p is a neighbor of v.

To see this, we first observe that $\overline{a}^T p = 0$ and $\overline{a}^T q > 0$ for all $q \in V_{cone}$ that are not multiples of p, so $\overline{h} \cap P \subseteq \overline{h} \cap C = \{tp \mid t \in \mathbb{R}\}$. On the other hand, we know from the construction of V_{cone} that p is the highest point of P on the ray $\{tp \mid t \in \mathbb{R}\}$, so we indeed get $\overline{h} \cap P = \mathrm{conv}(\{v, p\})$. □

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

Theorem 6. *Given OPT_P and a superset of edge directions D of a well-described polytope $P \subseteq \mathbb{R}^n$, Algorithm 1 computes the edge-skeleton of P in oracle total polynomial-time*

$$O\left(m\left(|D|\mathbb{O}(\langle P\rangle + \langle D\rangle) + \mathbb{LP}(4n^3|D|(\langle P\rangle + \langle D\rangle))\right)\right),$$

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 $O(m)$ 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 m) 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 2, these coordinates have encoding lengths bounded by $4n^2(\langle P\rangle + \langle D\rangle)$. □

3.1 Reverse search for edge-skeleton.

We define a reverse search procedure based on [AF92] 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 $c \in \mathbb{R}^n$ 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 4 consume exponential space in the worst case. This yields the following result (encoding length of P vertices comes from Lemma 2).

Corollary 7. *Given OPT_P and a superset of edge directions D, a variant of Algorithm 1 that uses reverse search runs in space $O(4n^2\langle P\rangle + \langle D\rangle)$ while keeping the same asymptotic time complexity.*

4 Applications

We examine two important classes of polytopes and provide new, total polynomial-time algorithms for their representation by an edge-skeleton: signed Minkowski sums, and resultant and secondary polytopes. These polytopes are well-described and naturally defined by optimization oracles, which provide a compact though implicit representation. Moreover, it is possible to compute their edge directions efficiently, as illustrated below.

4.1 Signed Minkowski sums.

Recall that the *Minkowski sum* of polytopes $A, B \subseteq \mathbb{R}^n$ is defined as $A+B := \{a+b \mid a \in A, b \in B\}$. Following [Sch93] the *Minkowski difference* is defined as $A-B := \{x \in \mathbb{R}^n \mid B+x \subseteq A\}$. Here

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

$$P = s_1P_1 + s_2P_2 + \cdots + s_rP_r,\ s_i \in \{-1, 1\}.$$

An optimization oracle for the signed Minkowski sum is given by the following.

Lemma 8. *If $P_1, \ldots, P_r \subset \mathbb{R}^n$ are given by optimization oracles, then we construct an optimization oracle for signed Minkowski sum $\sum_{i=1}^r s_iP_i$ in $O(r)$.*

Proof. Assume $s_1 = \cdots = s_k = 1 \neq s_{k+1} = \cdots = s_r = -1$. First we consider the additions. Given $w \in \mathbb{R}^n$, use the oracles for $P_1, \ldots, P_k$ to compute vertices $u_1 \in f_1^w, \ldots, u_k \in f_k^w$, where the $f_i^w \subset P_i$ are extremal faces in the direction of w. Then return $u_0 = u_1 + \cdots + u_k$, which is a vertex of $P_0 = P_1 + \cdots + P_k$ in the face which is extremal in direction w. We now consider the Minkowski differences, and wish to compute $P_0 - P_{k+1} - \cdots - P_r \in \mathbb{R}^n$. We again use the oracles for $P_{k+1}, \ldots, P_r$ to compute vertices $u_{k+1} \in f_{k+1}^w, \ldots, u_r \in f_r^w$ and return $u_0 - u_{k+1} - \cdots - u_r$. By combining the two cases, we obtain an oracle that determines uniquely a vertex of $\sum_{i=1}^r s_iP_i$ maximizing inner product with w. This is OPT_P of complexity $O(r)$ since, by definition of oracle polynomial-time, the oracle calls in every summand are of unit cost. □

Let m denote the number of vertices of P. An algorithm for the explicit representation for P follows from Proposition 1 and Lemma 8:

Corollary 9. *Given optimization oracles for $P_1, \ldots, P_r \subseteq \mathbb{R}^n$, we construct the V- and H-representations, and a triangulation T of signed Minkowski sum $P = P_1 + s_2P_2 + \cdots + s_rP_r$, $s_i \in \{-1, 1\}$ in output sensitive complexity, namely $O(n^5ms^2 + (m+f)r)$, where m, f are the number of vertices and facets in P and s the number of full-dimensional simplices of T.*

However, the output representation of the above algorithm can be exponential in m, thus we focus on total polynomial-time algorithms for the edge-skeleton of P. We assume the input is a superset of all edges for each P_i. If, instead, we are given the vertices of all summands P_i, then 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 P_i edges is polynomial.

Corollary 10. *Given optimization oracles for well-described $P_1, \ldots, P_r \subseteq \mathbb{R}^n$, 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 2, the encoding length of the coordinates of summand vertices is $4n^2\langle P_{max} \rangle$. Thus, $4n^2\langle P_{max} \rangle + \langle r \rangle$ is the encoding length of the coordinates of P vertices. Finally, $\langle P \rangle = n + 12n^4\langle P_{max} \rangle + 3n^2\langle r \rangle$ by Lemma 2. Now OPT_P is computed by Lemma 8 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 $P_1 - P_2 = P_3 \iff P_1 = P_2 + P_3$. □

Note that our algorithm assume that in the Minkowski difference $A-B$ of polytopes $A, B \subseteq \mathbb{R}^n$, B is a summand of A and cannot check whether this assumption holds.

4.2 Secondary and Resultant polytopes.

The *secondary polytope* Σ of a set of n points $A \subset \mathbb{Z}^d$ is a fundamental object since it expresses the triangulations of $\mathrm{conv}(A)$ via a polytope representation. For any triangulation T of $\mathrm{conv}(A)$,

define vector $\phi_T \in \mathbb{R}^n$ with coordinate

$$\phi_T(a) = \sum_{\sigma \in T \mid a \in \text{vtx}(\sigma)} \text{vol}(\sigma), \qquad a \in A,$$

summing over all simplices σ of T having a as a vertex, where $\text{vtx}(\sigma)$ is the vertex set of simplex σ. Now the secondary polytope $\Sigma(\mathcal{A})$ is defined as the convex hull of ϕ_T for all triangulations T. A famous theorem of [GKZ94], which is also the central result in [DLRS10], states that there is a bijection between the vertices of Σ and the regular triangulations of $\text{conv}(A)$. This extends to a bijection between the face poset of Σ and the poset of regular subdivisions of $\text{conv}(A)$. Moreover, Σ, although in ambient space $\mathbb{R}^n$, has actual dimension $\dim(\Sigma) = n - d - 1$.

Let us now consider resultant polytopes, for which optimization oracles provide the only plausible approach [EFKP12]. Let us consider sets $A_0, \ldots, A_d \subset \mathbb{Z}^d$. In the algebraic setting, these are the supports of $d+1$ polynomials in d variables. Let the Cayley set be defined by

$$A := \bigcup_{i=0}^{d} (A_i \times \{e_i\}) \subset \mathbb{Z}^{2d},$$

where $e_0, \ldots, e_d$ form an affine basis of $\mathbb{Z}^d$. Let $n := \sum_{i=0}^{d} |A_i|$, then given triangulation T of $\text{conv}(A)$, define vector $\rho_T \in \mathbb{R}^n$ with coordinate

$$\rho_T(a) := \sum_{a\text{-mixed } \sigma \in T \mid a \in \text{vtx}(\sigma)} \text{vol}(\sigma), \qquad a \in A,$$

where simplex σ is *a-mixed* if it contains 2 vertices from A_j, and the vertex from A that corresponds to $a \in A_i$, where $j \in \{0, 1, \ldots, d\} - \{i\}$. 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 $\mathbb{R}^n$ but has dimension $\dim(R) = n - 2d - 1$ [GKZ94]. There is a surjection, i.e. many to one relation, from the regular triangulations of $\text{conv}(A)$ to the vertices of R.

We consider d fixed because in practice it holds $d \ll n \ll m$. Note that R is computed as a full-dimensional polytope in a space of their intrinsic dimension [EFKP12] and this approach extends to Σ.

Computing the V-representation of Σ, R in [EFKP12] is not total polynomial. In fact, the complexity for R depends on the number of its vertices and facets, but also the number of simplices in a triangulation of R. However here, we show that Algorithm 1 computes Σ, R in oracle total polynomial-time. Our results are readily extended to discriminant polytopes.

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

Proof. For the case of Σ, given $A \in \mathbb{Z}^d$, let $\langle A \rangle$ be its encoding length and $\alpha := \text{vol}(\text{conv}(A))$. It is clear that $\alpha = O(\langle A \rangle^d)$ and thus $\langle \alpha \rangle = O(d\langle A \rangle)$. 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 $\text{vol}(\text{conv}(A))$. This bound is tight for the points $a \in A$ of a regular triangulation T where the simplices containing a partition $\text{conv}(A)$. It follows that the encoding length of Σ vertices is $\langle \alpha \rangle$ and thus $\langle \Sigma \rangle = 4n^2\langle \alpha \rangle + n = O(dn^2\langle A \rangle)$ by Lemma 2. Similarly, we bound the encoding length of ρ_T which yields that R is also a well-described polytope. □

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.

Circuits are minimum affinely dependent subsets of A with exactly two triangulations C_+, C_-. Triangulation T of A, which equals C_+ when restricted on circuit C, is supported

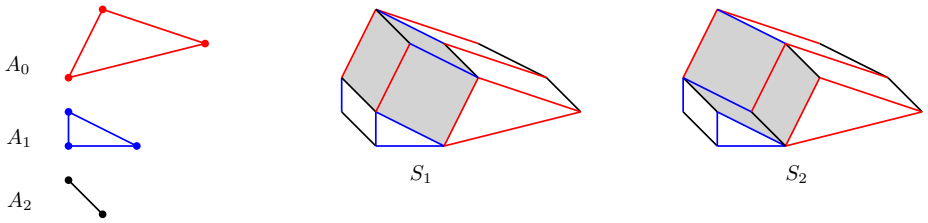

Figure 1: Two mixed subdivisions S_1, S_2 of $A_0 + A_1 + A_2$.

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 $A = \bigcup_{i=0}^{d} A_i$, a circuit C is called *cubical* if and only if $|C \cap A_i| \in \{0, 2\}$, $i = 0, \ldots, d$ (Figure 1). If A is in *generic position*, all the edges of R correspond to full dimensional cubical circuits [Stu94].

Example 1. *Let* $A_0 = \{\{0,0\},\{1,2\},\{4,1\}\}$, $A_1 = \{\{0,0\},\{0,1\},\{2,0\}\}$, $A_2 = \{\{1,0\},\{0,1\}\}$. *Two regular mixed subdivisions* S_1, S_2 *of their Minkowski sum* $A_0 + A_1 + A_2$ *are depicted in Figure 1. The gray cells correspond to the circuit that supports a flip, which is also cubical. Thus, this flip corresponds to an edge of resultant polytope* R.

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

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

In the case of R, where $A = \bigcup_{i=0}^{d} A_i$, a flip on T is cubical iff it is supported on a cubical circuit C.In generic position, $|C| = 2d + 2$. For those supporting cubical flips, $|C \cap A_i| = 2$, $i = 0, \ldots, d$. 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 [Stu94]. Given A, all such circuits are enumerated in $\binom{|A|}{2d+2} = O(n^{2d+2})$; a better bound is $O(t^{2d+2})$ if t bounds $|A_i|$, $i = 0, \ldots, d$. □

Lemma 13. [EFKP12] *For* $d+1$ *pointsets in* $\mathbb{Z}^d$ *of total cardinality* n, *optimization over* R *takes polynomial-time, when* d *is fixed.*

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

As described in Lemma 12 for Σ, R polytopes we also compute the lengths of the polytope edges in addition to their directions. This can be utilized to yield a more efficient edge-skeleton algorithm in the real RAM model of computation.

5 Future work

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 [ORT05]. It is also interesting to investigate new classes of convex combinatorial optimization problems where our algorithm offers a polynomial-time algorithm. Lastly, we wish to extend our approach to approximating the volume of polytopes given by an optimization oracle, as opposed to the classic assumption of possessing a membership (or separation) oracle. In [EFG13] some preliminary results are available based on random walks, such as hit-and-run, for generating an adequate sample of points in the polytope.

Acknowledgment. Work 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. We also thank K. Fukuda, C. Müller, S. Stich for discussions and bibliographic suggestions.

References

[ABD10] Federico Ardila, Carolina Benedetti, and Jeffrey Doker. Matroid polytopes and their volumes. *Discrete & Computational Geometry*, 43(4):841–854, 2010.

[AF92] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete & Comput. Geometry*, 8:295–313, 1992.

[BDH09] J.-D. Boissonnat, O. Devillers, and S. Hornus. Incremental construction of the Delaunay triangulation and the Delaunay graph in medium dimension. In *ACM Symp. on Comp. Geometry*, pages 208–216, 2009.

[BFS90] L.J. Billera, P. Filliman, and B. Sturmfels. Constructions and complexity of secondary polytopes. *Advances in Math.*, 83(2):155–179, 1990.

[DLRS10] J.A. De Loera, J. Rambau, and F. Santos. *Triangulations: Structures for Algorithms and Applications*, volume 25 of *Algorithms and Computation in Mathematics*. Springer, 2010.

[EFG13] I.Z. Emiris, V. Fisikopoulos, and B. Gärtner. Efficient volume and edge-skeleton computation for polytopes defined by oracles. In *Proc. EuroCG 2013*, Braunschweig, Germany, March 2013.

[EFKP12] I.Z. Emiris, V. Fisikopoulos, C. Konaxis, and L. Peñaranda. An output-sensitive algorithm for computing projections of resultant polytopes. In *Proc. ACM Symp. on Comp. Geom.*, pages 179–188, 2012.

[EPL82] J. Edmonds, W.R. Pulleyblank, and L. Lovász. Brick decompositions and the matching rank of graphs. *Combinatorica*, 2(3):247–274, 1982.

[Fuk04] K. Fukuda. From the zonotope construction to the Minkowski addition of convex polytopes. *J. Symbolic Computation*, 38, 2004.

[FW05] K. Fukuda and C. Weibel. Computing all faces of the Minkowski sum of V-polytopes. In *Canad. Conf. Comp. Geom.*, pages 253–256, 2005.

[GH99] P. Gritzmann and A. Hufnagel. On the algorithmic complexity of Minkowski's reconstruction problem. *J. London Math. Soc.*, 2:5–9, 1999.

[GKZ94] I.M. Gelfand, M.M. Kapranov, and A.V. Zelevinsky. *Discriminants, Resultants and Multidimensional Determinants*. Birkhäuser, Boston, 1994.

[GLS93] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 2nd corrected edition, 1993.

[GS93] P. Gritzmann and B. Sturmfels. Minkowski addition of polytopes: computational complexity and applications to Gröbner bases. *SIAM J. Discr. Math.*, 6(2):246–269, 1993.

[Hug06] P. Huggins. ib4e: A software framework for parametrizing specialized LP problems. In A. Iglesias and N. Takayama, editors, *Mathematical Software - ICMS*, volume 4151 of *LNCS*, pages 245–247. Springer, 2006.

[JKK02] M. Joswig, V. Kaibel, and F. Körner. On the k-systems of a simple polytope. *Israel J. Math.*, 129(1):109–117, 2002.

[Kha79] L.G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Math. Doklady*, 20(1):191–194, 1979.

[LHO+09] J.A. De Loera, R. Hemmecke, S. Onn, U.G. Rothblum, and R. Weismantel. Convex integer maximization via Graver bases. *J. Pure & Applied Algebra*, 213(8):1569–1577, 2009.

[MC00] T. Michiels and R. Cools. Decomposing the secondary cayley polytope. *Discr. Comput. Geometry*, 23:367–380, 2000.

[MII96] T. Masada, H. Imai, and K. Imai. Enumeration of regular triangulations. In *ACM Symp. on Comp. Geometry*, SoCG '96, pages 224–233, 1996.

[OR04] S. Onn and U.G. Rothblum. Convex combinatorial optimization. *Discrete & Computat. Geometry*, 32(4):549–566, 2004.

[OR07] S. Onn and U.G. Rothblum. The use of edge-directions and linear programming to enumerate vertices. *J. Combin. Optim.*, 14:153–164, 2007.

[Ore99] S.Yu. Orevkov. The volume of the Newton polytope of a discriminant. *Russ. Math. Surv.*, 54(5):1033–1034, 1999.

[ORT05] S. Onn, U.G. Rothblum, and Y. Tangir. Edge-directions of standard polyhedra with applications to network flows. *J. of Global Optimization*, 33(1):109–122, September 2005.

[Sch93] Rolf Schneider. *Convex bodies: the Brunn-Minkowski theory.* Cambridge: Cambridge University Press, 1993.

[Stu94] B. Sturmfels. On the Newton polytope of the resultant. *J. Algebraic Combin.*, 3:207–236, 1994.

[Zie95] G.M. Ziegler. *Lectures on Polytopes*. Springer, 1995.