An output-sensitive algorithm for computing (projections of) resultant polytopes

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

Department of Informatics, University of Athens

An interesting class of polytopes: resultant polytopes

- **Geometry:** Minkowski summands of secondary polytopes, equivalence classes of secondary vertices, generalization of Birkhoff polytopes
- **Motivation:** useful to express the solvability of polynomial systems
- **Applications:** discriminant and resultant computation, implicitization of parametric hypersurfaces
Existing work

- Theory of resultants, secondary polytopes, Cayley trick [GKZ '94]
- **TOPCOM** [Rambau '02] computes all vertices of secondary polytope.
- [Michiels & Cools DCG'00] describe a decomposition of $\Sigma(A)$ in Minkoski summands, including $N(\mathcal{R})$.
- Tropical geometry [Sturmfels-Yu '08] leads to algorithms for the resultant polytope (GFan library) [Jensen-Yu '11] and the discriminant polytope (TropLi software) [Rincón '12].
What is a resultant polytope?

- Given \( n + 1 \) point sets \( A_0, A_1, \ldots, A_n \subset \mathbb{Z}^n \)
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 \(e_i = (0, \ldots, 1, \ldots, 0) \subset \mathbb{Z}^n\)
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 $\text{conv}(\mathcal{A})$, a cell is $a$-mixed if it is the Minkowski sum of $n$ 1-dimensional segments from $A_j, j \neq i$, and some vertex $a \in A_i$. 

![Diagram of point sets and triangulation](image)
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 \( \text{conv}(\mathcal{A}) \), a cell is \( a \)-mixed if it is the Minkowski sum of \( n \) 1-dimensional segments from \( A_j, j \neq i \), and some vertex \( a \in A_i \).
- \( \rho_T(a) = \sum_{\sigma \in T: a \in \sigma} \text{vol}(\sigma) \in \mathbb{N}, \ a \in \mathcal{A} \)

\[ \begin{align*}
A_0 & \quad a_1 \quad \cdots \quad a_2 \\
A_1 & \quad a_3 \quad \cdots \quad a_4 \\
\mathcal{A} & \quad a_3, 1 \quad \cdots \quad a_4, 1 \\
& \quad a_1, 0 \quad \cdots \quad a_2, 0
\end{align*} \]

\( \rho_T = (0, 2, 1, 0) \)
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 \( \text{conv}(\mathcal{A}) \), a cell is \( a \)-mixed if it is the Minkowski sum of \( n \) 1-dimensional segments from \( A_j, j \neq i \), and some vertex \( a \in A_i \).
- \( \rho_T(a) = \sum_{\sigma \in T: a \in \sigma} a_{-\text{mixed}} \text{vol}(\sigma) \in \mathbb{N}, \quad a \in \mathcal{A} \)
- Resultant polytope \( N(R) = \text{conv}(\rho_T : T \text{ triang. of } \text{conv}(\mathcal{A})) \)
Connection with Algebra

- The **Newton polytope** of $f$, $N(f)$, is the convex hull of 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.

\[
A_0 \quad - \quad - \quad - \quad \bullet \quad f_0(x) = ax^2 + b
\]
\[
A_1 \quad - \quad - \quad - \quad \bullet \quad f_1(x) = cx^2 + dx + e
\]
\[
N(R) \quad - \quad - \quad - \quad \bullet \quad R(a, b, c, d, e) = ad^2b + c^2b^2 - 2caeb + a^2e^2
\]
Connection with Algebra

- The **Newton polytope** of \( f, N(f) \), is the convex hull of 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.

\[
\begin{align*}
A_0 & \quad A_1 \quad A_2 \\
\text{N}(R) & \quad 4\text{-dimensional Birkhoff polytope}
\end{align*}
\]

\[
\begin{align*}
f_0(x, y) &= ax + by + c \\
f_1(x, y) &= dx + ey + f \\
f_2(x, y) &= gx + hy + i \\
R(a, b, c, d, e, f, g, h, i) &= \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix}
\]

Connection with Algebra

- The **Newton polytope** of $f$, $N(f)$, is the convex hull of 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.

$$
\begin{align*}
A_0 & \\
A_1 & \\
A_2 & \\
N(R) & \\
\end{align*}
$$

$$
\begin{align*}
f_0(x, y) &= axy^2 + x^4y + c \\
f_1(x, y) &= dx + ey \\
f_2(x, y) &= gx^2 + hy + i \\
\end{align*}
$$

**NP-hard** to compute the resultant in the general case.
The idea of the algorithm

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

Simplistic method:

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

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

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

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

\[R^{\mid \mathcal{A} \mid}\]

face of \( \Sigma(\mathcal{A}) \)
The Oracle

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

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 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 $\rho_T \in N^{|\mathcal{A}|}$
**The Oracle**

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

**Output:** vertex $\in N(R)$, extremal wrt $\mathbf{w}$

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

**Oracle property:** its output is a vertex of the target polytope (Lem. 5).
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: $A$

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 \( Q_H \leftarrow \text{CH}(Q_V \cup \{v\}) \) else \( H \) is legal

Extending an illegal facet
Incremental Algorithm

Input: $\mathcal{A}$

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

Validating a legal facet
Incremental Algorithm

Input: $A$

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 $Q_H \leftarrow CH(Q_V \cup \{v\})$ else $H$ is legal
Incremental Algorithm

Input: $\mathcal{A}$

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

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 \( Q_H \leftarrow \text{CH}(Q_V \cup \{v\}) \) 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: $\mathcal{A}$

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: $\mathcal{A}$

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
   a. call oracle for $w$ and compute $v$, $Q_V \leftarrow Q_V \cup \{v\}$
   b. if $v \notin Q_V \cap H$ then $Q_H \leftarrow CH(Q_V \cup \{v\})$ else $H$ is legal
Incremental Algorithm

Input: $\mathcal{A}$

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: \( A \)
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 \( Q_H \leftarrow CH(Q_V \cup \{v\}) \) else \( H \) is legal
Incremental Algorithm

Input: $\mathcal{A}$

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: $\mathcal{A}$

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: \( A \)

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: $\mathcal{A}$

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: $A$


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: $\mathcal{A}$

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: $\mathcal{A}$

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: $\mathcal{A}$

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 \( Q_H \leftarrow \text{CH}(Q_V \cup \{v\}) \) 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 $m = \text{dim } N(R)$, and $|T|$ the number of full-dim faces of $T$.

Elements of proof

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

- C++

- 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

- Applications of ResPol on I. Emiris talk this afternoon (CGAL, an Open Gate to Computational Geometry!)
Output-sensitivity

- oracle calls $\leq \text{vtx}(N(R)) + \text{fct}(N(R))$
- 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 $\dim N(R) \leq 6$, else competitive

$$\dim(N(R)) = 4:$$
Ongoing and future work

- approximate resultant polytopes \( \text{dim}(N(R)) \geq 7 \) using approximate volume computation
- combinatorial characterization of 4-dimensional resultant polytopes
- computation of discriminant polytopes

More on I.Emiris talk this afternoon (CGAL, an Open Gate to Computational Geometry!)

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

(figure courtesy of M.Joswig)

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

(Figure courtesy of M. Joswig)

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

Thank You!
Convex hull implementations

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

\[
\dim(N(R)) = 4
\]