Efficient edge-skeleton computation for polytopes defined by oracles

Vissarion Fisikopoulos

Joint work with I.Z. Emiris (UoA), B. Gärtner (ETHZ)

Dept. of Informatics & Telecommunications, University of Athens

ACAC, 22.Aug.2013

Outline

Polytopes & Oracles

Algorithms for polytopes given by oracles

Outline

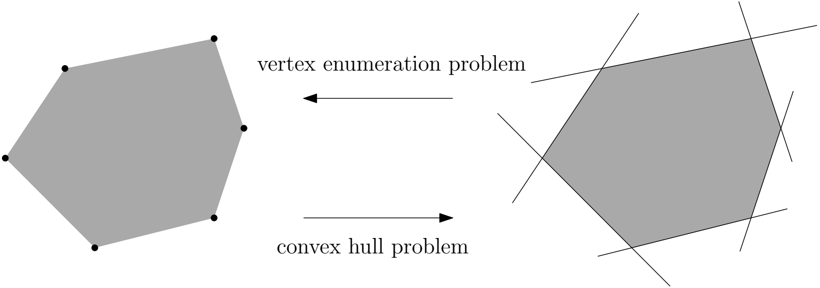
Polytopes & Oracles

Algorithms for polytopes given by oracles

Classical Polytope Representations

A convex polytope $P \subseteq \mathbb{R}^d$ can be represented as the

1. convex hull of a pointset $\{p_1, \ldots, p_n\}$ (V-representation)
2. intersection of halfspaces $\{h_1, \ldots, h_m\}$ (H-representation)



- These problems are equivalent by polytope duality.

Algorithmic Issues

- For general dimension d there is no polynomial algorithm for the convex hull (or vertex enumeration) problem since m can be $O\left(n^{\lfloor d/2 \rfloor}\right)$ [McMullen'70].

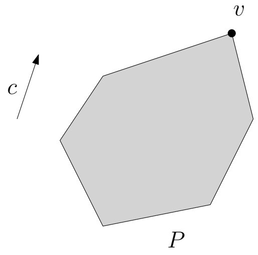
- It is open whether there exist a total poly-time algorithm for the convex hull (or vertex enumeration) problem, *i.e. runs in poly-time in* n, m, d.

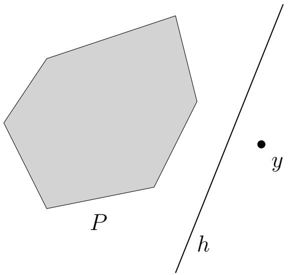
Polytope Oracles

Implicit representation for a polytope $P \subseteq \mathbb{R}^d$.

OPT$_P$: Given direction $c \in \mathbb{R}^d$ return the vertex $v \in P$ that maximizes $c^T v$.

SEP$_P$: Given point $y \in \mathbb{R}^d$, return yes if $y \in P$ otherwise a hyperplane h that separates y from P.





Well-described polytopes and oracles

Definition

A rational polytope $P \subseteq \mathbb{R}^d$ is well-described (with a parameter φ) if there exists an H-representation for P in which every inequality has encoding length at most φ. The encoding length of P is $\langle P \rangle = d + \varphi$.

Proposition (Grötschel et al.'93)

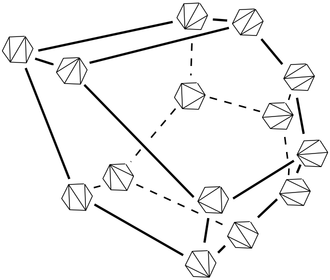
For a well-described polytope, we can compute OPT$_P$ from SEP$_P$ (and vice versa) in oracle polynomial-time. The runtime (polynomially) depends on d and φ.

Why oracles?

- Polynomial time algorithms for combinatorial optimization problems using the ellipsoid method [Grötschel-Lovász-Schrijver'93]
- Randomized polynomial-time algorithm for approximating the volume of convex bodies [Dyer-Frieze-Kannan '90]

Our Motivation

Resultant, Discriminant, Secondary polytopes



- Vertices $\rightarrow$ triangulations of a pointset's convex hull
- $\mathrm{OPT}_{\mathrm{P}}$ is available via a triangulation computation [Emiris-F-Konaxis-Peñaranda '12]

- Applications in Computational Algebraic Geometry, Geometric Modelling, Combinatorics

Outline

Polytopes & Oracles

Algorithms for polytopes given by oracles

Main problems

Vertex enumeration in the oracle model

Given OPT_P for $\mathrm{P} \subseteq \mathbb{R}^\mathrm{d}$, compute the vertices of P.

Vertex enumeration (in the oracle model) with edge-directions

Given OPT_P and a superset D of the edge directions $\mathrm{D}(\mathrm{P})$ of $\mathrm{P} \subseteq \mathbb{R}^\mathrm{d}$, compute the vertices of P.

Remark

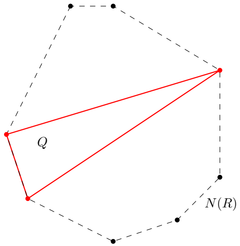
Edge-skeleton $=$ V-representation $+$ edges

Thus, edge-skeleton computation subsumes vertex enumeration.

Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

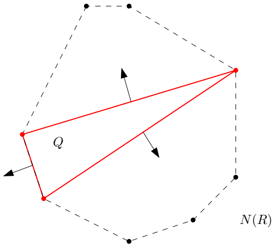
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\longrightarrow$ either validate this halfspace
  $\longrightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

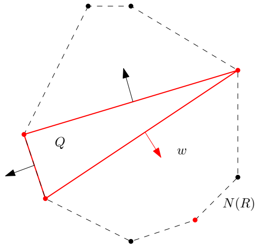
- first compute $d+1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\longrightarrow$ either validate this halfspace
  $\longrightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

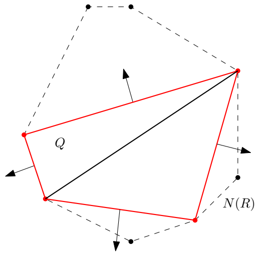
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

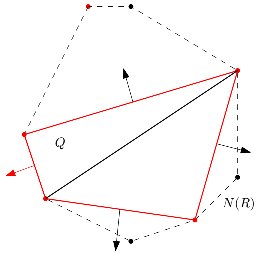
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

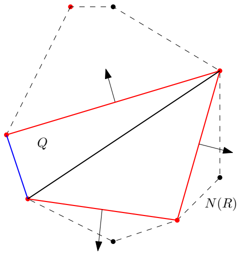
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\longrightarrow$ either validate this halfspace
  $\longrightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

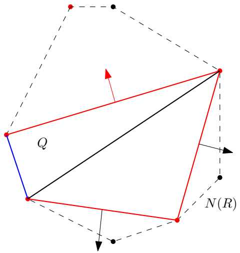
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

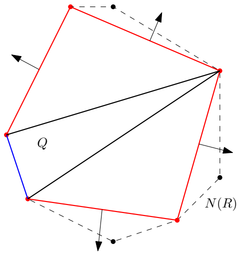
- first compute $d+1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

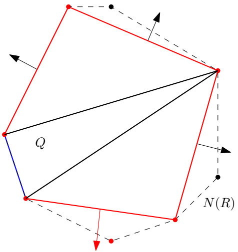
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

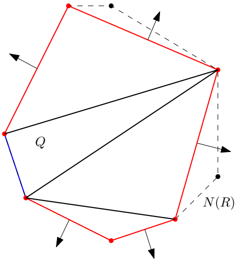
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

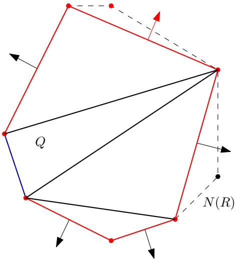
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

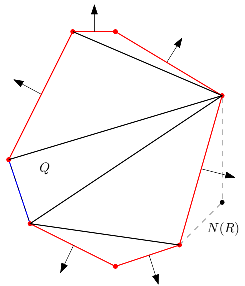
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

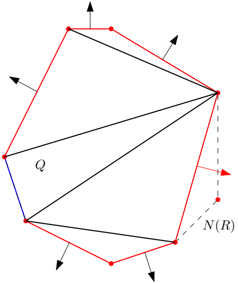
- first compute $d+1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

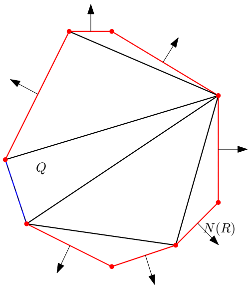
- first compute $d+1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

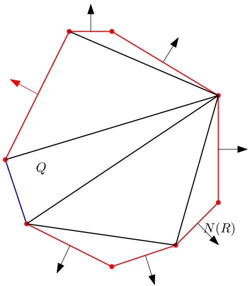
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

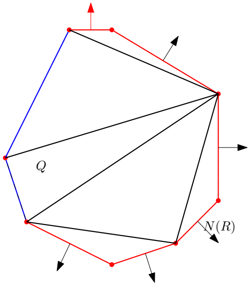
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

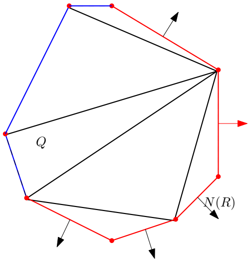
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

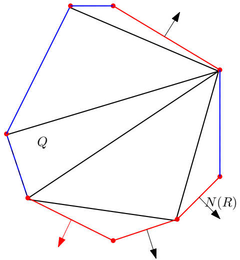
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

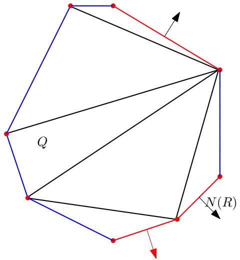
- first compute $d+1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

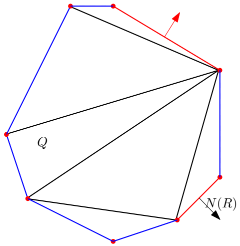
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

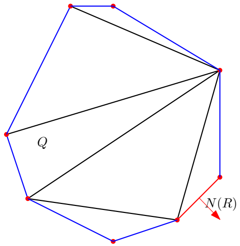
- first compute $d+1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

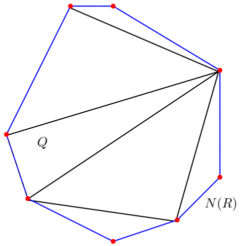
- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

- first compute $d + 1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull



Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

- first compute $d+1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull

Complexity

Given $P \subseteq \mathbb{R}^d$, H-, V-repr. & triang. T of P can be computed in

$$O(d^5 n s^2) \text{ arithmetic operations } + O(n+m) \text{ calls to } \mathrm{OPT}_P$$

s is the number of cells of T.

Vertex enumeration in the oracle model

Algorithm sketch [Emiris-F-Konaxis-Peñaranda '12]

- first compute $d+1$ aff. independent vertices of P and compute their convex hull
- at each step call OPT_P with the outer normal vector of a halfspace and
  $\rightarrow$ either validate this halfspace
  $\rightarrow$ or add a new vertex to the convex hull

Complexity

Given $P \subseteq \mathbb{R}^d$, H-, V-repr. & triang. T of P can be computed in

$$O(d^5 n s^2) \text{ arithmetic operations } + O(n+m) \text{ calls to } \mathrm{OPT}_P$$

s is the number of cells of T.

BUT: s can be $O\left(n^{\lfloor d/2 \rfloor}\right)$

Vertex enumeration with edge-directions

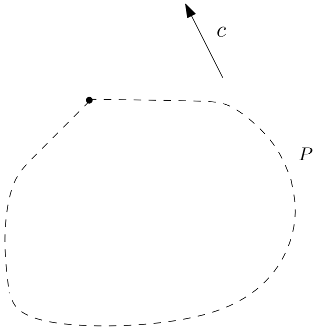
Given $\mathrm{OPT_P}$ and a superset D of the edge directions $\mathrm{D(P)}$ of $\mathrm{P} \subseteq \mathbb{R}^{\mathrm{d}}$, compute the vertices P.

Proposition (Rothblum-Onn '07)

Let $\mathrm{P} \subseteq \mathbb{R}^{\mathrm{d}}$ *given by* $\mathrm{OPT_P}$*, and* $\mathrm{D} \supseteq \mathrm{D(P)}$*. All vertices of* P *can be computed in*

$$\mathrm{O}(|\mathrm{D}|^{\mathrm{d}-1}) \textit{ calls to } \mathrm{OPT_P} + \mathrm{O}(|\mathrm{D}|^{\mathrm{d}-1}) \textit{ arithmetic operations.}$$

The edge-skeleton algorithm



Input:

- OPT_P
- Edge vec. P (dir. & len.): D

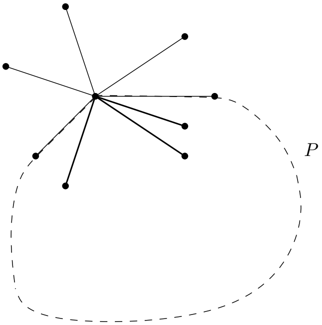
Output:

- Edge-skeleton of P

Sketch of **Algorithm**:

- Compute a vertex of P ($x = \mathrm{OPT}_P(c)$ for arbitrary $c^T \in \mathbb{R}^d$)

The edge-skeleton algorithm



Input:

- OPT_P
- Edge vec. P (dir. & len.): D

Output:

- Edge-skeleton of P

Sketch of **Algorithm**:

- Compute a vertex of P ($x = \mathrm{OPT}_P(c)$ for arbitrary $c^T \in \mathbb{R}^d$)
- Compute segments $S = \{(x, x + d)$, for all $d \in D\}$

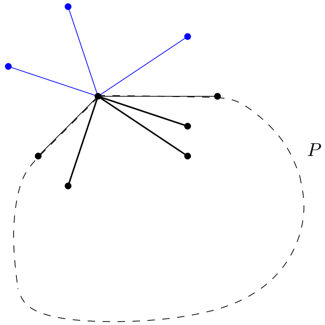
The edge-skeleton algorithm

Input:

- OPT_P
- Edge vec. P (dir. & len.): D

Output:

- Edge-skeleton of P



Sketch of **Algorithm**:

- Compute a vertex of P ($x = \mathrm{OPT}_P(c)$ for arbitrary $c^T \in \mathbb{R}^d$)
- Compute segments $S = \{(x, x + d)$, for all $d \in D\}$
- Remove from S all segments (x, y) s.t. $y \notin P$ ($\mathrm{OPT}_P \rightarrow \mathrm{SEP}_P$)

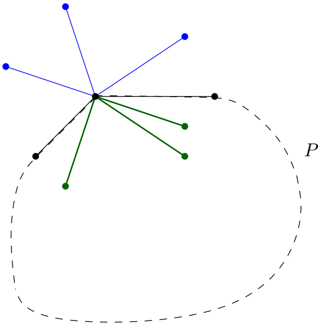
The edge-skeleton algorithm

Input:

- OPT_P
- Edge vec. P (dir. & len.): D

Output:

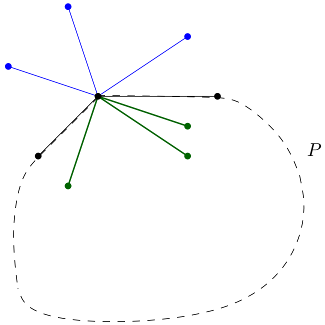
- Edge-skeleton of P



Sketch of **Algorithm**:

- Compute a vertex of P ($x = OPT_P(c)$ for arbitrary $c^T \in \mathbb{R}^d$)
- Compute segments $S = \{(x, x+d)$, for all $d \in D\}$
- Remove from S all segments (x, y) s.t. $y \notin P$ ($OPT_P \rightarrow SEP_P$)
- Remove from S the segments that are not extreme

The edge-skeleton algorithm



Input:

- OPT_P
- Edge vec. P (dir. & len.): D

Output:

- Edge-skeleton of P

Sketch of **Algorithm**:

- Compute a vertex of P ($x = \mathrm{OPT}_P(c)$ for arbitrary $c^T \in \mathbb{R}^d$)
- Compute segments $S = \{(x, x+d)$, for all $d \in D\}$
- Remove from S all segments (x, y) s.t. $y \notin P$ ($\mathrm{OPT}_P \to \mathrm{SEP}_P$)
- Remove from S the segments that are not extreme
- Can be altered to work with edge directions only

Runtime of the edge-skeleton algorithm

Theorem

Given OPT_P *and a superset of edge directions* D *of a well-described polytope* P, *the edge skeleton of* P *can be computed in oracle total polynomial-time*

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

- $\langle D\rangle$ *is the binary encoding length of the vector set* D,
- n *the number of vertices of* P,
- $\mathbb{O}(\langle P\rangle)$: *runtime of oracle conversion algorithm for* P,
- $\mathbb{LP}(\langle A\rangle + \langle b\rangle + \langle c\rangle)$ *runtime of max* $c^T x$ *over* $\{x : Ax \leq b\}$.

Applications

Corollary

The edge skeleton of resultant, secondary and discriminant polytopes (under some genericity assumption) can be computed in oracle total polynomial-time.

Convex combinatorial optimization: generalization of linear combinatorial optimization. [Rothblum-Onn '04]

Convex integer programming: maximize a convex function over the integer hull of a polyhedron. [De Loera et al. '09]

Conclusions

- New & simple algorithm for vertex enumeration of a polytope given by an oracle and known edge directions
- Remove the exponential dependence on the dimension
- First total polynomial time algorithms for resultant, discriminant polytopes (under some genericity assumption)

Future work

- Remove the assumption on the knowledge of edge directions
- Volume computation for polytopes given by optimization oracles

Thank you!