Exact and approximate algorithms for resultant polytopes

Vissarion Fisikopoulos

Joint work with I.Z. Emiris and C. Konaxis*

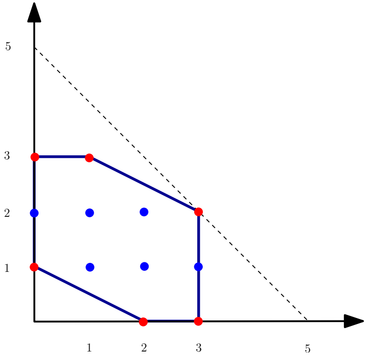
Dept Informatics & Telecoms, University of Athens
* currently with Univeristy of Crete

EuroCG 2012, Assisi, Perugia, Italy, 20.March.2012

Polynomials and Newton polytopes

- The support of a polynomial f is the set of exponents of its monomials with non-zero coefficient.
- The Newton polytope of f is the convex hull of its support.

$f(x, y) =$
$8y + xy - 24y^2 - 16x^2 + 220x^2y - 34xy^2 - 84x^3y + 6x^2y^2 - 8xy^3 + 8x^3y^2 + 8x^3 + 18y^3$



Polynomial systems

We study polynomials that expresses the solvability of polynomial systems.

Given a system of $n+1$ linear polynomials $f_0, f_1, \ldots, f_n$, on n variables the determinant is a polynomial on the coefficients which is zero iff the system has a common solution.

Polynomial systems

Given a system of $n+1$ linear polynomials $f_0, f_1, \ldots, f_n$, on n variables the determinant is a polynomial on the coefficients which is zero iff the system has a common solution.

$$f_0 = ax + by + c = 0$$
$$f_1 = dx + ey + f = 0$$
$$f_2 = gx + hy + i = 0$$

$$\begin{vmatrix} \mathrm{a} & \mathrm{b} & \mathrm{c} \\ \mathrm{d} & \mathrm{e} & \mathrm{f} \\ \mathrm{g} & \mathrm{h} & \mathrm{i} \end{vmatrix}$$

Polynomial systems

Given a system of $n+1$ linear polynomials $f_0, f_1, \ldots, f_n$, on n variables the determinant is a polynomial on the coefficients which is zero iff the system has a common solution.

$$f_0 = 4x + y + 2 = 0$$
$$f_1 = x + 2y + 1 = 0$$
$$f_2 = x + y + 8 = 0$$

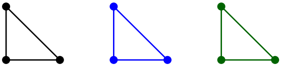
$$\begin{vmatrix} 4 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 8 \end{vmatrix} = 51$$

Polynomial systems

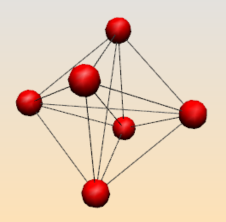
Given a system of $n+1$ linear polynomials $f_0, f_1, \ldots, f_n$, on n variables the determinant is a polynomial on the coefficients which is zero iff the system has a common solution.

$$f_0 = ax + by + c = 0$$
$$f_1 = dx + ey + f = 0$$
$$f_2 = gx + hy + i = 0$$

supports



Newton polytope of determinant (Birkhoff polytope)



$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}$$

Polynomial systems

Given a system of $n+1$ ~~linear~~ general polynomials $f_0, f_1, \ldots, f_n$, on n variables the ~~determinant~~ resultant is a polynomial on the coefficients which is zero iff the system has a common solution.

$f_0 = 4xy^2 + x^4y + 2 = 0$
$f_1 = x + 2y = 0$
$f_2 = 3x^2 + y + 8 = 0$

Polynomial systems

Given a system of $n+1$ ~~linear~~ general polynomials $f_0, f_1, \ldots, f_n$, on n variables the ~~determinant~~ resultant is a polynomial on the coefficients which is zero iff the system has a common solution.

$f_0 = 4xy^2 + x^4y + 2 = 0$
$f_1 = x + 2y = 0$
$f_2 = 3x^2 + y + 8 = 0$

hard to compute
the resultant

Polynomial systems

Given a system of $n+1$ ~~linear~~ general polynomials $f_0, f_1, \ldots, f_n$, on n variables the ~~determinant~~ resultant is a polynomial on the coefficients which is zero iff the system has a common solution.

$f_0 = 4xy^2 + x^4y + 2 = 0$
$f_1 = x + 2y = 0$
$f_2 = 3x^2 + y + 8 = 0$

supports

Newton polytope of ~~determinant~~ resultant
(~~Birkhoff~~ resultant polytope Π)

hard to compute
the resultant

?

Polynomial systems

Given a system of $n+1$ ~~linear~~ general polynomials $f_0, f_1, \dots, f_n$, on n variables the ~~determinant~~ resultant is a polynomial on the coefficients which is zero iff the system has a common solution.

supports

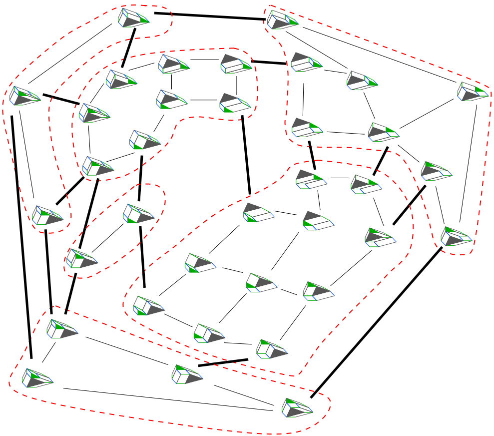
Newton polytope of ~~determinant~~ resultant
(~~Birkhoff~~ resultant polytope Π)

?

The idea of the algorithm

The input supports define a pointset $\mathcal{A} \in \mathbb{Z}^{2n}$

Naive method: compute the secondary polytope $\Sigma(\mathcal{A})$ to compute Π

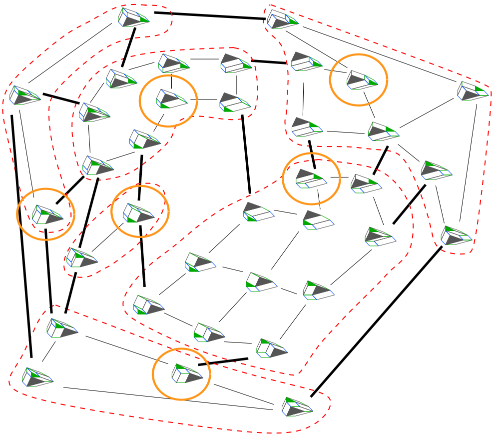


The idea of the algorithm

The input supports define a pointset $\mathcal{A} \in \mathbb{Z}^{2n}$

Naive method: compute the secondary polytope $\Sigma(\mathcal{A})$ to compute Π

Idea: incrementally construct Π using an **oracle** that given a direction produces vertices of Π

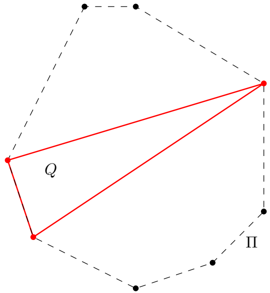


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \Pi$

1. initialization step



initialization:

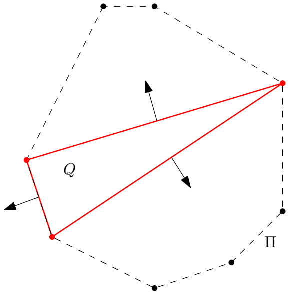
- $Q \subset \Pi$
- dim(Q)=dim(Π)

Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \mathit{\Pi}$

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



2 kinds of hyperplanes of Q_H:

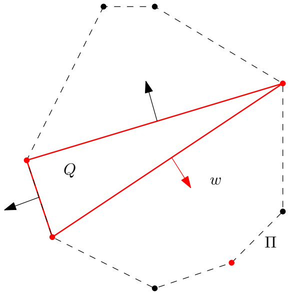
- legal if it supports facet $\subset \mathit{\Pi}$
- illegal otherwise

Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \Pi$

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, $Q_V \leftarrow Q_V \cup \{v\}$

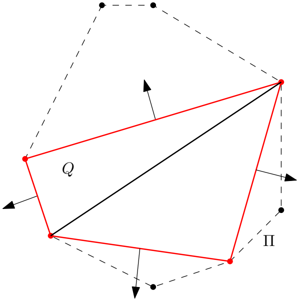


Extending an illegal facet

Incremental Algorithm

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

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



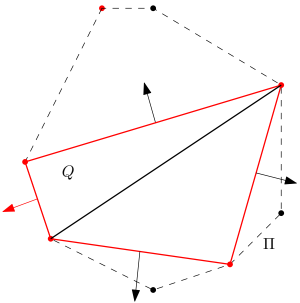
Extending an illegal facet

Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \Pi$

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

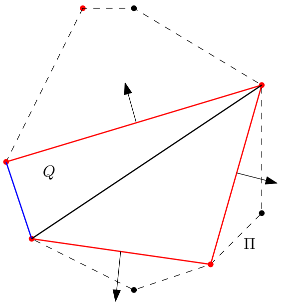


Validating a legal facet

Incremental Algorithm

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

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

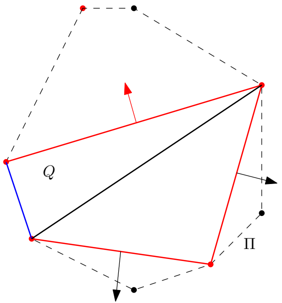


Validating a legal facet

Incremental Algorithm

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

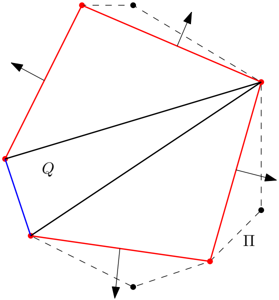
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else`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



Incremental Algorithm

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

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



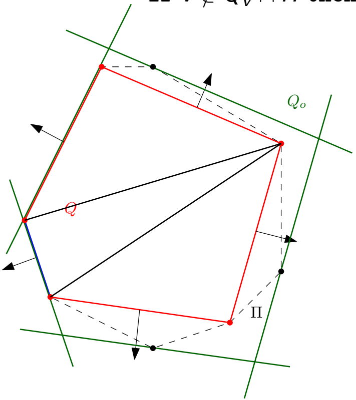
At any step, Q is an inner approximation ...

Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \Pi$

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

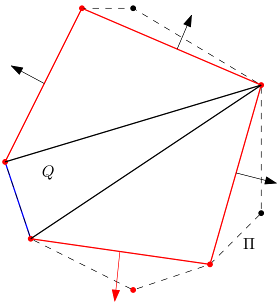


At any step, Q is an inner approximation ... from which we can compute an outer approximation Q_o.

Incremental Algorithm

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

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

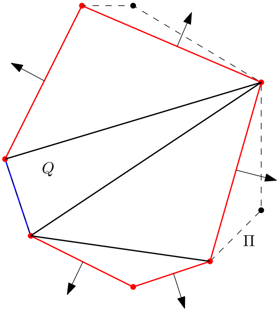


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \Pi$

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

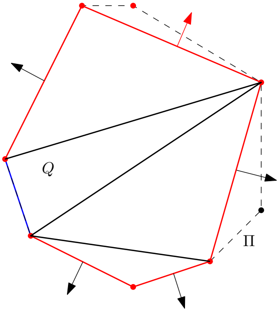


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \Pi$

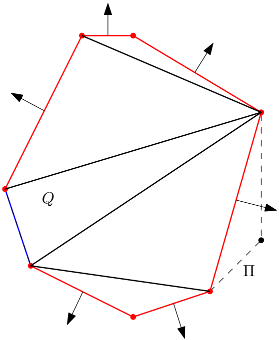
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else`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



Incremental Algorithm

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

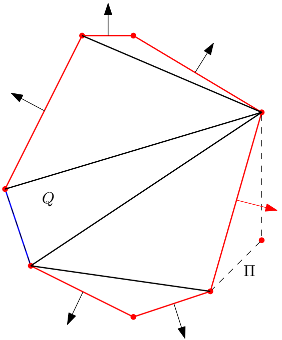
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else`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



Incremental Algorithm

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

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

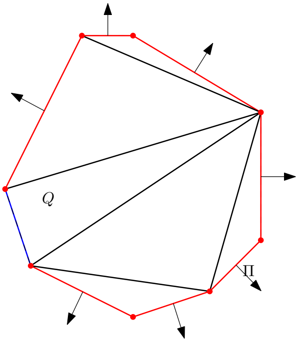


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \Pi$

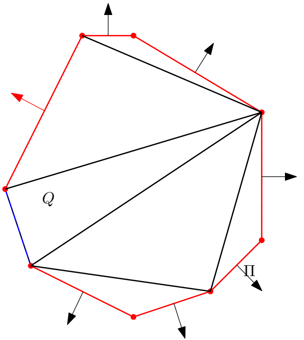
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else`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



Incremental Algorithm

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

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

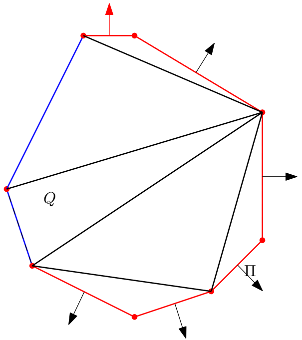


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \Pi$

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

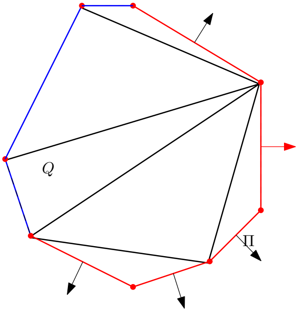


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \Pi$

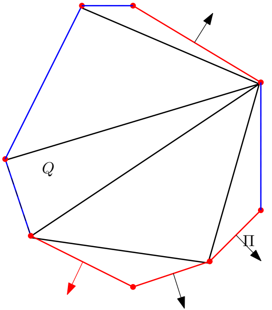
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else`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



Incremental Algorithm

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

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

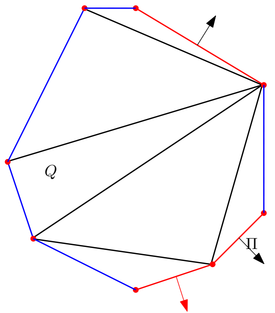


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \Pi$

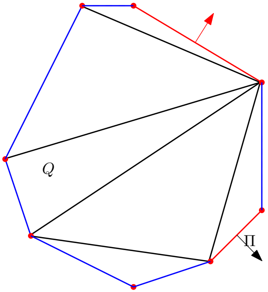
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else`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



Incremental Algorithm

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

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

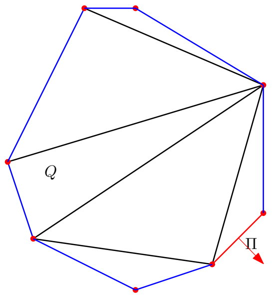


Incremental Algorithm

Input: $\mathcal{A}$

Output: H-rep. Q_H, V-rep. Q_V of $Q = \Pi$

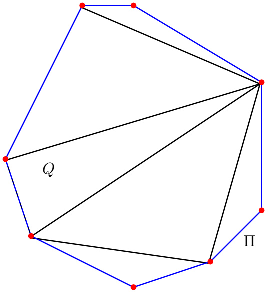
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - call oracle for w and compute v, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else`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



Incremental Algorithm

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

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 Π, as well as a triangulation T of Π, in

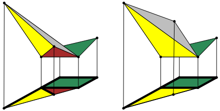
$$O^*(\, m^5\, |vtx(\Pi)| \cdot |T|^2),$$

where $m = \dim \Pi$, and $|T|$ the number of full-dim faces of T.

Elements of proof

- Most computation is done in dimension $\leq m$.
- At most $\leq \mathrm{vtx}(\Pi) + \mathrm{fct}(\Pi)$ oracle calls
- Beneath-Beyond algorithm for converting V-rep. to H-rep. (bottleneck)

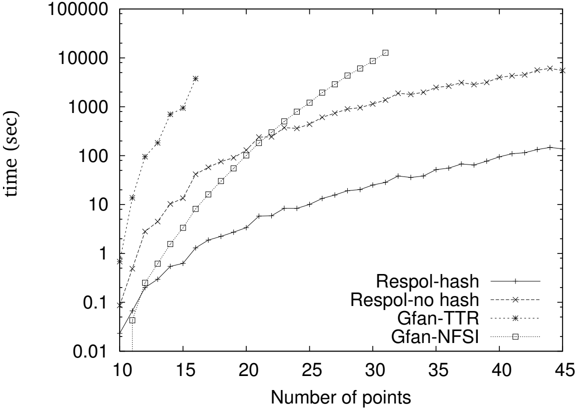
ResPol Implementation



Tools

C++, CGAL, `triangulation` [Boissonnat,Devillers,Hornus], `extreme_points_d` [Gärtner]

Experiments ($dim(\Pi) = 4$)



Future work

- approximate resultant polytopes ($dim(\Pi) \geq 7$)
- preliminary results:

input	m	3	3	4	4	5	5
	$\lvert\mathcal{A}\rvert$	200	490	20	30	17	20
exact	#vtx(Π)	98	133	416	1296	1674	5093
	time	2.03	5.87	3.72	25.97	51.54	239.96
approx.	#vtx(Q_{in})	15	11	63	121	–	–
	vol(Q_{in})/vol(Π)	0.96	0.95	0.93	0.94	–	–
	vol(Q_{out})/vol(Π)	1.02	1.03	1.04	1.03	–	–
	time	0.15	0.22	0.37	1.42	> 10hr	> 10hr

- approximate volume computation [Lovász-Vempala06]

References

The code

- `http://respol.sourceforge.net`

The full version of the paper

- `http://arxiv.org/abs/1108.5985v2`

References

The code

- http://respol.sourceforge.net

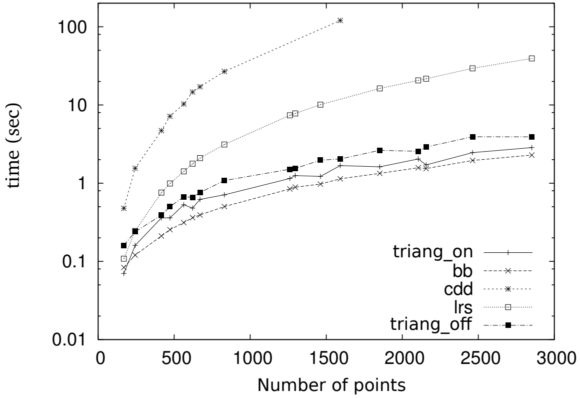
The full version of the paper

- http://arxiv.org/abs/1108.5985v2

Thank You !

Convex hull implementations

- From V- to H-rep. of Π.
- `triangulation (on/off-line)`, polymake `beneath-beyond`, `cdd`, `lrs`



$dim(\Pi) = 4$