Constructing Polytopes via a Vertex Oracle

Vissarion Fisikopoulos

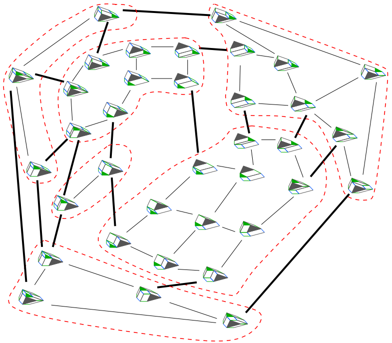
Joint work with I.Z. Emiris, C. Konaxis (now U. Crete) and
L. Peñaranda (now IMPA, Rio)

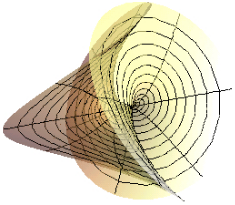
Department of Informatics, University of Athens

Mittagsseminar, ETH, Zurich, 12.Jul.2012

Main actor: resultant polytope

- Geometry: Minkowski summands of secondary polytopes, equival. classes of secondary vertices, generalize Birkhoff polytopes
- Motivation: useful to express the solvability of polynomial systems
- Applications: discriminant and resultant computation, implicitization of parametric hypersurfaces





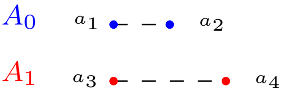
Enneper's Minimal Surface

Existing work

- Theory of resultants, secondary polytopes, Cayley trick [GKZ '94]
- TOPCOM [Rambau '02] computes all vertices of secondary polytope.
- [Michiels & Verschelde DCG'99] coarse equivalence classes of secondary polytope vertices.
- [Michiels & Cools DCG'00] decomposition of $\Sigma(\mathcal{A})$ in Minkoski summands, including $N(\mathcal{R})$.
- Tropical geometry [Sturmfels-Yu '08]: algorithms for resultant polytope (`GFan` library) [Jensen-Yu '11] and discriminant polytope (`TropLi` software) [Rincn '12].

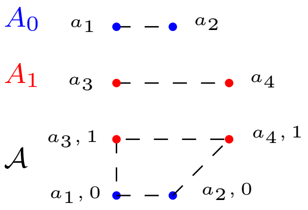
What is a resultant polytope?

- Given $n+1$ point sets $A_0, A_1, \dots, A_n \subset \mathbb{Z}^n$



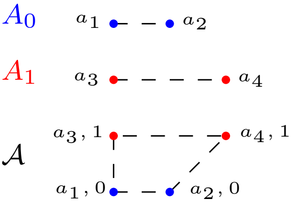
What is a resultant polytope?

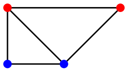
- Given $n+1$ point sets $A_0, A_1, \dots, A_n \subset \mathbb{Z}^n$
- $\mathcal{A} = \bigcup_{i=0}^{n} (A_i \times \{e_i\}) \subset \mathbb{Z}^{2n}$ where $e_i = (0, \dots, 1, \dots, 0) \subset \mathbb{Z}^n$



What is a resultant polytope?

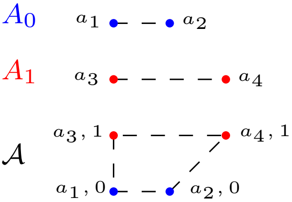
- Given $n+1$ point sets $A_0, A_1, \ldots, A_n \subset \mathbb{Z}^n$
- $\mathcal{A} = \bigcup_{i=0}^{n} (A_i \times \{e_i\}) \subset \mathbb{Z}^{2n}$ where $e_i = (0, \ldots, 1, \ldots, 0) \subset \mathbb{Z}^n$
- Given T a triangulation of $conv(\mathcal{A})$, a cell is a-mixed if it contains 2 vertices from A_j, $j \neq i$, and one vertex $a \in A_i$.

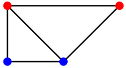




What is a resultant polytope?

- Given $n+1$ point sets $A_0, A_1, \dots, A_n \subset \mathbb{Z}^n$
- $\mathcal{A} = \bigcup_{i=0}^{n} (A_i \times \{e_i\}) \subset \mathbb{Z}^{2n}$ where $e_i = (0, \dots, 1, \dots, 0) \subset \mathbb{Z}^n$
- Given T a triangulation of $conv(\mathcal{A})$, a cell is a-mixed if it contains 2 vertices from A_j, $j \neq i$, and one vertex $a \in A_i$.
- $\rho_T(a) = \sum_{\substack{a-\text{mixed} \\ \sigma \in T : a \in \sigma}} \text{vol}(\sigma) \ \in \mathbb{N}, \quad a \in \mathcal{A}$

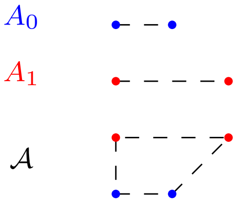


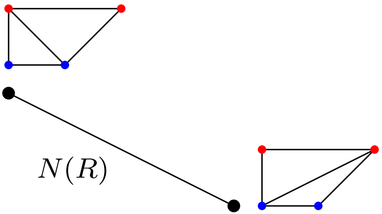


$$\rho_T = (0, 2, 1, 0)$$

What is a resultant polytope?

- Given $n+1$ point sets $A_0, A_1, \ldots, A_n \subset \mathbb{Z}^n$
- $\mathcal{A} = \bigcup_{i=0}^{n} (A_i \times \{e_i\}) \subset \mathbb{Z}^{2n}$ where $e_i = (0, \ldots, 1, \ldots, 0) \subset \mathbb{Z}^n$
- Given T a triangulation of $conv(\mathcal{A})$, a cell is a-mixed if it contains 2 vertices from A_j, $j \neq i$, and one vertex $a \in A_i$.
- $\rho_T(a) = \sum_{\substack{a-\text{mixed} \\ \sigma \in T : a \in \sigma}} \text{vol}(\sigma) \in \mathbb{N}, \quad a \in \mathcal{A}$
- Resultant polytope $N(R) = conv(\rho_T : T \text{ triang. of } conv(\mathcal{A}))$



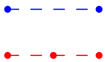


Connection with Algebra

- The support of a polynomial is the the set of exponents of its monomials with non-zero coefficient.
- The resultant R is the polynomial in the coefficients of a system of polynomials which is zero iff the system has a common solution.
- The resultant polytope $N(R)$, is the convex hull of the support of R.

A_0

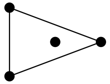
A_1



$$f_0(x) = ax^2 + b$$

$$f_1(x) = cx^2 + dx + e$$

$N(R)$



$$R(a,b,c,d,e) = ad^2b + c^2b^2 - 2caeb + a^2e^2$$

Connection with Algebra

- The support of a polynomial is the the set of exponents of its monomials with non-zero coefficient.
- The resultant R is the polynomial in the coefficients of a system of polynomials which is zero iff the system has a common solution.
- The resultant polytope $N(R)$, is the convex hull of the support of R.

A_0

$$f_0(x,y) = ax + by + c$$

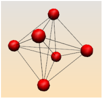
A_1

$$f_1(x,y) = dx + ey + f$$

A_2

$$f_2(x,y) = gx + hy + i$$

$N(R)$

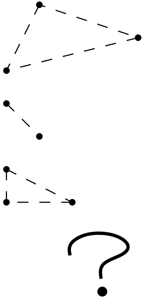


4-dimensional Birkhoff polytope

$$R(a,b,c,d,e,f,g,h,i) = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}$$

Connection with Algebra

- The support of a polynomial is the the set of exponents of its monomials with non-zero coefficient.
- The resultant R is the polynomial in the coefficients of a system of polynomials which is zero iff the system has a common solution.
- The resultant polytope $N(R)$, is the convex hull of the support of R.



A_0 $\qquad$ $$f_0(x, y) = axy^2 + x^4y + c$$

A_1 $\qquad$ $$f_1(x, y) = dx + ey$$

A_2 $\qquad$ $$f_2(x, y) = gx^2 + hy + i$$

$N(R)$

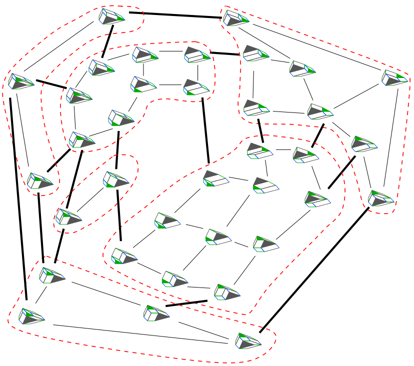
NP-hard to compute the resultant
in the general case

The idea of the algorithm

Input: $\mathcal{A} \in \mathbb{Z}^{2n}$ defined by $A_0, A_1, \ldots, A_n \subset \mathbb{Z}^n$
Simplistic method:

- compute the secondary polytope $\Sigma(\mathcal{A})$
- many-to-one relation between vertices of $\Sigma(\mathcal{A})$ and $N(R)$ vertices

Cannot enumerate 1 representative per class by walking on secondary edges

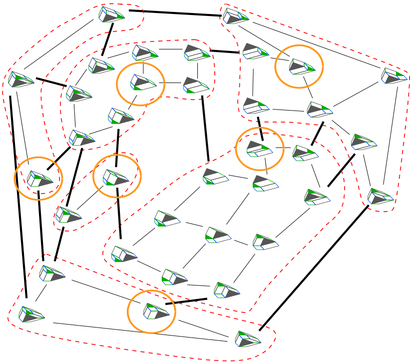


The idea of the algorithm

Input: $\mathcal{A} \in \mathbb{Z}^{2n}$ defined by $A_0, A_1, \ldots, A_n \subset \mathbb{Z}^n$

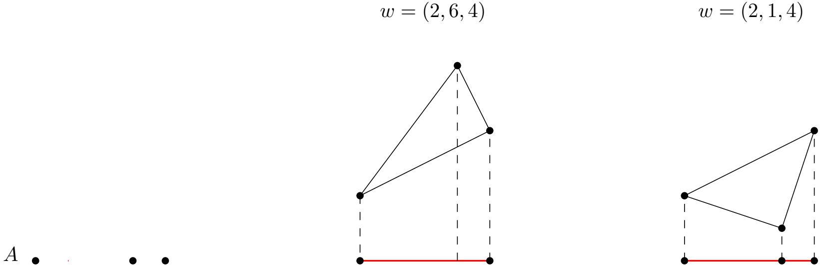
New Algorithm:

- Vertex oracle: given a direction vector compute a vertex of $N(R)$
- Output sensitive: computes only one triangulation of $\mathcal{A}$ per $N(R)$ vertex + one per $N(R)$ facet
- Computes projections of $N(R)$ or $\Sigma(\mathcal{A})$



A basic tool for the oracle:

Regular triangulations of $A \subset \mathbb{R}^d$ are obtained by projecting the lower (or upper) hull of A lifted to $\mathbb{R}^{d+1}$ via a generic lifting function $w \in (\mathbb{R}^{|\mathcal{A}|})^{\times}$.

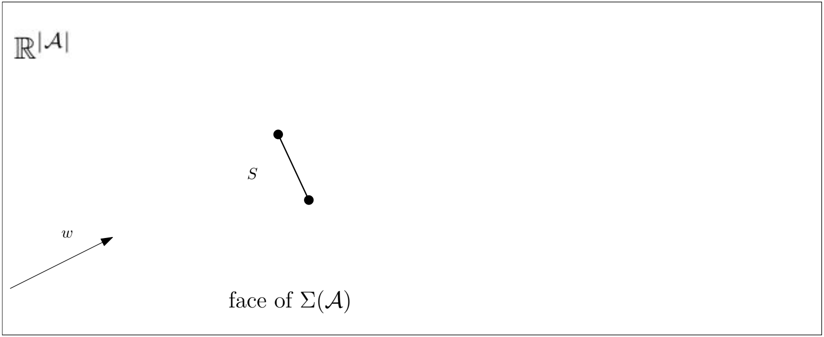


If w is not generic then we construct a regular subdivision.

The Vertex (Optimization) Oracle

Input: $\mathcal{A} \subset \mathbb{Z}^{2n}$, direction $w \in (\mathbb{R}^{|\mathcal{A}|})^\times$
Output: vertex $\in N(R)$, extremal wrt w

1. use w as a lifting to construct regular subdivision S of $\mathcal{A}$

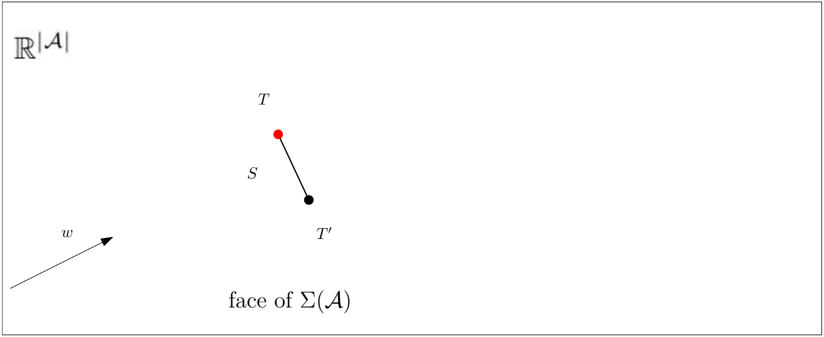


The Vertex (Optimization) Oracle

Input: $\mathcal{A} \subset \mathbb{Z}^{2n}$, direction $w \in (\mathbb{R}^{|\mathcal{A}|})^{\times}$

Output: vertex $\in N(R)$, extremal wrt w

1. use w as a lifting to construct regular subdivision S of $\mathcal{A}$
2. refine S into triangulation T of $\mathcal{A}$

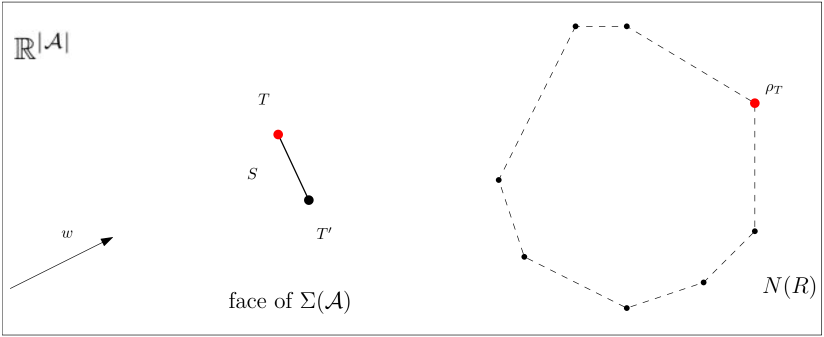


The Vertex (Optimization) Oracle

Input: $\mathcal{A} \subset \mathbb{Z}^{2n}$, direction $w \in (\mathbb{R}^{|\mathcal{A}|})^\times$

Output: vertex $\in N(R)$, extremal wrt w

1. use w as a lifting to construct regular subdivision S of $\mathcal{A}$
2. refine S into triangulation T of $\mathcal{A}$
3. return $\rho_T \in \mathbb{N}^{|\mathcal{A}|}$



The Vertex (Optimization) Oracle

Input: $\mathcal{A} \subset \mathbb{Z}^{2n}$, direction $w \in (\mathbb{R}^{|\mathcal{A}|})^{\times}$

Output: vertex $\in N(R)$, extremal wrt w

1. use w as a lifting to construct regular subdivision S of $\mathcal{A}$
2. refine S into triangulation T of $\mathcal{A}$
3. return $\rho_T \in \mathbb{N}^{|\mathcal{A}|}$

Lemma

Oracle's output is

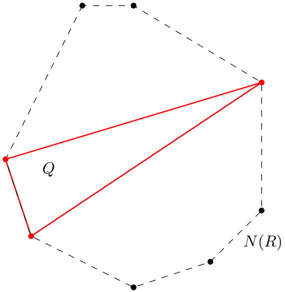
- *always a vertex of the target polytope,*
- *extremal wrt w.*

Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step



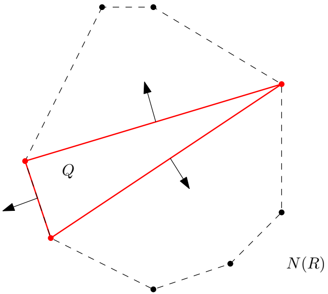
initialization:

- $Q \subset N(R)$
- dim(Q)=dim($N(R)$)

Incremental Algorithm

Input: $\mathcal{A}$
Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal



2 kinds of hyperplanes of Q_H :

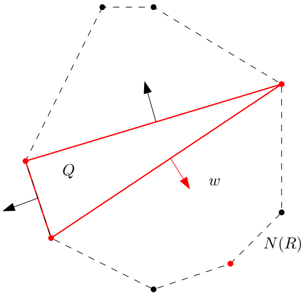
- legal if it supports facet $\subset N(R)$
- illegal otherwise

Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$

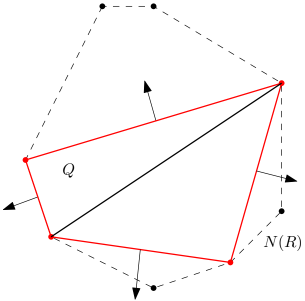


Extending an illegal facet

Incremental Algorithm

Input: $\mathcal{A}$
Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal



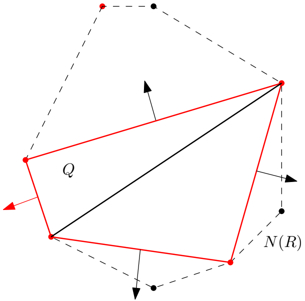
Extending an illegal facet

Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal



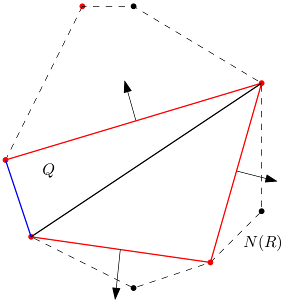
Validating a legal facet

Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal



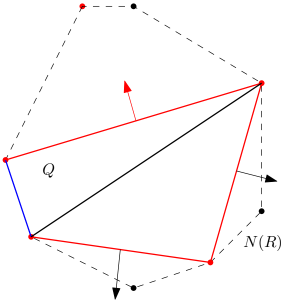
Validating a legal facet

Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

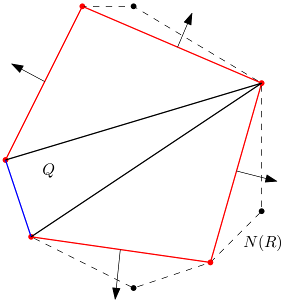
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal



Incremental Algorithm

Input: $\mathcal{A}$
Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal



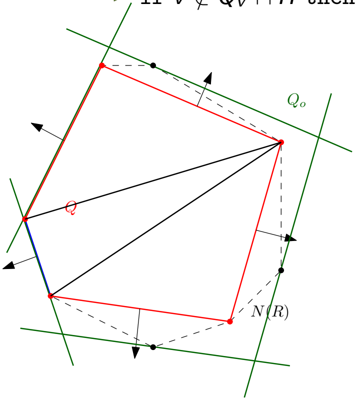
At any step, Q is an inner approximation ...

Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal



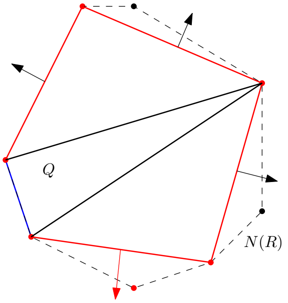
At any step, Q is an inner approximation ... from which we can compute an outer approximation Q_o.

Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

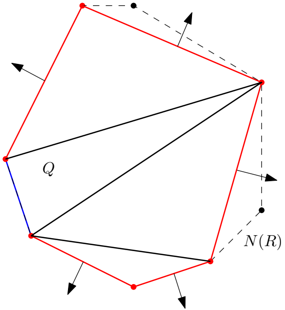
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
    - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
    - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal



Incremental Algorithm

Input: $\mathcal{A}$
Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

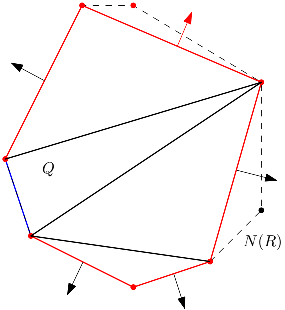
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal



Incremental Algorithm

Input: $\mathcal{A}$
Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal

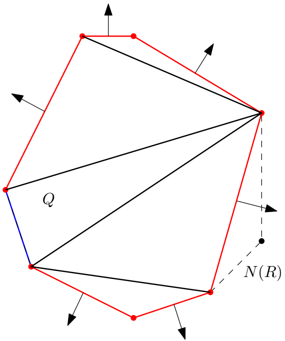


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal

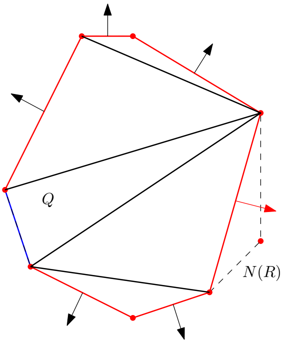


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal

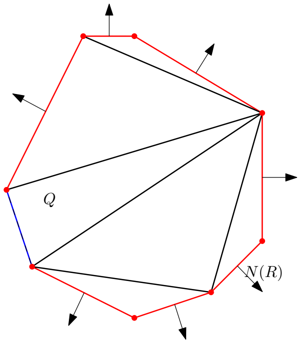


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal

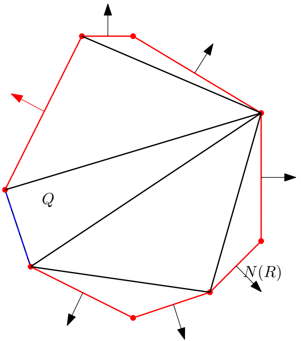


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal

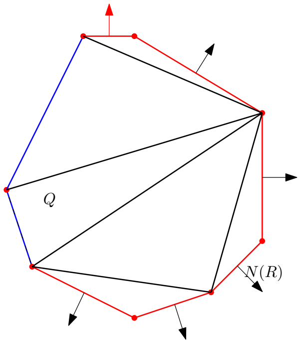


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal

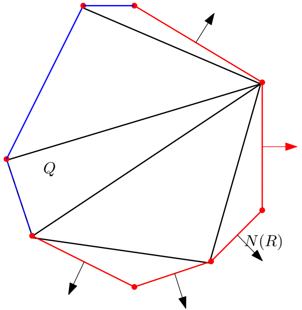


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
    - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
    - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal

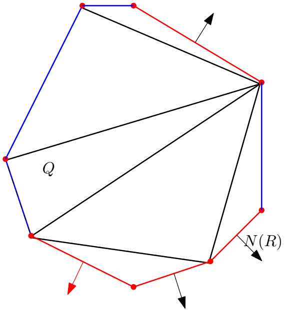


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

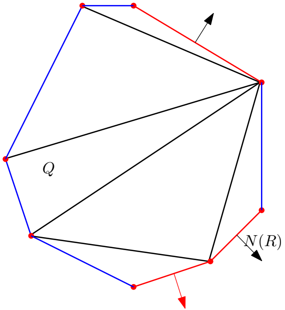
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal



Incremental Algorithm

Input: $\mathcal{A}$
Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
    - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
    - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal

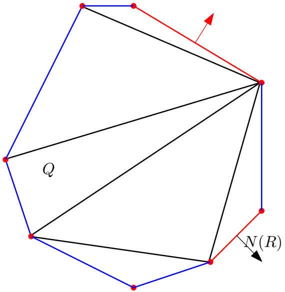


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

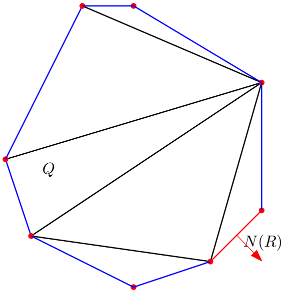
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal



Incremental Algorithm

Input: $\mathcal{A}$
Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

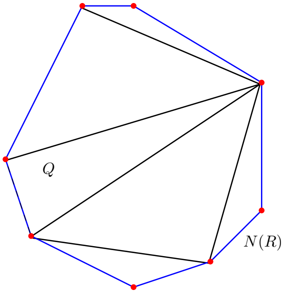
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
    - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
    - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal



Incremental Algorithm

Input: $\mathcal{A}$
Output: H-rep. Q_H, V-rep. Q_V of $Q = N(R)$

1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal



Complexity

Theorem

We compute the Vertex- and Halfspace-representations of $N(R)$, as well as a triangulation T of $N(R)$, in

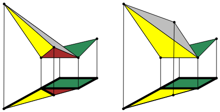
$$O^*(\, m^5 \, |vtx(N(R))| \cdot |T|^2),$$

where $m = \dim N(R)$, and $|T|$ the number of full-dim faces of T.

Elements of proof

- Computation is done in dimension $m = |\mathcal{A}| - 2n + 1$, $N(R) \subset \mathbb{R}^{|\mathcal{A}|}$.
- At most $\leq \mathrm{vtx}(N(R)) + \mathrm{fct}(N(R))$ oracle calls (Lem. 9).
- Beneath-and-Beyond algorithm for converting V-rep. to H-rep [Joswig '02].

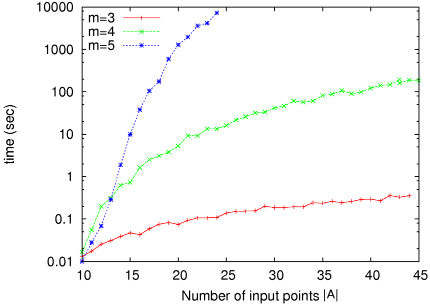
ResPol package

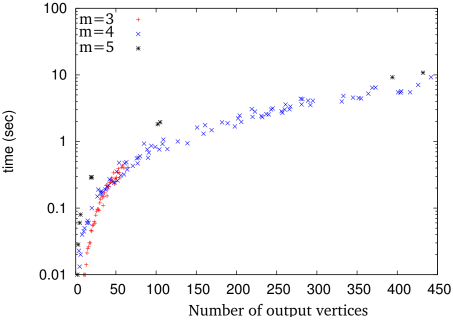


- C++
- towards high-dimensional CGAL
- `triangulation` [Boissonnat,Devillers,Hornus]
  `extreme_points_d` [Gärtner] (preprocessing step)
- Hashing of determinantal predicates: optimizing sequences of similar determinants
- `http://sourceforge.net/projects/respol`

Output-sensitivity

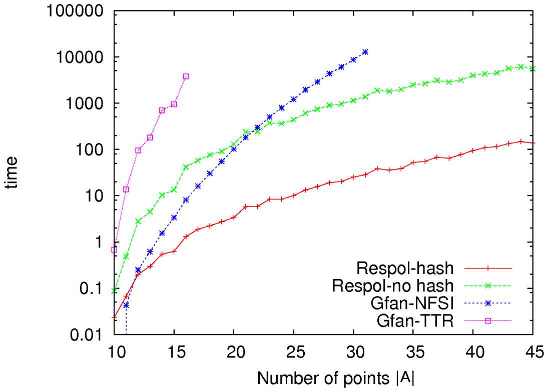
- oracle calls $\leq \mathrm{vtx}(N(R)) + \mathrm{fct}(N(R))$
- output vertices bound polynomially the output triangulation size
- subexponential runtime wrt to input points (L), output vertices (R)





Hashing and Gfan

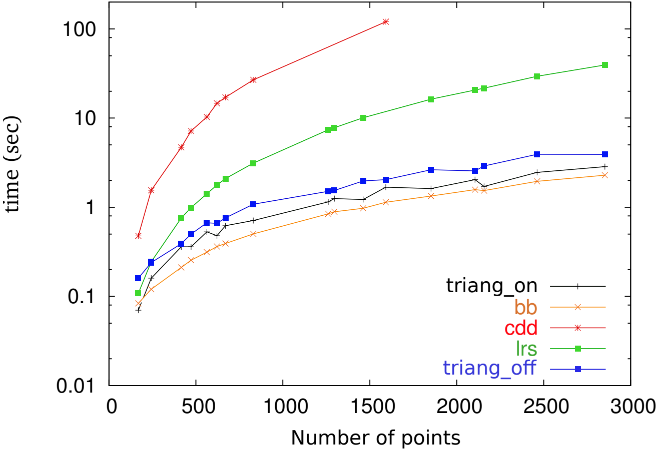
- *hashing determinants* speeds $\leq$ 10-100x when $dim(N(R)) = 3, 4$
- faster than `Gfan` [Yu-Jensen'11] for $dimN(R) \leq 6$, else competitive



$dim(N(R)) = 4$:

Computing the convex hull of N(R)

- `triangulation`, polymake beneath-beyond (`bb`), `cdd`, `lrs`



$dim(N(R)) = 4$

f-vectors of 4-dimensional N(R)

(6, 15, 18, 9)
(8, 20, 21, 9)
(9, 22, 21, 8)
.
.
.
(17, 48, 45, 14)
(17, 48, 46, 15)
(17, 48, 47, 16)
(17, 49, 47, 15)
(17, 49, 48, 16)
(17, 49, 49, 17)
(17, 50, 50, 17)
(18, 51, 48, 15)
(18, 51, 49, 16)
(18, 52, 50, 16)
(18, 52, 51, 17)
(18, 53, 51, 16)
(18, 53, 53, 18)
(18, 54, 54, 18)
(19, 54, 52, 17)
(19, 55, 51, 15)
(19, 55, 52, 16)
(19, 55, 54, 18)
(19, 56, 54, 17)
(19, 56, 56, 19)
(19, 57, 57, 19)
(20, 58, 54, 16)
(20, 59, 57, 18)
(20, 60, 60, 20)
(21, 62, 60, 19)
(21, 63, 63, 21)
(22, 66, 66, 22)

Open

Almost symmetric f-vector?

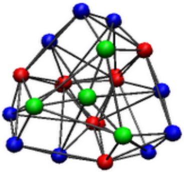
Ongoing and future work

- Extension of hashing determinants to CH computations (with L.Peñaranda) (to appear in ESA'12)
- Combinatorial characterization of 4-dimensional resultant polytopes (with I.Z.Emiris, A.Dickenstein)
- Computation of discriminant polytopes (with I.Z.Emiris, A.Dickenstein)
- Membership oracles from vertex (optimization) oracles (with B.Gärtner)

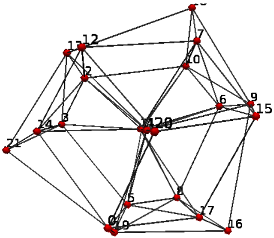
References

- The paper: "An output-sensitive algorithm for computing projections of resultant polytopes." in SoCG'12
- The code: `http://respol.sourceforge.net`

The end...



(figure courtesy of M.Joswig)



Facet and vertex graph of the largest 4-dimensional resultant polytope

Thank You !