Practical polyhedral computations under symmetry

Mathieu Dutour Sikirić

January 16, 2010

I. Basic definitions

Polytopes, definition

- A polytope $P \subset \mathbb{R}^n$ is defined alternatively as:
  - The convex hull of a finite number of points $v^1, \ldots, v^m$:

$$P = \{v \in \mathbb{R}^n \mid v = \sum_i \lambda_i v^i \text{ with } \lambda_i \geq 0 \text{ and } \sum \lambda_i = 1\}$$

  - The following set of solutions:

$$P = \{x \in \mathbb{R}^n \mid f^i(x) \geq b_i \text{ with } f_i \text{ linear}\}$$

    with the condition that P is bounded.
- The cube is defined alternatively as
  - The convex hull of the 2^n vertices

$$\{(x_1, \ldots, x_n) \text{ with } x_i = \pm 1\}$$

  - The set of points $x \in \mathbb{R}^n$ satisfying to

$$x_i \leq 1 \text{ and } x_i \geq -1$$

Facets and vertices

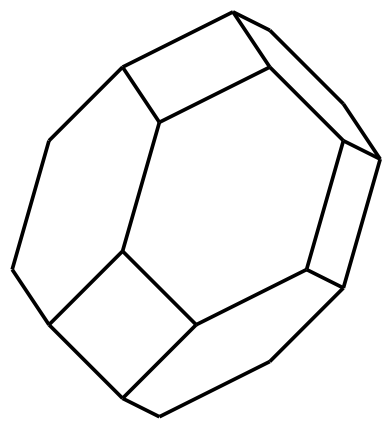
- A vertex of a polytope P is a point $v \in P$, which cannot be expressed as $v = \lambda v^1 + (1-\lambda)v^2$ with $0 < \lambda < 1$ and $v^i \in P$.
- A polytope is the convex hull of its vertices and this is the minimal set defining it.
- A facet of a polytope is an inequality $f(x) - b \geq 0$, which cannot be expressed as $f(x) - b = \lambda(f^1(x) - b_1) + (1-\lambda)(f^2(x) - b_2)$ with $f^i(x) - b_i \geq 0$ on P.
- A polytope is defined by its facet inequalities. and this is the minimal set of linear inequalities defining it.
- The dual-description problem is the problem of passing from one description to another.

Faces

- Given an inequality $f(x) \geq b$, which is valid on P, the face defined by $f(x) \geq b$ is

$$x \in P \text{ such that } f(x) = b$$

  and its dimension is the dimension of the smallest affine plane containing it.
- The dimension of faces of a n dimensional polytope P varies from 0 to $n-1$. A face of dimension 0 is a vertex, a face of dimension $n-1$ is a facet.
- Faces are defined by the set of vertices contained in them.
- The inclusion relation between faces defines a lattice.



Homogeneous coordinates and duality

- Linear functions are expressed in terms of scalar product.

$$f(x) = a_1 x_1 + \cdots + a_n x_n = \langle a, x \rangle$$

- A polyhedral cone is a cone defined by linear inequalities $f(x) \geq 0$. The vertices correspond to extreme ray.
- Formulas are easier for the polyhedral cones, all programs are designed for polyhedral cones and not for polytopes.
- But we can reduce polytope to polyhedral cones:
  - If $v \in \mathbb{R}^n$ is a vertex then we map it to a vector $v' = (1, v) \in \mathbb{R}^{n+1}$.
  - If $f(x) = \langle a, x \rangle \geq b$, we map it to a vector $a' = (-b, a)$.
  - The inequality $f(v) \geq b$ is then rewritten as $\langle v', a' \rangle \geq 0$.
- The two problems:
  1. given the vertices of P, find the facets,
  2. given the facets of P, find the vertices,

  are now expressed exactly identically:

  Find extreme rays of the cone $\langle a_i, x \rangle \geq 0$ with $1 \leq i \leq m$

II. Linear programming

Linear programming

- If $f(x)$, $f_i(x)$ are affine functions on $\mathbb{R}^n$, $b_i \in \mathbb{R}$, then the linear programming problem is:

$$\begin{aligned} \textbf{maximize} \quad & f(x) \\ \textbf{subject to} \quad & f_i(x) \geq b_i \end{aligned}$$

- Two main class of methods exist:
  - **The simplex method**: It goes from one vertex of the solution to another adjacent vertex until an optimal vertex is obained. Not polynomial in general, very good in practice.
  - **Interior point methods**: It takes an interior point and converges to a better and better vertex.
    With the primal dual method the method returns an interval, which can be made as small as possible.
    Polynomial in theory, relatively bad for us.

  Generally we use simplex methods because they use exact arithmetic and for the kind of computation is usually not the limiting factor.

Computations related to linear programming

- Take $P = conv(v_1, \ldots, v_M)$ a polytope.
  - Testing if an element v belongs to the interior of P is lin.prog.
  - Testing if an element v belongs to P is lin.prog.
  - Determining the vertices amongst the v_i is lin.prog (M times).
  - Determining the adjacency $v_i - v_j$ amongst the v_i is lin.prog ($M(M-1)/2$ times).
- Take $P = \{x \in \mathbb{R}^n \ : \ f_i(x) \geq b_i \text{ for } 1 \leq i \leq N\}$.
  - Testing $P = \emptyset$ is lin.prog.
  - Computing the dimension of P is lin.prog.
  - Determining facet defining inequalities is lin.prog.
  - Finding one vertex is lin.prog.
- In principle we can obtain all the facets from such linear combinations but we will see faster methods.
- Linear programming is ok, when not used too much. If that is the case, then it is necessary to compute the dual description.

III. The dual description problem

Computing dual description

- The dual description problem is important to many many computations:
  - It allows to test membership questions easily.
  - It allows to get the full face-set if needed.
- In high dimension the problem becomes difficult:
  - The number of vertices, facets grows very fast.
  - Even if the number is small, it can be difficult to compute.
- Some known programs exist (`cdd`, `lrs`, `ppl`, `pd`, `porta`, `qhull`, etc.), their efficiency varies widely and sometimes they take too much time.
- In many cases the polytope considered have a "big" symmetry group and the orbits of facets is the really needed information.
- We will expose some techniques for dealing with this problem.

Limitations of hope

- If the quotient $\frac{\#facets}{|G|}$ is really too large then the problem becomes impossible.
- Combinatorial explosion is the driving phenomenon. Using symmetry has only limited efficiency.

polytope	dim.	$\|V\|$	$\|G\|$	# orbits fac.	# facets
CUT_4	6	8	1152	1	16
CUT_5	10	16	1920	2	56
CUT_6	15	32	23040	3	368
CUT_7	21	64	322560	11	116764
CUT_8	28	128	5160960	147?	
CUT_9	36	256	185794560	$\geq 1.10^6$	

CUT_n is a polytope arising in combinatorial optimization.

- In practice, the method explained here allow to compute the required list if its size is reasonable.

Program comparisons

We consider a polytope defined by a set $\mathcal{LF}$ of inequalities for which we want its vertex set $\mathcal{LV}$.

- lrs: it iterates over all admissible basis in the simplex algorithm of linear programming
  - It is a tree search, no memory limitation.
  - Some repetition can occur in the output.
  - Ideal if the polytope has a lot of vertices.
- cdd: it adds inequalities one after the other and maintain the double description throughout the computation
  - All vertices and facets are stored memory limitation.
  - Good performance if the polytope has degenerate vertices.
- pd: We have a partial list of vertices, we compute the facets with lrs. If it does not coincide with $\mathcal{LF}$ then we can generate a missed vertex by linear programming.
  - It is a recommended method if there is less vertices than facets.
  - Bad performance for general polytopes.
- So, in general, choosing the right method is really difficult.

V. The adjacency decomposition method

The adjacency decomposition method

`Input`: The vertex-set of a polytope P and a group G acting on P.
`Output`: $\mathcal{O}$, the orbits of facets of P.

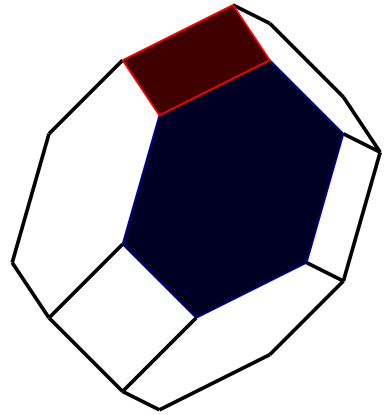
- Compute some initial facet F (by linear programming) and insert the corresponding orbit into $\mathcal{O}$ as `undone`.
- For every `undone` orbit O of facet:
  - Take a representative F of O.
  - Find the ridges contained in F, i.e. the facets of the facet F (this is a dual description computation).
  - For every ridge R, find the corresponding adjacent facet F' such that $R = F \cap F'$.
  - For every adjacent facet found test if the corresponding orbit is already present in $\mathcal{O}$. If no insert it as `undone`.
  - Mark the orbit O as `done`.
- Terminate when all orbits are `done`.

Reinvented many times (D. Jaquet 1993, T. Christof and G. Reinelt 1996, A. Deza et al. 2001).

General feature of the algorithm

It is a graph traversal algorithm:

- The algorithm starts by computing the orbits of lowest incidence, which are the one for which the dual description is easiest to be done.
- Sometimes it seems that no end is in sight, we get a lower bound on the number of orbits.
- At the end, only the orbits of highest incidence remains.
- In most cases, the orbits of highest incidence do not yield new orbits but in a few cases, this happens



Balinski theorem

The skeleton of a polytope is the graph formed by its facets with two vertices adjacent if and only if the facets are adjacent.

- Balinski theorem The skeleton of a n-dimensional polytope is n-connected, i.e. the removal of any set of $n-1$ vertices leaves it connected.
- So, if the number of facets in remaining orbits is at most $n-1$, then we know that no more orbits is to be discovered.

Scope of application:

- the criterion is usually not applicable to the polytopes of combinatorial optimization, i.e. the orbits of facets of such polytopes are usually relatively big.
- For the polytopes arising in geometry of numbers, it is sometimes applicable.
- Very cheap to test, huge benefits if applicable.

The recursive adjacency method

In all cases considered so far, the orbits of maximum incidence also have the highest symmetry and are the most difficult to compute.

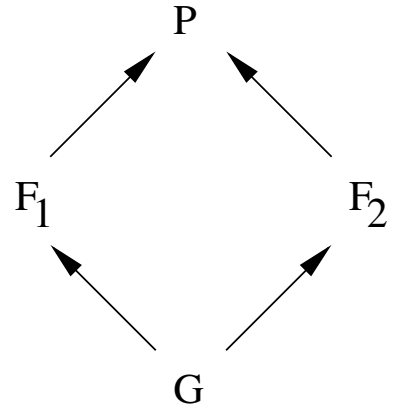
- The computation of adjacent facets is a dual-description computation.
- So, the idea is to apply the Adjacency Decomposition method to those orbits as well.
- Based on informations on the symmetry group and on the incidence, we decide if we should respawn the adjacency method at another level.

Issues:

- The number of cases to consider can grow dramatically.
- If one takes the stabilizer of a face, then the size of the groups involved may be quite small to be efficient.

Banking methods

- When one applies the Recursive Adjacency decomposition method, one needs to compute the dual description of faces.
- F_1 and F_2 are two facets of P to which we apply the Adjacency Decomposition Method.
  G is a common facet of F_1 and F_2.
  The dual description of G is computed twice:



- The idea is to store the dual description of faces in a bank and when a dual description is needed to see if it has been already done.

IV. Symmetry questions

Permutation groups

- Polytopes of interest have usually less than 1000 vertices $v_1, \ldots, v_N$, their symmetry group can be represented as a permutation of their vertex-set.
- The first benefit is that permutation group algorithms have been well studied for a long time and have good implementation in GAP.
  - A. Seress, *Permutation group algorithms*, Cambridge University Press, 2003.
  - D.F. Holt, B. Eick and E.A. O'Brien, *Handbook of computational group theory*, Chapman & Hall/CRC, 2005.
- The second benefit is that a facet of a polytope thus corresponds to a subset of $\{1, \ldots, N\}$ and that permutation group acting on sets have a very good implementation in GAP.
- In some extreme cases (# vertices > 100000) permutation groups might not work as quietly and other methods have to be used.

Symmetry questions

Usually, most of the computational time is spent in symmetry computations.

- We always need two operations:
  - Isomorphism tests between two objects.
  - Computation of the stabilizer or automorphism group of an object.
- There are three different contexts:
  (1) Identifying orbits when the full orbit has been generated.
  (2) Given a polytope P and a group G acting on P, test if two faces are equivalent under G.
  (3) Test if two polytopes are isomorphic.

(1) Full orbit

- Eventually, the Recursive Adjacency Decomposition Method will call `lrs`, `cdd`, etc for generating the full dual-description.
- Hence, we need to split the output into orbits.
- The idea is then to code those orbits by 0/1-vectors and to identify the full orbits, use `C++` and the `STL` for identifying them.
- This is memory-limited and sometimes we cannot identify the orbits correctly. Then we do another respawn of the adjacency method.

(2) In the Adjacency decomposition iteration

We have a fixed group G of a polytope P and we want to test if two faces F_1 and F_2 are equivalent under G.

- We represent G as permutation group on the set of vertices $(v_i)_{1 \leq i \leq N}$ of P and the faces by their incidence, i.e. subsets of $\{1, \ldots, N\}$.
- Then, we use two following functions in GAP
  - `Stabilizer(G, S1, OnSets);`
  - `RepresentativeAction(G, S1, S2, OnSets);`

  The important fact is that the action OnSets is extremely efficient and uses backtrack search, i.e. in practice we never build the full orbit.
- The main reason why our program is working is because GAP has efficient implementation of those functions.

(3) Symmetry groups of polytopes

- Suppose P is a polyhedral cone generated by vectors $(v_1)_{1\leq i\leq N}$ in $\mathbb{R}^n$. There are three possible groups
  - Combinatorial symmetry group $Comb(P)$: this is the group of transformations $\sigma \in \mathrm{Sym}(N)$ preserving the set of faces of P globally.
  - Projective symmetry group $Proj(P)$: this is the group of transformations $\sigma \in \mathrm{Sym}(N)$ such that there exist $\alpha_\sigma > 0$, $A \in \mathrm{GL}_n(\mathbb{R})$ with $Av_i = \alpha_\sigma(i)v_{\sigma(i)}$.
  - Linear symmetry group $Lin(P)$: this is the group of transformations $\sigma \in \mathrm{Sym}(N)$ such that there exist $A \in \mathrm{GL}_n(\mathbb{R})$ with $Av_i = v_{\sigma(i)}$.

  We have $Lin(P) \subset Proj(P) \subset Comb(P)$
- It can be proved that we need "only" the facets to compute $Comb(P)$. Since this is the objective itself, we have to be content with $Proj(P)$ and $Lin(P)$.

Computing $Lin(P)$

- Define the form

$$Q = \sum_{i=1}^{N} {}^t v_i v_i$$

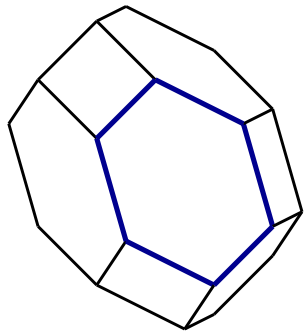
- Define the edge colored graph on N vertices with edge color

$$c_{ij} = v_i Q^{-1\,t} v_j$$

- The automorphism group of the edge colored graph corresponds to the automorphism group of the vector family.
- The automorphism group of the edge colored graph is computed with `nauty` and a reduction to a vertex colored graph. If G has n vertices and k colors, then we have the following reductions:
  - Line graph: $\frac{n(n-1)}{2}$ vertices.
  - Every color is a graph: nk vertices.
  - Every bit of a color is a graph: $n\log(k)$ vertices.
  - Another construction: $n\sqrt{\log(k)}$ vertices.

Going from high to low symmetries

- The symmetry group of the face might be larger than its stabilizer under the bigger group.



  - The stabilizer of the face has order 6
  - The symmetry group of the face has order 12.

- Suppose that we have a set of orbits for the big symmetry group G

$$\mathcal{F} = x_1 G \cup \cdots \cup x_n G$$

  we want to represent $\mathcal{F}$ as list of orbits for a subgroup H of G.

- For every x_i do a double coset decomposition

$$G = G_{x_i} g_1 H \cup \ldots \cup G_{x_i} g_p H$$

  with G_{x_i} the stabilizer of x_i in G.

- So, $x_i G = \cup_j x_i g_j H$

VI. Case Studies

Perfect domain Dom(E_8)

- Context: The Voronoi algorithm for computing perfect forms in dimension n needs to find the facets of their perfect domains.
- The perfect domain Dom(E_8) has 120 extreme rays and is of dimension 36 symmetry group has size 348364800.
- There are 2507556693758 4 facets in 83092 orbits.
- 4 orbits required a secondary application of the ADM.
- The orbit made of facets of incidence 75 have a stabilizer of size 23040 but a symmetry group of size 737280, therefore allowing us to finish the computation.
- Total running time with ons and offs was 15 months.

Contact polytope of O_{23}

- Context: The determination of overlattice of O_{23} of minimum 3 requires the computation of vertices of the contact polytope of O_{23}.
- The polytope $Contact(O_{23})$ has 4600 facets, dimension 23 and the symmetry group $\mathbb{Z}_2 \times Co_1$.
- There are 1561558479368414 vertices in 269 orbits.
  - One vertex correspond to a 22-dimensionl simplicial polytope of 44 vertices with a group transitive on simplices.
  - HS, M_{22}, M_{23} appear as stabilizer of vertices.
  - One orbit is incident to 275 facets and has group McL.
  - One orbit of simplices is incident only to the above orbit.
- Main computational difficulty is in checking if two vertices are equivalent.
- Total running time is two days.

Delaunay polytopes of Coxeter lattices

- Coxeter lattices A_n^r are some lattices with a symmetry group $\mathbb{Z}_2 \times \mathrm{Sym}(n+1)$
- The key step of the computation of Delaunay polytopes is computing facets of Delone polytopes. The problem is that some Delaunay polytopes have more than 300000 vertices.
- Every face of a Delaunay is encoded by its barycenter, thus we do not need permutation representations on huge number of vertices.
- The heuristic is to respawn the ADM whenever the number of vertices is greater than 70. This makes sometimes 16 levels of recursion.

lattice	# orbits		
A_{13}^2	10	A_{14}^3	17
A_{15}^2	10	A_{17}^3	26
A_{17}^2	15	A_{20}^3	40
A_{19}^2	15	A_{23}^3	55
A_{21}^2	21	A_{26}^3	75

VII. Other methods using symmetry for dual description

Techniques apparented to Adjacency Decomposition

- Computing perfect forms in dimension n: This can be seen as computing vertices of polytope defined by following inequalities:

$$\{A \in S^n \text{ such that } xAx^T \geq 1 \text{ for all } x \in \mathbb{Z}^n - \{0\}\}$$

- Computing Delaunay polytopes of a lattice $L \subset \mathbb{R}^n$. This can be seen as computing facets of the polytope defined by following vertex-set

$$\{(x, \|x\|^2) \text{ for all } x \in L\}$$

- All kinds of space decompositions by polyhedra (T-type, L-types, etc.)

The incidence technique

The incidence technique is the logical competitor of the Adjacency Decomposition Method.

- Suppose that the vertex set $\mathcal{E}$ of P is partitionned into orbits $\{O_1, \ldots, O_p\}$ of representative v_i.
- For every $1 \leq i \leq p$, consider the space

$$P_i^* = \{f \in (\mathbb{R}^n)^* \mid f(v) \geq 0 \text{ for } v \in \mathcal{E} \text{ and } f(v_i) = 0\}$$

  Every facet of P is equivalent to a facet in P_i^* for some i.
- The description of P_i^* may be redundant, so elimination of redundant facets by linear programming is necessary.
- The incidence method admits extensions to edges of P, 2-dimensional faces of P, etc.

This is invented several times (V. Grishukhin 1992, T. Christof and G. Reinelt 1996, A. Deza et al. 2004).

The cascade algorithm

The Cascade algorithm (a reincarnation of Fourier-Motzkin) has been introduced by D. Jaquet 1992:

- If P is a polytope of dimension n with m vertices, then it is the projection of a simplex of dimension $m-1$.
- If P' is a polytope in $\mathbb{R}^q$, f a projection on an hyperplane of $\mathbb{R}^q$, then the facets of $f(P')$ are:
  - Projections of facets of P'.
  - Projections of intersection of adjacent facets of P'.
- We can compute the orbits of facets of the projection $f(P')$ from the orbits of facets of the polytope P'.
- This yields an algorithm for enumerating facets of P:
  - Start with the simplex of dimension $m-1$.
  - Project $m-1-n$ times to get the facets of P.

The problem is that the intermediate polytopes have a much smaller symmetry group than the original polytope.

Orbit polytope

- Suppose G is a group acting on $\mathbb{R}^n$, $v \in \mathbb{R}^n$. We want to compute the facets of the orbit polytope $\mathrm{conv}(G.v)$.
- We problem is that $G.v$ the vertex-set might be too large to store in memory and the facets be very large too.
- The technique is store the set S of vertices adjacent to v, say, $S = \{v_1, \ldots, v_m\} = v.\{g_1, \ldots, g_m\}$ (Poincaré Polyhedron theorem).
- Use an iteration
  - Determine an initial set S with (g_i) generating G.
  - By the group action, we know the vertices adjacent to S.
  - We check if those vertices are adjacent to v.
    - If yes, we update the set S.
    - If no, we return the set S as the reply.
- It works for the group M_{24} for $v = (0^{19}, 1, 2, 3, 4, 5)$, $|G.v| = 5100480$.
- But this is a very specific example: $M_{24}.v = \mathrm{Sym}(24).v$, whose face-lattice is given by the Wythoff construction.

VIII. Application of dual descriptions

Face-lattice under symmetry

The face-lattice of a polytope is usually "fat":

- The number of faces of intermediate dimension is much larger than the number of vertices and facets.

There is an efficient algorithm for enumerating the faces under symmetry:

- We first compute the facets of the polytope.
- We represent faces by the list of incident vertices and the action OnSets.
- For every face F of dimension k, we use the facets to find the faces of dimension $k+1$ to which F belongs.
- We then reduce by isomorphism.

If one wants only the k-faces for small k, then the facets are not necessary a priori and linear programming suffices.

Flag system under symmetry

- The number of flags is much larger than the number of faces.
- But there is an efficient algorithm for enumerating orbits of flags under symmetry.
- The idea is to extend flags $(F_0, \ldots, F_k)$ to flags $(F_0, \ldots, F_{k+1})$ with $\dim F_i = i$.
- The only trick is the isomorphism test:
  - Take two flags $f = (F_0, \ldots, F_k)$ and $f' = (F'_0, \ldots, F'_k)$
  - Check isomorphism of F_0 and F'_0 under G with OnSets. If not-isomorphic leave.
  - If $F'_0 = F_0.g$ then replace f by $f.g$.
  - Replace G by the stabilizer of F'_0.
  - Consider faces of dimension $1, \ldots, n$.

Group homology

- If G is a group, X a classifying space, then $H_i(G) = H_i(X/G)$.
- A classifying space X is one such that G acts fixed-point-free on it.
- If G is a finite matrix group, then $\mathrm{conv}(G.v)$ provides an "approximation" of it, i.e. stabilizer of faces are small.
- If i is small, the fact that the action is not fixed-point-free can be taken care of by using a technique named "C.T.C. Wall Lemma".
- This gives a method for computing $H_i(G)$ for i small.

IX. Inertia moment

Inertia matrix of a polytope

- If $P \in \mathbb{R}^n$ is a polytope then we want to compute the integrals of volume, barycenter and inertia moment

$$\int_P dx, \quad \int_P x_i dx \quad \text{and} \quad \int_P x_i x_j dx$$

- This integral is rewritten as a symmetric $(n+1) \times (n+1)$ matrix integral:

$$I_{0,1,2}(P) = \int_P (1, x)(1, x)^T dx$$

- If one is satisfied with approximate results, then Monte Carlo methods are to be preferred. They are much faster and fairly accurate.

Decomposition method

- All known methods for computing integrals over a polytope P rely on decomposing it into an union (signed or not) of simplices.
  - B. Büeler B., A. Enge and K. Fukuda, *Exact Volume Computation for Polytopes: a Practical Study*, Polytopes—combinatorics and computation (Oberwolfach, 1997), 131–154, DMV Sem., **29**, Birkhäuser, Basel, 2000.
- Two methods are used by us:
  - lrs can return a simplicial decomposition if one computes the facets from the vertices.
  - If one takes a random quadratic form and computes a Delaunay decomposition for it then "most" Delaunays are simplices. The remaining can be decomposed by further application of the method.

How to use symmetries?

- We need decompositions of the space, which are invariant space decompositions into simplices.
- One way to get such a decomposition is to compute the orbits of flags $F_0 \subset F_1 \subset \cdots \subset F_n$ of P.
- For every such flag we associate the simplex

$$(Iso(F_0), Iso(F_1), \ldots, Iso(F_n), Iso(P))$$

  The decomposition is then invariant under $Lin(P)$.
- The isobarycenter $Iso(F)$ is the isobarycenter of the vertex-set of F, not of F itself as a polytope.
- The problem is that polytopes have generally a lot of orbits of flags even "very symmetric ones".

Lassere decomposition method

Suppose we have a n-dimensional polytope P and a group G acting on it by isometries.

- Compute the orbits of facets $O_1, \ldots, O_s$ of representative $F_1, \ldots, F_s$
- Compute the isobarycenter $Iso(P)$ of the vertex-set of P.
- One has the formulas.

$$\begin{aligned} vol(\text{conv}(F_i, Iso(P))) &= \tfrac{1}{n} vol(F_i) \times d(F_i, Iso(P)) \\ vol(P) &= \sum_{i=1}^{s} |O_i| \,\text{vol}(\text{conv}(F_i, Iso(P))) \end{aligned}$$

- More generally, we can express the integral $I_{0,1,2}(\text{conv}(F_i, Iso(P)))$ in terms of $I_{0,1,2}(F_i)$ and $Iso(P)$.
- The integral $I_{0,1,2}(P)$ expands as

$$\sum_{i=1}^{s} |O_i| \left(\frac{1}{|G|} \sum_{g \in G} g I_{0,1,2}(\text{conv}(F_i, Iso(P))) g^T \right)$$

Averaging operation

- If G is a group generated by $g_1, \ldots, g_s$ acting affinely on $\mathbb{R}^n$, $v \in \mathbb{R}^n$, we want to compute the barycenter of the orbit Gx:

$$Iso(v) = \frac{1}{|G|} \sum_{g \in G} g.v$$

  but we don't want to compute the orbit itself.
- Denote by $Aff(G, v)$ the smallest affine subspace of $\mathbb{R}^n$ invariant under G containing the point v.
  The method is simply to add points to a basis $\mathcal{B}$ of $Aff(G, v)$ inductively until the obtained subspace is invariant.
- Take an affine basis $v_1, \ldots, v_m$ of $Aff(G, v)$ and write $Iso(v)$ as

$$Iso(v) = \sum_{i=1}^{m} \alpha_i v_i \quad \text{with} \quad \sum_{i=1}^{m} \alpha_i = 1,$$

  which is then the unique solution of the equations $g_i(Iso(v)) = Iso(v)$

The recursive decomposition method for $I_2(DV(L))$

- We use Lassere's method recursively until the number of vertices is low enough.
- Faces of $DV(L)$ are encoded by their dual Delaunay and vertices generated only when needed.
- We have a banking system to keep computed integral.
- Some results:

Lattice L	$Q(L)$
Λ_9	$\frac{151301}{2099520} \approx 0.07206$
Λ_9^*	$\frac{1371514291}{19110297600} \approx 0.07176$
A_9^2	$\frac{2120743}{\sqrt[9]{5.2^8 13271040}} \approx 0.072166$
A_9^5	$\frac{8651427563}{\sqrt[9]{2.5^8 26578125000}} \approx 0.072079$
D_{10}^+	$\frac{4568341}{64512000} \approx 0.07081$
D_{12}^+	$\frac{29183629}{412776000} \approx 0.070700$
K_{12}	$\frac{797361941}{\sqrt{36567561000}} \approx 0.070095$

Availability

The software polyhedral is available from my web page

`http://www.liga.ens.fr/~dutour/polyhedral/`

Other features:

- The system works, as a database, by saving on disk:
  - This works by guaranteeing atomicity of operations.
  - This is useful in case of power failure, no loss of work.
  - It is also useful when we change the heuristics of the respawning. All computations simply update the database.
- Written in `GAP`, `perl`, `C++` using many people's other programs (`nauty`, `cdd`, `lrs`).
- Examples, but no manual yet.

THANK

YOU