

# **Polyhedral Computation in Algebraic Statistics**

**Komei Fukuda**

**Swiss Federal Institute of Technology**

**Lausanne and Zurich, Switzerland**

**fukuda@ifor.math.ethz.ch**

**<http://www.ifor.math.ethz.ch/~fukuda/>**

**November 14, 2005**

## **Fundamental Problems in Polyhedral Computation**

### **(Closely Related to Algebraic Statistics)**

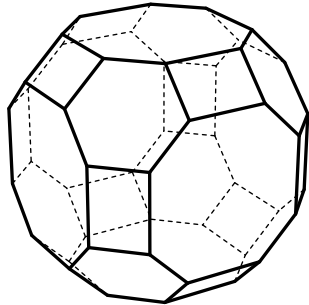
- Representation Conversion (V-Polytope  $\iff$  H-Polytope),
- Redundancy Removal (for V- and H-Polytopes)
- Arrangement/Zonotope Construction
- Minkowski Addition of Polytopes
- Gröbner Walk and Gröbner Fan Construction

### **Ideal Algorithms**

- Time-Efficient Algorithm (Polynomial-Time)
- Space-Efficient Algorithm (Compactness)

# Convex Polyhedra

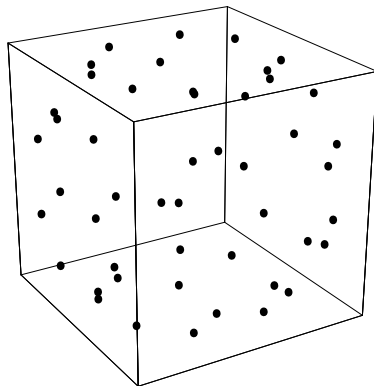
---



A convex polyhedron or simply polyhedron  $P$  in  $\mathbb{R}^d$  is the set of solutions to a (finite) system of linear inequalities in  $d$ -variables:

$$P = \{x \in \mathbb{R}^d : Ax \leq b\}$$

where  $A \in \mathbb{R}^{m \times d}$  and  $b \in \mathbb{R}^m$ . A convex polytope is a bounded polyhedron.



A polyhedron is called H-polyhedron (resp. V-polyhedron) if it is given by an inequality system (resp. a set of generators).

## Quick Tour on Polyhedra: Minkowski-Weyl

---

**Theorem 0.3.** [Minkowski-Weyl's Theorem]

For a subset  $P$  of  $R^d$ , the following statements are equivalent:

(a)  $P$  is an H-polyhedron, i.e., for some matrix  $A$  and vector  $b$ ,

$$P = \{x : Ax \leq b\};$$

(b)  $P$  is a V-polyhedron, i.e., for some vectors  $v_1, v_2, \dots, v_n$  and

$$r_1, r_2, \dots, r_s,$$

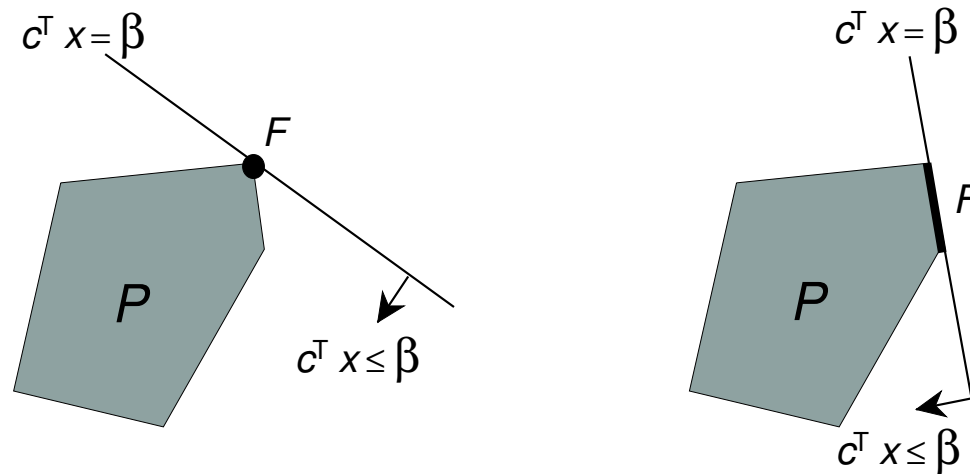
$$P = \text{conv}\{v_1, v_2, \dots, v_n\} + \text{nonneg}\{r_1, r_2, \dots, r_s\}.$$

where  $P + Q = \{p + q : p \in P \text{ and } q \in Q\}$  is the Minkowski sum.

## Quick Tour on Convex Polyhedra: Faces

---

Let  $P = \{x \in \mathbb{R}^d : Ax \leq b\}$  and  $\dim(P) = d$ .



A subset  $F$  of a polyhedron  $P$  is called a face of  $P$  if it is represented as  $F = P \cap \{x : c^T x = \beta\}$  for some valid inequality  $c^T x \leq \beta$ .

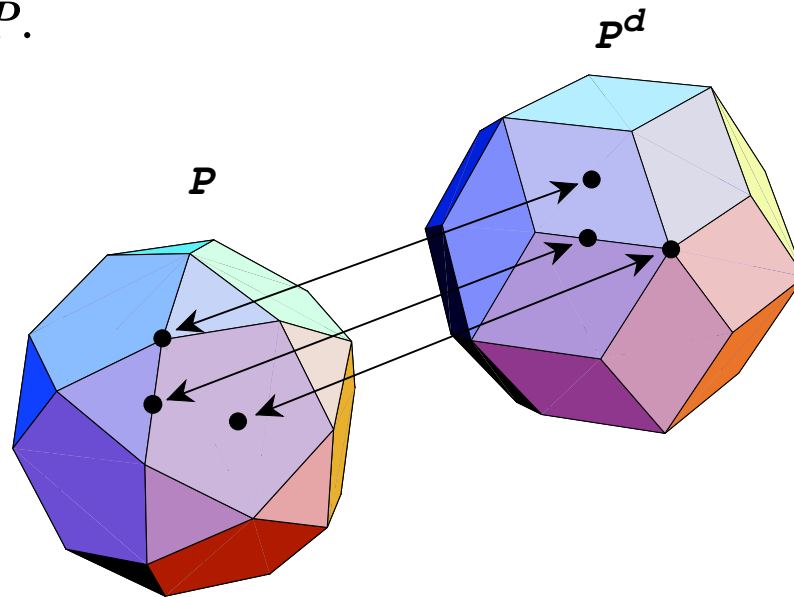
Note that both  $\emptyset$  and  $P$  are faces, called trivial faces. The faces of dimension 0 and  $d - 1$  are called the vertices (extreme points) and the facets.

## Quick Tour on Polyhedra: **Duality**

---

The set of all faces of a polytope  $P$ , denoted by  $\mathcal{F}(P)$ , ordered by set inclusion is called the face lattice of the polytope.

**Theorem.** [Duality of Polytopes] For any polytope  $P$ , there exists a polytope  $P^d$  (called dual) whose face lattice is isomorphic to the polar of the face lattice of  $P$ .



## Quick Tour on Polyhedra: Upper Bound Theorem

---

Denote by  $f_i(P)$  the number of  $i$ -dimensional faces of  $P$ .

**Theorem.** [Upper Bound Theorem, McMullen '70] For any  $d$ -polytope  $P$  with  $n$  vertices, the following inequality holds for each  $i = 0, \dots, d - 1$ :

$$f_i(P) \leq f_i(C(n, d)).$$

where  $C(n, d)$  is a cyclic  $d$ -polytope with  $n$  vertices.

In particular, for  $i = d - 1$ ,

$$f_{d-1}(C(n, d)) = \binom{n - \lfloor (d+1)/2 \rfloor}{n-d} + \binom{n - \lfloor (d+2)/2 \rfloor}{n-d}.$$

Thus, for any fixed  $d$ ,  $f_{d-1} = O(n^{\lfloor d/2 \rfloor})$ .

## Quick Tour on Polyhedra: **Random Polytopes**

---

**Theorem.** [Random Points on the Sphere, Buchta et al '85]

Let  $P$  be the convex hull of  $n$  points randomly chosen from the unit sphere in  $\mathbb{R}^d$ . Then,

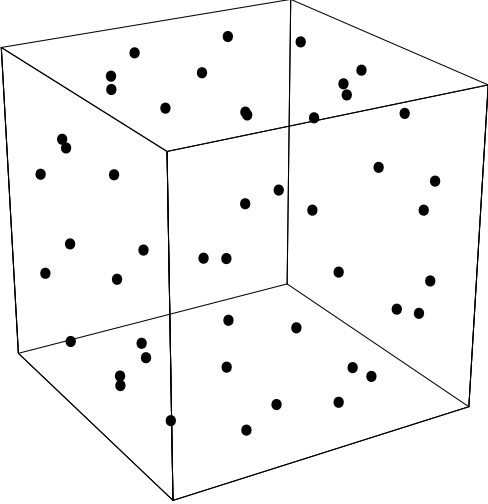
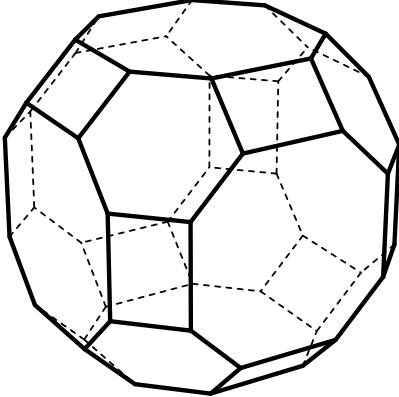
$$E(f_{d-1}(P)) = g(d)(n + o(1)),$$

where  $g(d) = \frac{2}{d}\gamma((d-1)^2)\gamma(d-1)^{-(d-1)}$ .

For example:

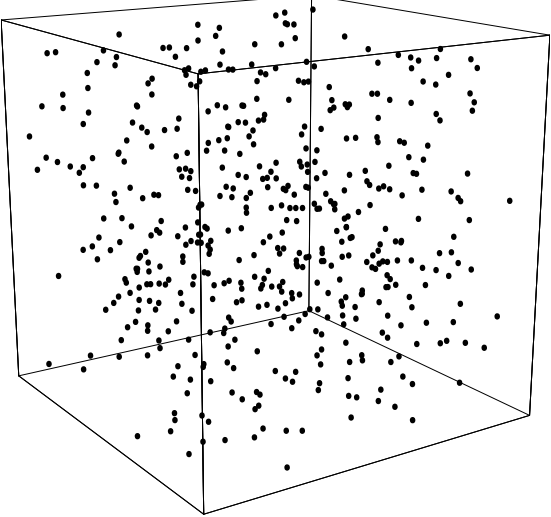
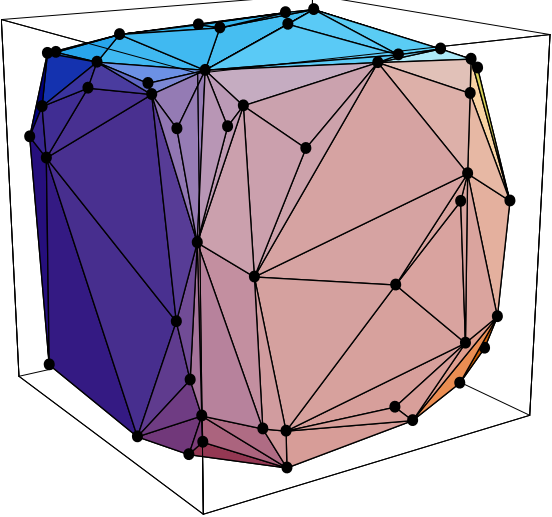
$d$	4	5	6	7	8	9	10
$g(d)$	6.77	31.78	186.7	1296	10262	90425	872190

# Facet Listing (Representation Conversion)

<b>Input</b> $A$	<b>Output</b> $\lambda_{\text{EXT}}(A)$
 <p data-bbox="347 991 1010 1139">a set of <math>n(= 48)</math> points in <math>d(= 3)</math> space, a V-polytope</p>	 <p data-bbox="1288 991 1827 1139">all <math>m(= 26)</math> inequalities, an H-polytope</p>

- It is also known as the **Convex Hull** problem.
- The reverse problem **Vertex Listing** is equivalent by duality.
- For  $d = 2, 3$ , there is an optimal  $O(n \log n)$  algorithm.

## Redundancy Removal (for V-Polytopes)

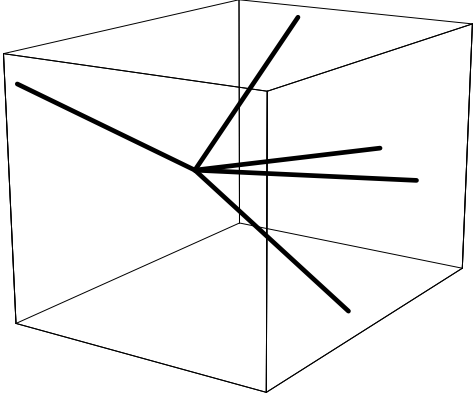
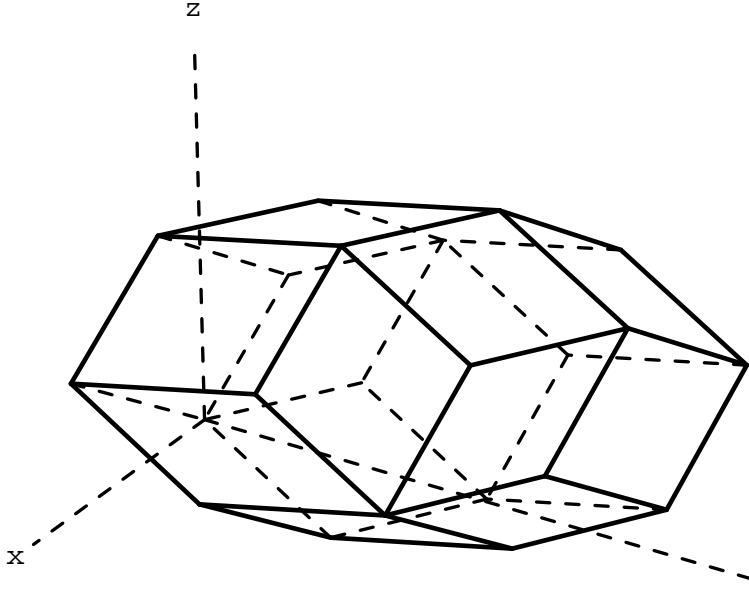
<b>Input</b> $A$	<b>Output</b> $\lambda_{\text{ESS}}(A)$
 <p data-bbox="347 986 1010 1133">a set of <math>n(= 500)</math> points in <math>d(= 3)</math> space, a V-polytope</p>	 <p data-bbox="1249 986 1861 1133">all <math>n'(= 69)</math> extreme points, a minimal V-representation</p>

- In general, **much easier than the representation conversion.**  
(One can compute it for very large  $d$ , by solving many LPs.)
- Yet, in lower dimensions (2, 3), it is faster to use the repr. conversion.

# Arrangement Construction

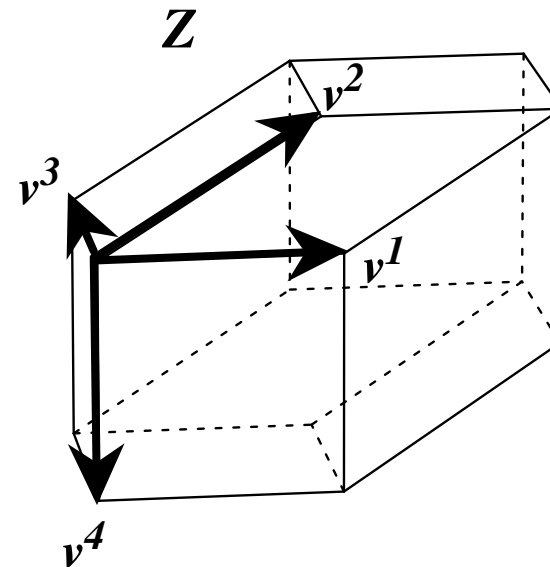
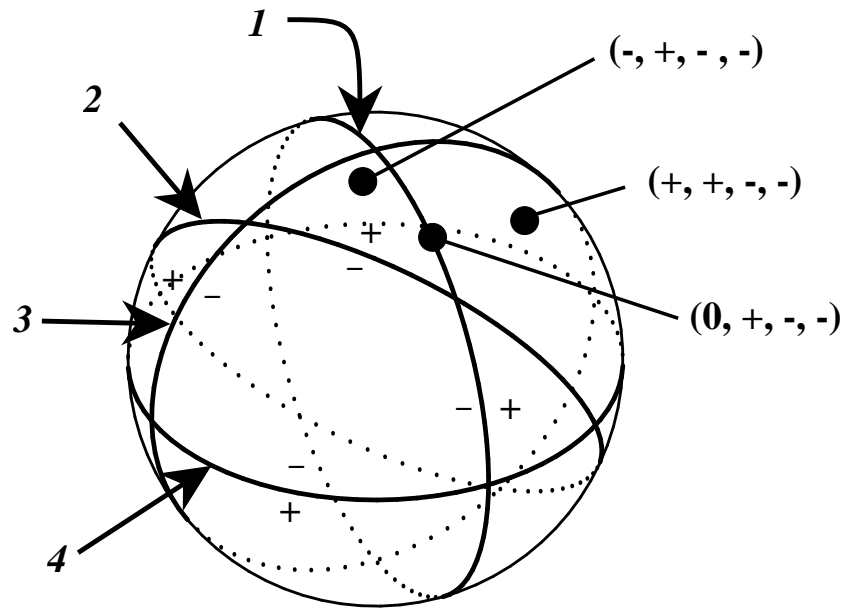
Input $V$	Output $\lambda_{\text{CELL}}(V)$
<div data-bbox="392 518 1120 1077" data-label="Diagram"> </div> <p data-bbox="448 1236 1019 1388"><math>k(=4)</math> hyperlane normals in dimension <math>d(=3)</math></p>	<p data-bbox="1332 614 1926 1045"> <math>(-, +, -, -)</math>, <math>(+, +, -, -)</math>,  <math>(-, -, -, -)</math>, <math>(+, -, -, -)</math>,  <math>(+, +, +, -)</math>, <math>(-, -, +, -)</math>,  <math>(-, +, +, -)</math>,                      and the negatives.                 </p> <p data-bbox="1265 1236 2004 1388">all 14 sign vectors (the underlying oriented matroid)</p>

# Zonotope Construction

<b>Input</b> $I_1, I_2, \dots, I_k$	<b>Output</b> $\lambda_{\text{ZO}}(I_1, I_2, \dots, I_k)$
 <p data-bbox="338 1219 808 1369"><math>k(= 5)</math> line segments in dimension <math>d(= 3)</math></p>	 <p data-bbox="1133 1219 1727 1369">all <math>n(= 22)</math> extreme points of <math>I_1 + I_2 + \dots + I_k</math></p>

Minkowski-sum:  $P = P_1 + P_2 := \{v_1 + v_2 : v_1 \in P_1 \text{ and } v_2 \in P_2\}$ .

# Duality of Arrangements and Zonotopes



Cells

$$X = (-, +, -, -) \iff$$

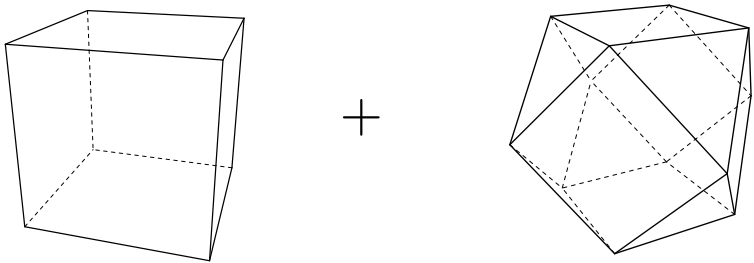
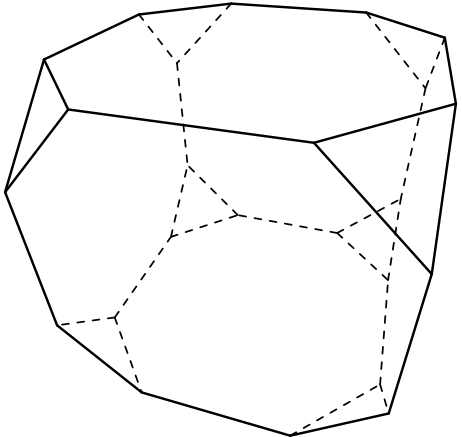
$$X = (+, +, -, -) \iff$$

Extreme points

$$z = v^2$$

$$z = v^1 + v^2$$

# Minkowski Addition of V-Polytopes

<b>Input</b> $V_1, V_2, \dots, V_k$	<b>Output</b> $\lambda_{\text{MI}}(V_1, V_2, \dots, V_k)$
<div style="text-align: center;">  </div> <p style="text-align: center;"> <math>k(= 2)</math> V-polytopes  in dimension <math>d(= 3)</math> </p>	<div style="text-align: center;">  </div> <p style="text-align: center;"> all <math>n(= 24)</math> extreme points,  a V-representation of the sum </p>

Minkowski-sum:  $P = P_1 + P_2 := \{v_1 + v_2 : v_1 \in P_1 \text{ and } v_2 \in P_2\}$ .

## Gröbner Basis Listing (Gröbner Fan Construction)

Input $V$	Output $\lambda_{\text{GR}}(V)$
<p>A polynomial ideal <math>I =</math>  <math>\langle b - a^2, c - a^3, d - a^6, b^3 - d \rangle</math>  <math>\subset \mathbb{C}[a, b, c, d]</math></p> <p><math>n(= 4)</math> generating polynomials  in <math>d(= 4)</math> variables</p>	<p><math>\mathcal{G}_0 = \{b - a^2, c - a^3, d - a^6\},</math>  <math>\mathcal{G}_1 = \{c^2 - d, ab - c, b^2 - ac, a^2 - b\},</math>  <math>\mathcal{G}_2 = \{c^2 - d, a^3 - c, b - a^2\},</math>  <math>\vdots</math>  <math>\mathcal{G}_{11} = \{a^6 - d, b - a^2, c - a^3\}.</math></p> <p>all <math>m(= 12)</math> reduced Gröbner bases</p>

For example,  $b^3 - d$  is redundant in the **input**, because

$$b^3 - d = (b^2 + ba^2 + a^4)(b - a^2) + (-1)(d - a^6).$$

## Ideal Algorithms?

There are no uniformly accepted complexity notions for LISTING problems for which the **output** size can be LARGE.

Nevertheless, one can extend the notion of polynomial algorithms naturally.

- An algorithm is polynomial if it runs in TIME polynomial in both the **input** size and the **output** size. (Some people call this “output polynomial” or “output sensitive”.)
- An algorithm is compact if it runs in SPACE polynomial in the **input** size ONLY.

An ideal algorithm is a compact polynomial algorithm.

[Alternative goal: Worst-output-case optimal algorithms.]

## Current Status of General Dimensional Polyhedral Computation

Problem	Algorithms	Eff.	Implementations
Representation conversion	IS (Motzkin'53, Grünbaum'63, etc) RS (Avis-KF '91) PD (Bremner-KF-Marzetta '96)	!po, !co po*, co po*, co*	cdd, cgal, qhull lrs pd (based on lrs)
Arr./Zonotope construction	IS (Edelsbrunner et al '86) RS (Avis-KF '92)	!po, !co po, co	rs_tope(+cddlib)
Minkowski addition	IS (Gritzmann-Sturmfels'93) RS (KF '02)	!po, !co po, co	minksum(+cddlib)
Gröbner bases	RS (KF-Jensen-Thomas 04')	?	gfan(+cddlib)

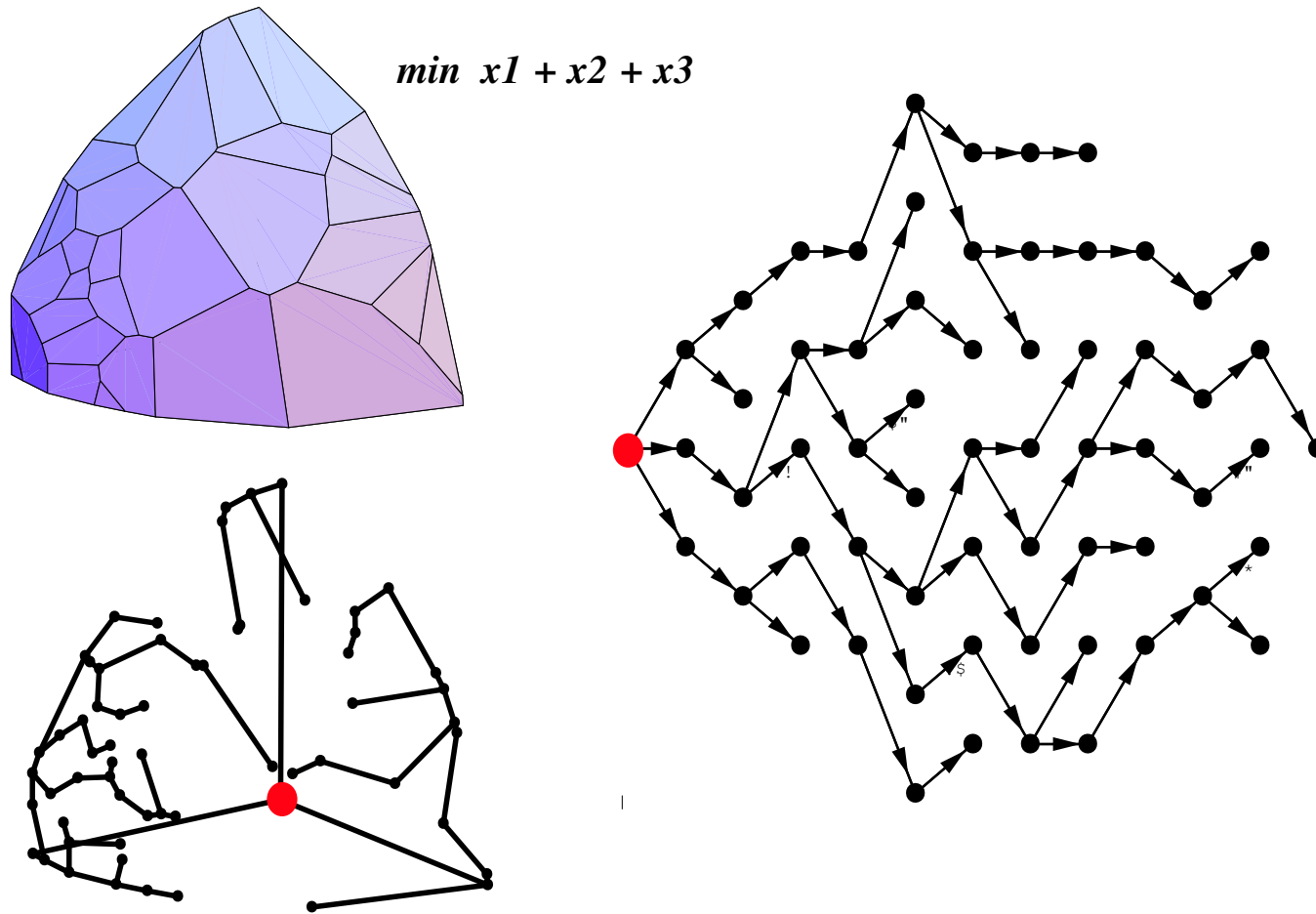
po=polynomial; co= compact; (\*) under non-degeneracy

IS = Incremental Search; RS=Reverse S.; PD=Primal-Dual S.

cdd(KF),cgal(many),gfan(Jensen),qhull(Barbar),lrs(Avis),minksum(Weibel),pd(Marzetta)

## Reverse Search for Vertex Listing

Reverse the simplex method from the **optimal vertex** in all possible ways:



**Complexity:**  $O(mdf_0)$ -time and  $O(md)$ -space (under nondegeneracy).

## Certificates for Vertex Enumeration?

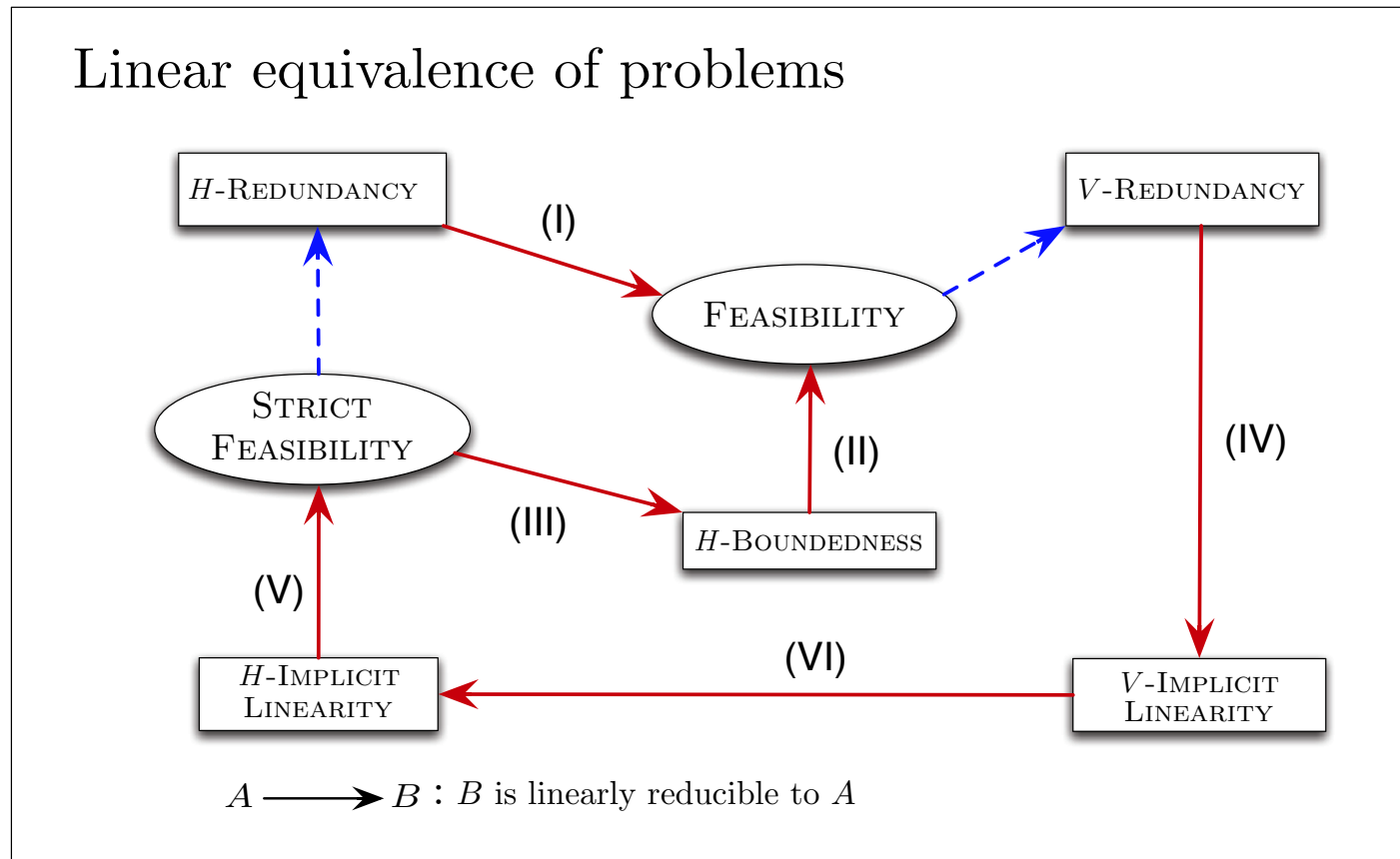
### Polytope Verification Problem (Lovasz):

Given a rational **H**-polytope  $P$  and a rational **V**-polytope  $P'$ , decide whether  $P = P'$ .

- PVP is clearly in coNP.
- No one knows whether PVP is in coNPC.  
(The **hardness result** has been given for polyhedra, i.e. the unbounded case, by Khachiyan et al in 2005.)
- The representation conversion is polynomially solvable if and only if PVP is in P. See Polyhedral Computation FAQ [2].

# Complexity of Redundancy Removal

**Theorem.** (KF-Liebling-Picozzi '05)



**H-Redundancy:** Given  $A \in \mathbb{Q}^{m \times d}$ ,  $b \in \mathbb{Q}^m$  and  $i \in [m]$ , determine whether  $A_i x \leq b_i$  is redundant in the system  $A x \leq b$ .

## Complexity of Redundancy Removal

By the linear equivalence theorem, the H-redundancy (or V-redundancy) checking takes time proportional to  $LP(m, d)$ , that is, the time necessary to solve a linear program of size  $m \times d$ .

However, one can do better to remove **all** redundancies than the trivial bound  $m \times LP(m, d)$ .

**Theorem.** (Clarkson '94)

An algorithm to detect **all** redundancies from an H(V)-representation in time  $m \times LP(s, d)$  exists, where  $s (\leq m)$  is the number of essential inequalities (points).

**Addition in Polytope Algebra:** Computing the convex hull of the union of V-polytopes is essentially the V-redundancy removal.

## Arrangement and Zonotope Construction

There are different approaches.

**Theorem** [Edelsbrunner-O'Rourke-Seidel '86].

For  $d \geq 3$ , there exists an **incremental algorithm** to generate all vertices of a zonotope given by  $k$  generators in  $\mathbb{R}^d$  in  $O(k^{d-1})$  time and  $O(k^{d-1})$  space for fixed  $d$ .

This algorithm is worst-case optimal, but it is neither polynomial nor compact.

**Theorem** [Avis-KF '96 and Ferrez-KF-Liebling '01].

There exists a **reverse search algorithm** to generate all  $v$  vertices in  $O(k \text{ LP}(k, d) v)$  time and  $O(k d)$  space.

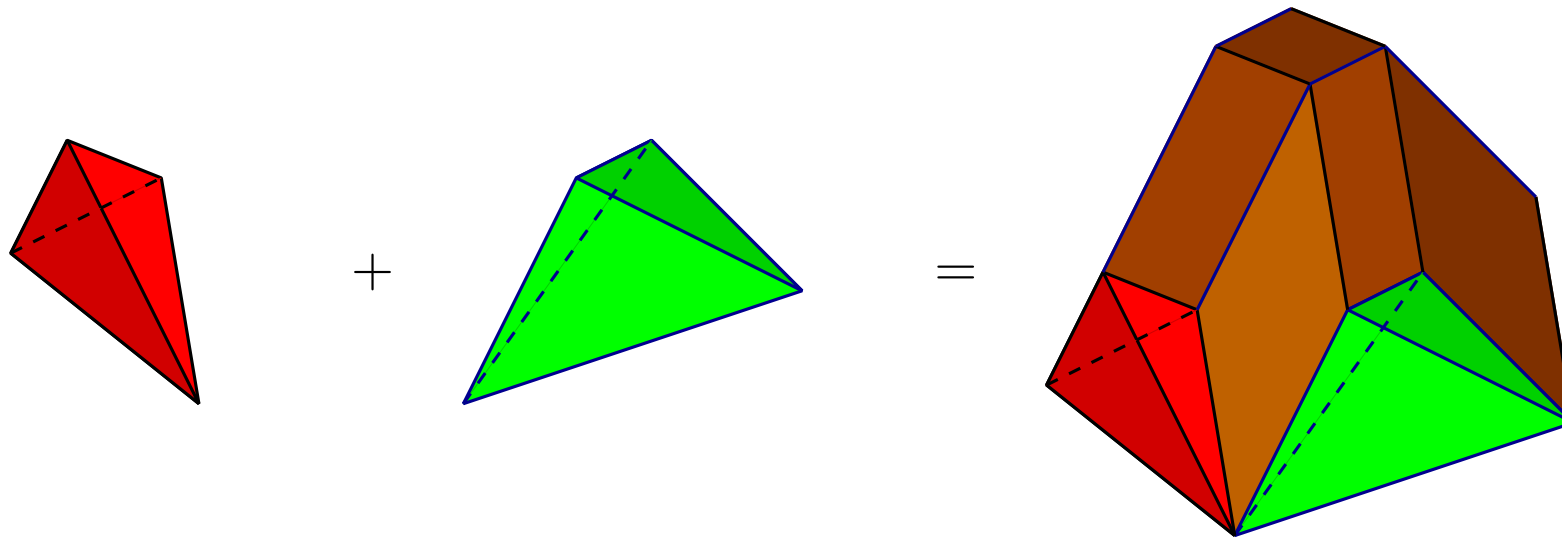
This algorithm is both polynomial and compact.

## Complexity of Minkowski-Addition of Polytopes

**Theorem (Tight Upper Bound)** [KF-Weibel '05 [6]].

In dimension  $d \geq 3$ , it is possible to choose  $k (\leq d - 1)$  polytopes  $P_1, \dots, P_k$  so that the trivial upper bound for the number of vertices is attained by their Minkowski sum.

$$f_0(P_1 + P_2 + \dots + P_k) = f_0(P_1) \times f_0(P_2) \times \dots \times f_0(P_k).$$



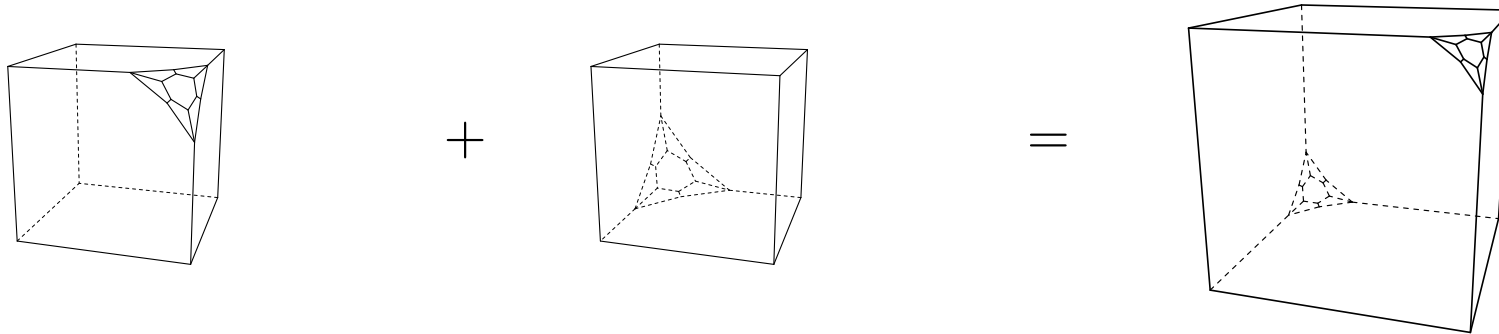
## Complexity of Minkowski-Addition of Polytopes

**Theorem (Zonotope Bound)** [Gritzmann-Sturmfels '93].

Let  $m$  be the number of non-parallel edges in  $P_1, P_2, \dots, P_k$ . Then,

$$f_0(P_1 + P_2 + \dots + P_k) \leq 2 \sum_{h=0}^{d-1} \binom{m-1}{h} = O(m^{d-1}) \quad [d \text{ fixed}].$$

**Proposition (Linearly Bounded Minkowski-Addition).** For each  $k \geq 2$  and  $d \geq 2$ , there is an infinite family of Minkowski additions for which  $f_0(P_1 + P_2 + \dots + P_k) \leq f_0(P_1) + f_0(P_2) + \dots + f_0(P_k)$ .

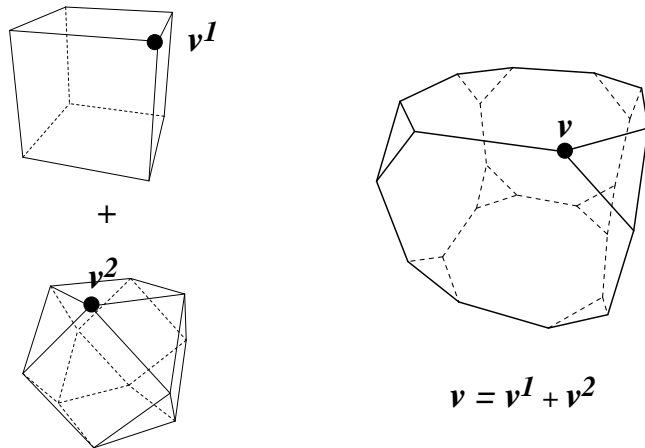


## Gritzmann-Sturmfels' Algorithm I (1993)

This is an input-polynomial algorithm for fixed  $k$ .

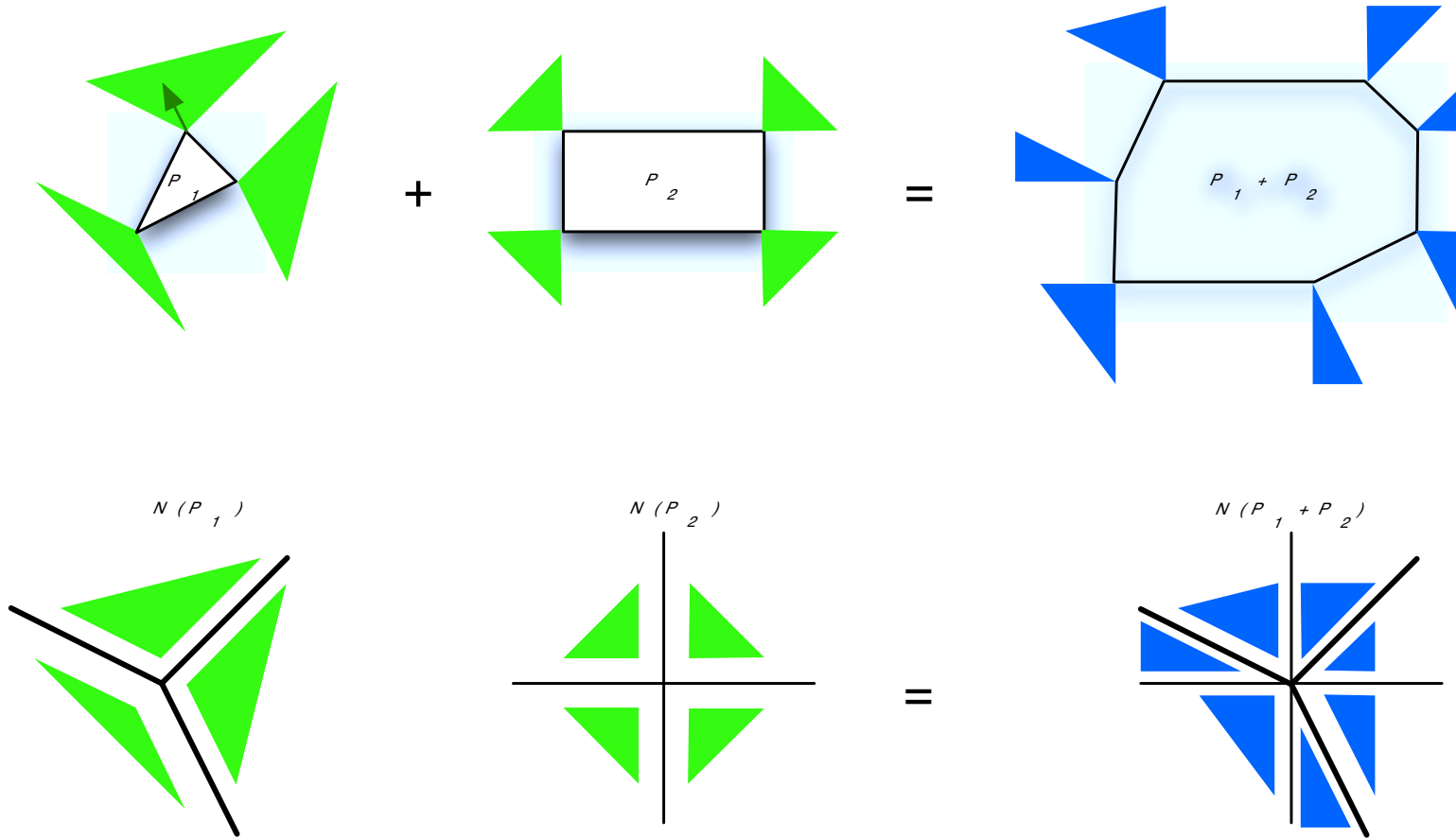
**Input:**  $V_1, V_2, \dots, V_k$

**Algorithm:** For all tuples  $(v^1, v^2, \dots, v^k)$  with  $v^i \in V_i$ ,  
decide whether  $v = v^1 + v^2 + \dots + v^k$  is extreme in  $P_1 + P_2 + \dots + P_k$ .  
(This can be done by solving an LP.)



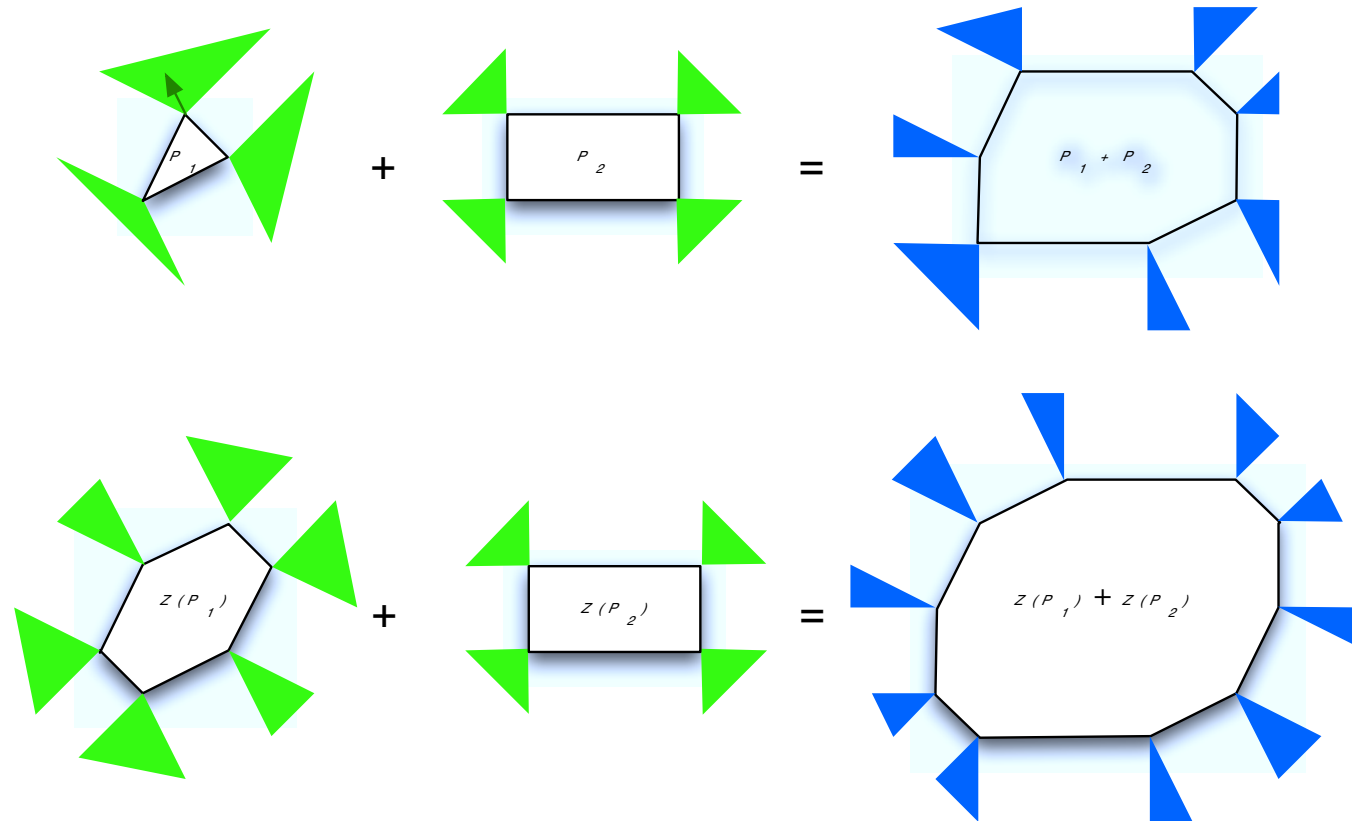
**Complexity:**  $O(s \cdot LP(s - d, s))$ , where  $s = |V_1| \times |V_2| \times \dots \times |V_k|$ .

# Minkowski Addition and Outer Normal Cones/Fans



Computing the Minkowski addition can be considered as superimposing the fans of outer normal cones.

# Gritzmann-Sturmfels' Algorithm II (1993)

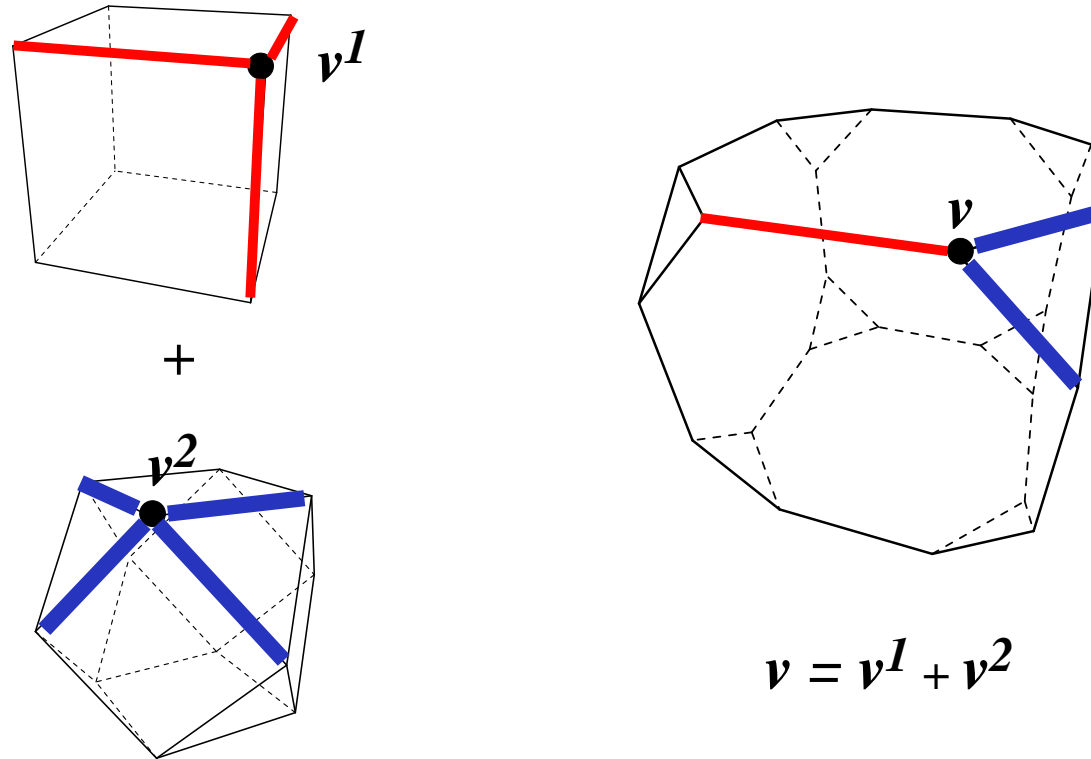


Compute  $N(Z(P_1) + Z(P_2))$  by the incremental zonotope construction algorithm in  $O(m^{d-1})$  time and  $O(m^{d-1})$  space. Then, merge some cones to get  $N(P_1 + P_2)$ .

## New Idea: Adjacency in the Minkowski Addition

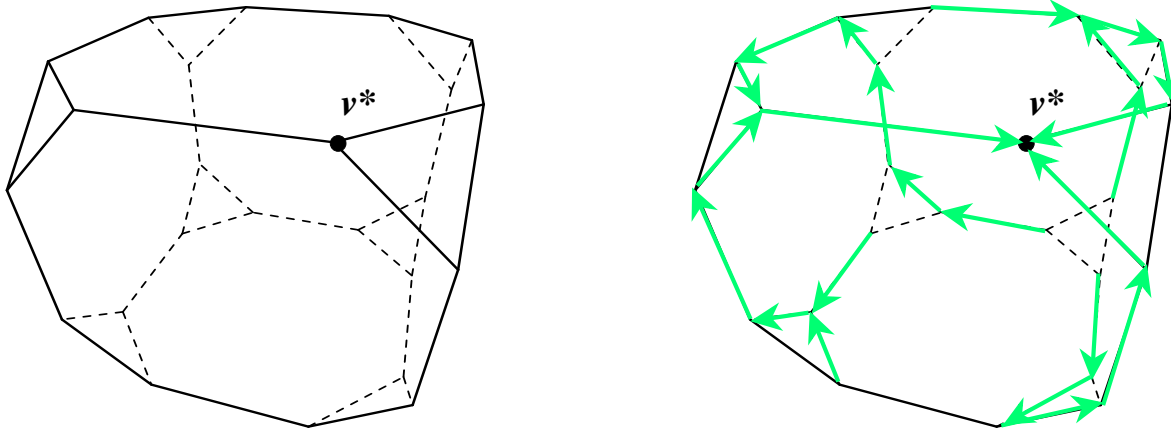
Listing all neighbors of a given vertex  $v$  is easy via LPs.

They are inherited from adjacency in the corresponding vertices of  $P_i$ 's.



## Reverse Search for the Minkowski Addition

Define a **unique directed spanning tree** rooted at any fixed vertex  $v^*$ , e.g, the simplex pivot tree (with a fixed rule).



A reverse search algorithm **traces reversely** the tree from the root  $v^*$  in depth-first manner, using an adjacency oracle.

**Time complexity:**  $O(\delta LP(\delta, d) f_0(P))$ , where  $\delta$  is the sum of the max degrees of  $G(P_i)$ 's.

## An Implementation of Minkowski Sum by Weibel (2005)

- A faithful implementation of the reverse search algorithm by KF('04), freely available [8].
- It is written in C++, relying on GMP and the rational arithmetic LP code of cddlib by KF.

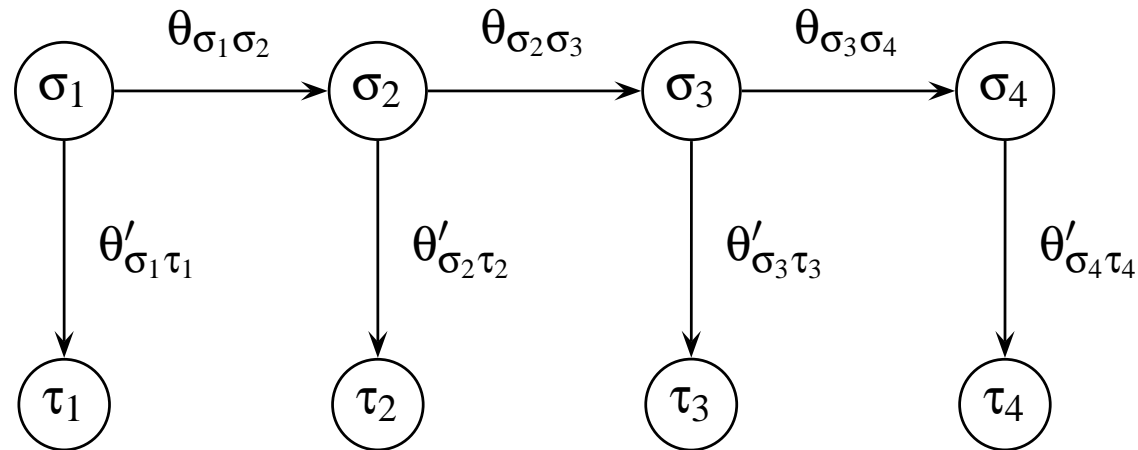
## Experiments (The sum of a simplex and its dual) on Pentium 1.7 MHz

d	cpu (sec)	cpu_init	cpu_lp	#vert	#edges	#lp	lp_size
10	4.21	0.79	1.79	110	990	704	20x11
20	91.91	16.39	51.74	420	7980	3004	40x21
30	601.61	108.28	371.06	930	26970	6904	60x31

## The Hardest Problem Solved

A Minkowski sum of 9 polytopes in  $\mathbb{R}^{27}$ , each of which has only 6 vertices. It took about a month to generate all 2,372,583 vertices.

## Hidden Markov Model (HMM)



The probability of a certain observation is a polynomial in the parameters:

$$P_{\tau_1\tau_2\tau_3\tau_4} = \frac{1}{2} \sum_{\sigma_1} \sum_{\sigma_2} \sum_{\sigma_3} \sum_{\sigma_4} \theta'_{\sigma_1\tau_1} \theta_{\sigma_1\sigma_2} \theta'_{\sigma_2\tau_2} \theta_{\sigma_2\sigma_3} \theta'_{\sigma_3\tau_3} \theta_{\sigma_3\sigma_4} \theta'_{\sigma_4\tau_4}$$

# Computation of the Newton Polytope

The Newton polytope  $N(f)$  of the binary HMM can be computed by summing  $2^n$  polytopes in  $\mathbb{R}^8$  each of which are 4-dimensional. The Minkowski sum ( $n \geq 3$ ) is a 5-dimensional polytope.

$n$	$[f_0, f_1, \dots, f_{d-1}]$	# LP	LP size	time (sec)	lp time (sec)
2	[38, 84, 62, 16]	95	$(12 \times 9)$	0.40	0.22
3	[398, 1136, 1150, 478, 68]	3172	$(48 \times 9)$	28.50	12.02
4	[1570, 4404, 4326, 1712, 222]	21745	$(136 \times 9)$	460.12	156.24
5	[5266, 15014, 14972, 5968, 746]	135480	$(328 \times 9)$	$\sim 10$ hrs	
6	[17534, ?, ?, ?, 3507]			$\sim 3$ days	
7	[55230, ?, ?, ?, ?]			$\sim 7$ days	

## An Implementation of a Gröbner Fan Algorithm

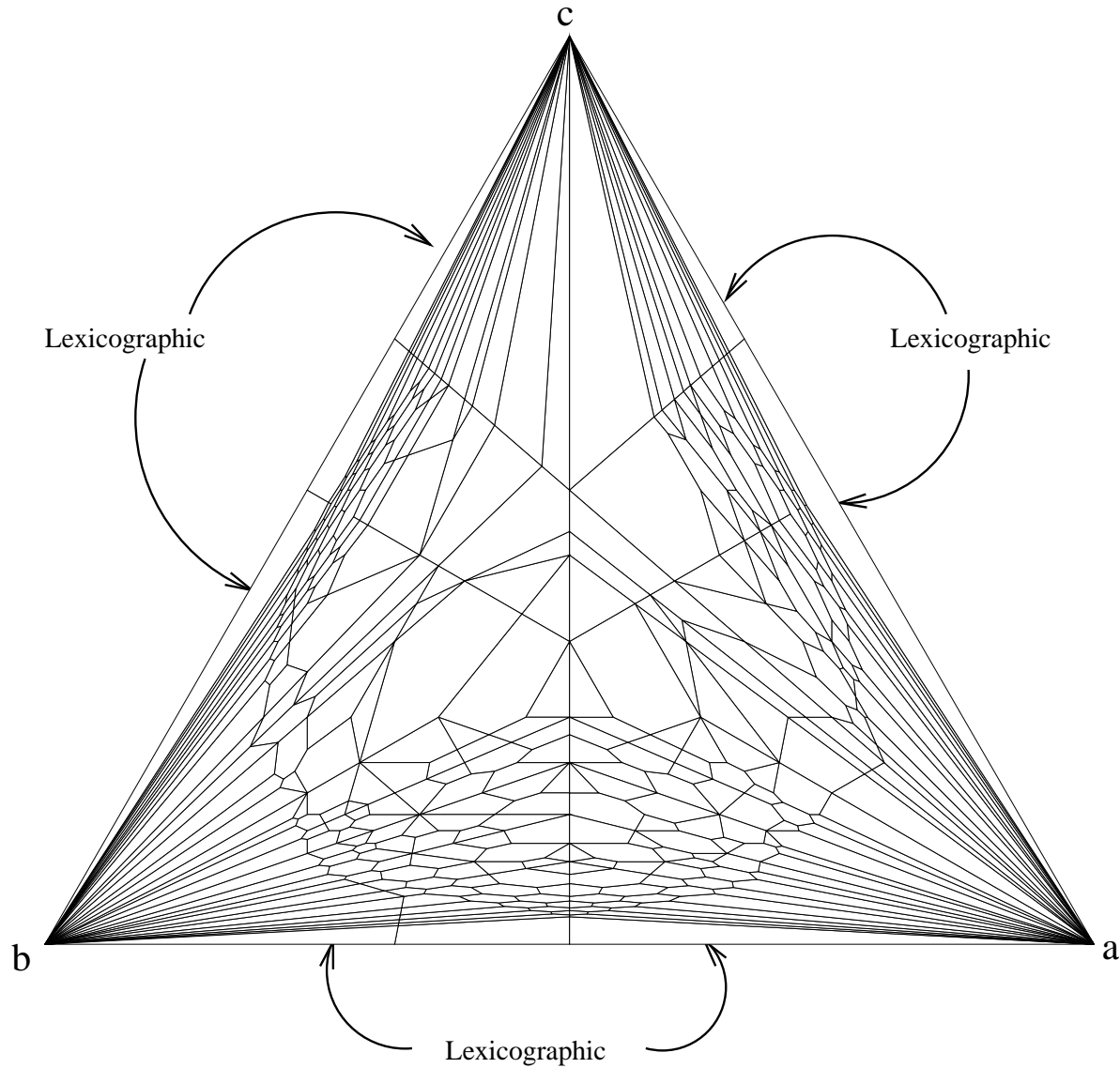
(by Jensen 2004)

- A faithful implementation (freely available [7]) of the reverse search algorithm due to KF-Jensen-Thomas (2005), an extended version of Sturmfels' Algorithm (1995).
- It is written in C++, using both GMP and the rational arithmetic LP code in cddlib by KF.

### A Computed Example (Example 3.9 in Sturmfels' book)

The ideal  $I = \langle a^5 + b^3 + c^2 - 1, a^2 + b^2 + c - 1, a^6 + b^5 + c^3 - 1 \rangle$  has exactly 360 Gröbner bases. It took 105 seconds on a laptop (1.8 GHz AMD XP-M). One third of the time is spent in the LP solving.

**The Gröbner Fan of  $\langle a^5 + b^3 + c^2 - 1, a^2 + b^2 + c - 1, a^6 + b^5 + c^3 - 1 \rangle$**



## Concluding Remarks

- Many problems in polyhedral computation require a large number of small-scale LPs to be solved. Even **one failure** out of one million LPs can result in a useless result.
- Problems arising from our applications are often highly **degenerate** and numerically sensitive.
- Thus, using an LP solver that is rock solid is essential. For the moment, we use an exact rational arithmetic solver of the C-library [cddlib-094b](#). (Both Cplex and soplex are not solid enough.)
- Reverse search algorithms work efficiently in massively parallel environment. Extending current implementations to run in parallel is highly desired.
- There are many **open problems** in Polyhedral Computation. The PVP problem, exploiting symmetries, Minkowski H-additions, etc.

# References

- [1] K. Fukuda. From the zonotope construction to the Minkowski addition of convex polytopes. Journal of Symbolic Computation, 38(4):1261–1272, 2004.
- [2] K. Fukuda. Polyhedral computation FAQ, 2004. Both html and ps versions available from <http://www.ifor.math.ethz.ch/~fukuda/fukuda.html>.
- [3] K. Fukuda. cdd, cddplus and cddlib homepage. Technical report, Swiss Federal Institute of Technology, Lausanne and Zurich, 2005. [http://www.ifor.math.ethz.ch/~fukuda/cdd\\_home/index.html](http://www.ifor.math.ethz.ch/~fukuda/cdd_home/index.html).
- [4] K. Fukuda, A. Jensen, and R. Thomas. Computing Gröbner fans. preprint, 2005. <http://www.arxiv.org/abs/math.AC/0509544>.
- [5] K. Fukuda and C. Weibel. On  $f$ -vectors of Minkowski additions of convex polytopes. Technical report, 2005. <http://www.arxiv.org/abs/math.CO/0510470>.
- [6] A.N. Jensen. Gfan version 0.2: A User's Manual. Department of Mathematical Sciences, University of Aarhus and Institute for Operations Research, ETH Zurich, 2005. available from <http://home.imf.au.dk/ajensen/software/gfan/gfan.html>.

- [7] C. Weibel. Minksum version 1.3. Mathematics Institute, EPF Lausanne, 2005. available from <http://roso.epfl.ch/cw/poly/public.php>.