An output-sensitive algorithm for computing projections of resultant polytopes

Vissarion Fisikopoulos

Joint work with I.Z. Emiris, C. Konaxis* and L. Peñaranda

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

ΕΡΓΑ-GALAAD New Year's meeting, 9.Jan.2012

The setting

Problem

- given a system of $n+1$ polynomials on n variables ($\mathcal{A} \subset \mathbb{Z}^{2n}$)
- compute the (projection of the) Newton polytope of the resultant or *resultant polytope* Π (for some orthogonal projection π)

Idea

- not compute the whole secondary polytope $\Sigma(\mathcal{A})$
- incrementally construct Π using an oracle that given a direction produces vertices of Π

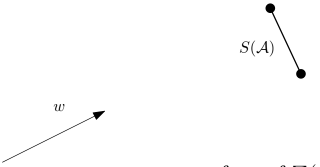
The Oracle

Vertex$\Pi(\mathcal{A}, w)$

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

Output: vertex $\in \Pi$, extremal wrt w

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



face of $\Sigma(\mathcal{A})$

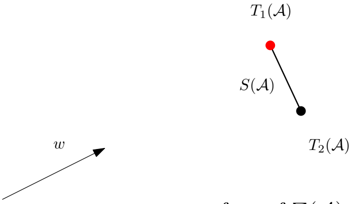
The Oracle

Vertex$\Pi(\mathcal{A}, w)$

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

Output: vertex $\in \Pi$, extremal wrt w

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



face of $\Sigma(\mathcal{A})$

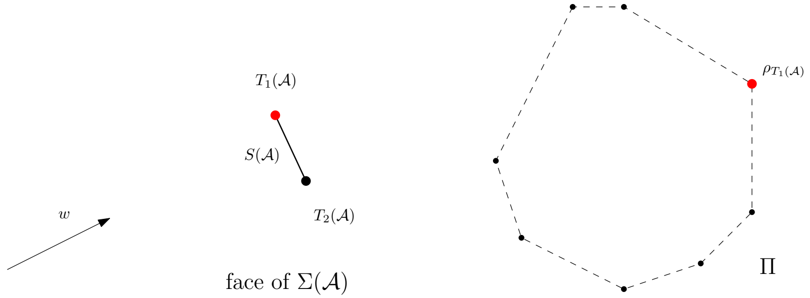
The Oracle

Vertex$\Pi(\mathcal{A}, w)$

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

Output: vertex $\in \Pi$, extremal wrt w

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

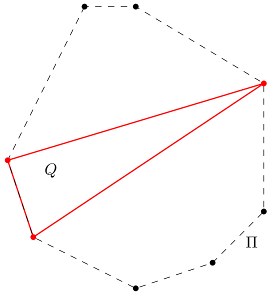


Incremental Algorithm

Input: $\mathcal{A}$

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

1. initialization step



initialization:

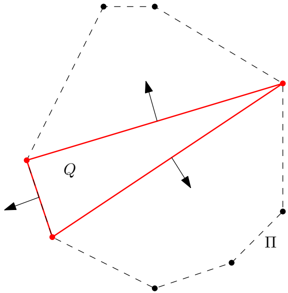
- $Q \subset \Pi$
- dim(Q)=dim(Π)

Incremental Algorithm

Input: $\mathcal{A}$

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

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



2 kinds of hyperplanes of Q_H:

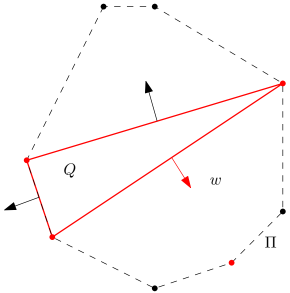
- legal if it supports facet $\subset \Pi$
- illegal otherwise

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`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$

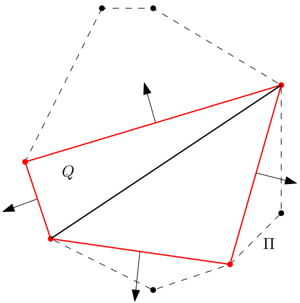


Extending an illegal facet

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`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal

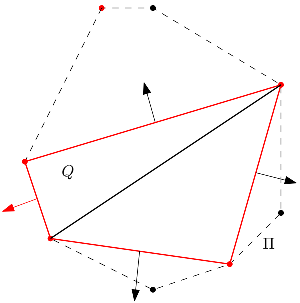


Extending an illegal facet

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`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal

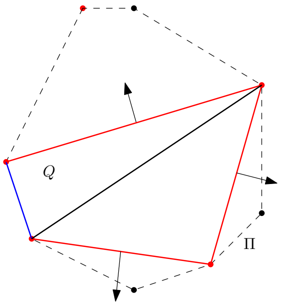


Validating a legal facet

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`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal



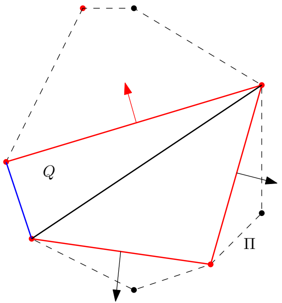
Validating a legal facet

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`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal

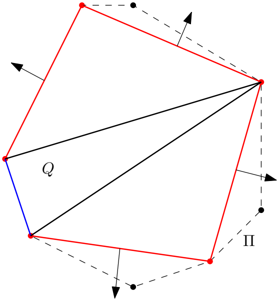


Incremental Algorithm

Input: $\mathcal{A}$

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

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`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal

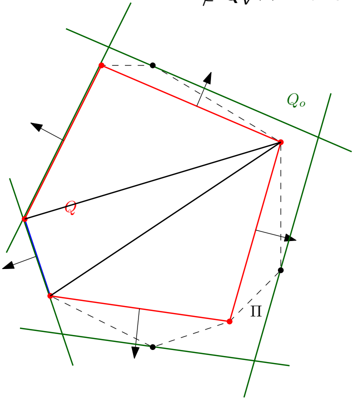


At any step, Q is an inner approximation ...

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`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal



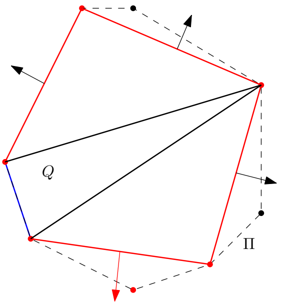
At any step, Q is an inner approximation ... from which we can compute an outer approximation Q_o.

Incremental Algorithm

Input: $\mathcal{A}$

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

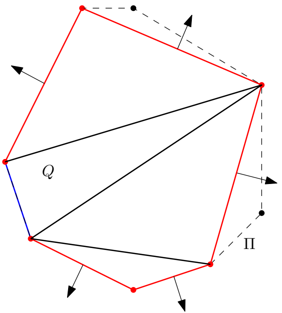
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup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



Incremental Algorithm

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

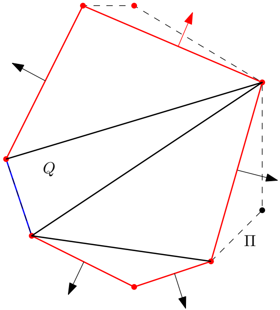
1. initialization step
2. all hyperplanes of Q_H are illegal
3. while $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w do
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - if $v \notin Q_V \cap H$ then $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$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



Incremental Algorithm

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

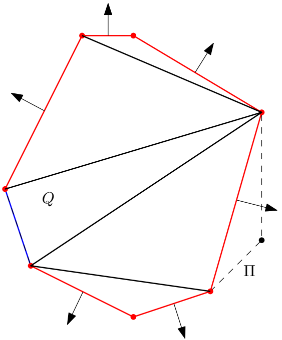
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup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



Incremental Algorithm

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

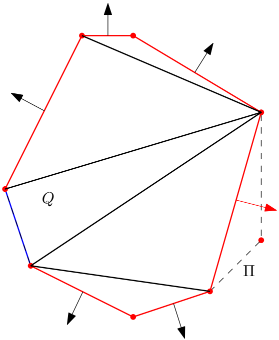
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup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



Incremental Algorithm

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

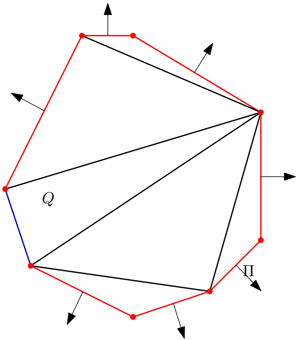
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup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



Incremental Algorithm

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

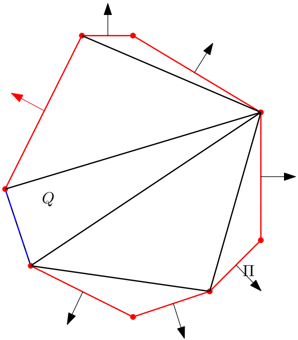
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup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



Incremental Algorithm

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

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`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal

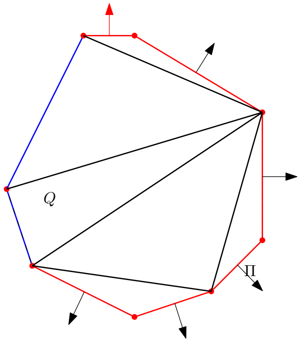


Incremental Algorithm

Input: $\mathcal{A}$

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

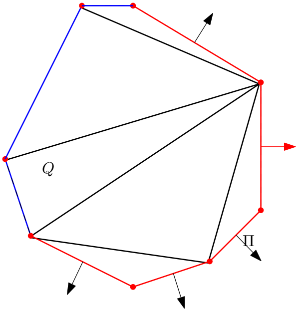
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup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



Incremental Algorithm

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

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`
    - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
    - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal

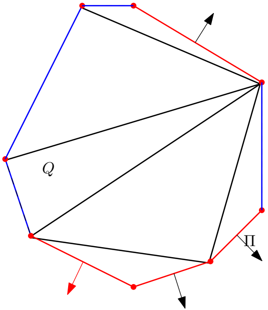


Incremental Algorithm

Input: $\mathcal{A}$

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

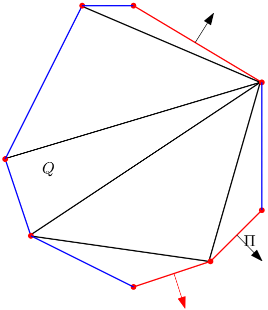
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup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



Incremental Algorithm

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

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`
   - compute $v = \mathrm{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \mathrm{CH}(Q_V \cup \{v\})$ `else` H is legal

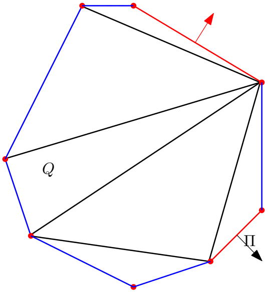


Incremental Algorithm

Input: $\mathcal{A}$

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

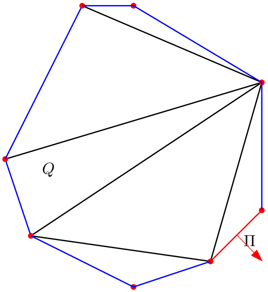
1. initialization step
2. all hyperplanes of Q_H are illegal
3. `while` $\exists$ illegal hyperplane $H \subset Q_H$ with outer normal w `do`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup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



Incremental Algorithm

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

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`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal

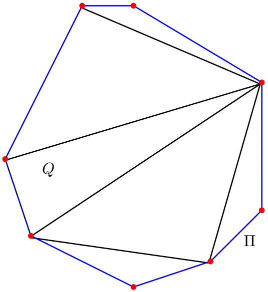


Incremental Algorithm

Input: $\mathcal{A}$

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

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`
   - compute $v = \text{Vertex}\Pi(\mathcal{A}, w)$, $Q_V \leftarrow Q_V \cup \{v\}$
   - `if` $v \notin Q_V \cap H$ `then` $Q_H \leftarrow \text{CH}(Q_V \cup \{v\})$ `else` H is legal



Complexity

Theorem

We compute the Vertex- and Halfspace-representations of Π, as well as a triangulation T of Π, in

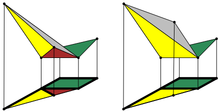
$$O^*(\, m^5 \, |vtx(\Pi)| \cdot |T|^2),$$

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

Elements of proof

- All computation 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]

Hashing of determinantal predicates

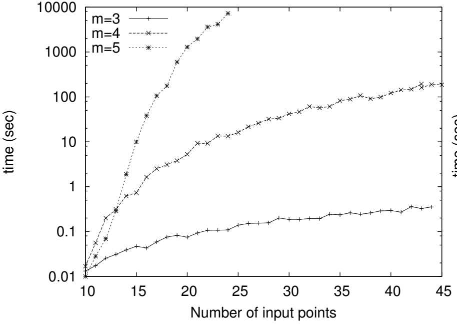
optimizing sequences of similar determinants

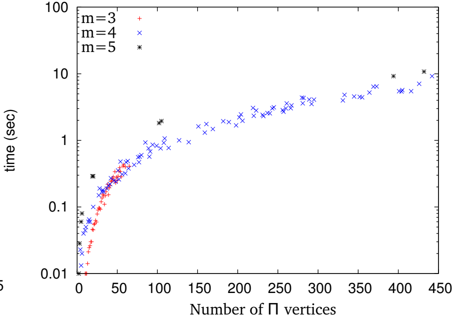
$$\widetilde{\mathcal{A}} = \begin{vmatrix} 0 & 0 & 0 & 1 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 0 & 1 & 2 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 & w_9 \end{vmatrix}$$

- Laplace (cofactor) expansion wrt the last row + Hash minors
- If all needed minors computed, orientation $= O(n^2)$, volume $= O(n)$

Output-sensitivity

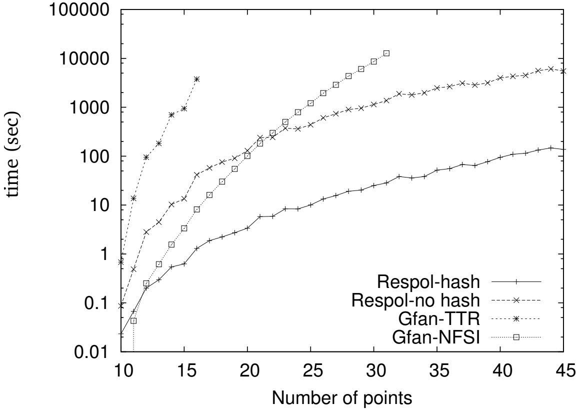
- oracle calls $\leq \text{vtx}(\Pi) + \text{fct}(\Pi)$
- output vertices bound polynomially the output triangulation size
- subexponential runtime wrt to input points (L), output vertices (R)





Hashing and Gfan

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



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 paper

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

References

The code

- http://respol.sourceforge.net

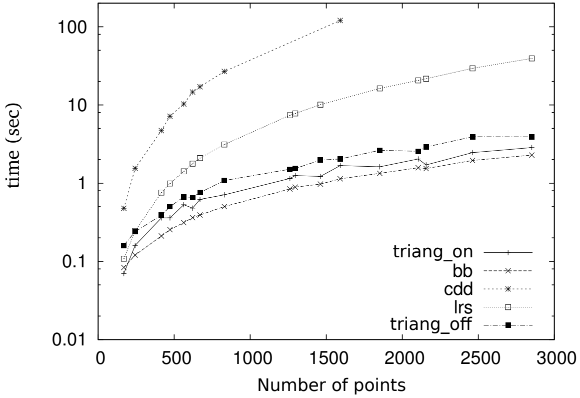
The paper

- 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`



$dim(\Pi) = 4$