polyhedral: A GAP package for polytope and lattice computations using symmetries

Mathieu Dutour Sikirić

August 28, 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.
II. The dual description problem
The dual description problem is important to many computations:

- It allows to test membership questions easily.
- It allows to get the full face-set if needed.
- It allows to do enumeration of mathematical objects.

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 $\frac{\text{# facets}}{|G|}$ 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.
III. The adjacency decomposition method
The adjacency decomposition method

**Input:** The vertex-set of a polytope $P$ and a group $G$ acting on $P$.

**Output:** $O$, the orbits of facets of $P$.

- Compute some initial facet $F$ (by linear programming) and insert the corresponding orbit into $O$ 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 $O$. If no insert it as **undone**.
  - Mark the orbit $O$ as **done**.
- Terminate when all orbits are **done**.

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.

Other connectness theorems have proved very useful. Proving new connectness theorem could be key to new enumeration results.
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.
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.

- 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.
Eventually, the Recursive Adjacency Decomposition Method will call \texttt{lrs}, \texttt{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 \texttt{C++} and the \texttt{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.
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:
  - $\text{Stabilizer}(G, S_1, \text{OnSets})$;
  - $\text{RepresentativeAction}(G, S_1, S_2, \text{OnSets})$;

The important fact is that the action $\text{OnSets}$ 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.
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 $\sigma \in \text{Sym}(N)$ preserving the set of faces of $P$ globally.
- Projective symmetry group $Proj(P)$: this is the group of transformations $\sigma \in \text{Sym}(N)$ such that there exist $\alpha_\sigma > 0$, $A \in \text{GL}_n(\mathbb{R})$ with $Av_i = \alpha_\sigma(i)v_\sigma(i)$.
- Linear symmetry group $Lin(P)$: this is the group of transformations $\sigma \in \text{Sym}(N)$ such that there exist $A \in \text{GL}_n(\mathbb{R})$ 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 $Lin(P)$. 

(3) Symmetry groups of polytopes
Computing $\text{Lin}(P)$

- 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 \texttt{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
  \[ G = G_{x_i} g_1 H \cup \cdots \cup G_{x_i} g_p H \]
  with $G_{x_i}$ the stabilizer of $x_i$ in $G$.

- So, $x_i G = \bigcup_j x_i g_j H$
V. Case Studies
Perfect domain $\text{Dom}(E_8)$

- **Context:** The Voronoi algorithm for computing perfect forms in dimension $n$ needs to find the facets of their perfect domains.
- The perfect domain $\text{Dom}(E_8)$ has 120 extreme rays and is of dimension 36 symmetry group has size 348364800.
- There are 25075566937584 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.
Contact polytope of Leech lattice

- The Leech lattice is a famous lattice with many extremely interesting properties (best packing, best kissing number, conjectured best covering, etc.)
- The polytope $\text{Contact}(\text{Leech})$ has 196560 facets, dimension 24 and the symmetry group $\text{Co}_0$.
- There are 1, 197, 362, 269, 604, 214, 277, 200 many facets in 232 orbits.
  - One vertex corresponds to a vector of distance 6 to the origin.
  - $\text{Co}_1, \ M_{23}, \ M_{24}$ appear as stabilizer of vertices.
- Main computational difficulty is in checking if two vertices are equivalent.
  For some vertices $v$ and $v'$ we had
    - $\text{Stabilizer}($\text{GRPleech}, \ eSet1, \ \text{OnSets});;$
    - $\text{Stabilizer}($\text{GRPleech}, \ eSet2, \ \text{OnSets});;$
  computed easily (and found to be $M_{24}$ and $M_{23}$). But
    - $\text{RepresentativeAction}($\text{GRPleech}, \ eSet1, \ eSet2, \ \text{OnSets});;$
  lasting forever.
Non rational polytopes

- For some polytopes like Dodecahedron, we need to work with non-rational coordinates.
- Right now, we have functionalities for working with fields $\mathbb{Q}(\sqrt{N})$.
- Any totally real field could be implemented.
- We had to modify cdd for that.
- We could not do it for lrs since its infrastructure requires greatest common divisors operations (which do not exist if the class number is greater than 1) and signs for unicity $((\sqrt{2} - 1)^j$ is always invertible thus making it problematic).
- Context of such application:
  - Computing convex hull of $H_4$ which is an open problem.
  - Computing $T$-spaces coming from number theory.
VI. Other dual-description computations
Delaunay computations

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

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

- Applications of this computation:
  - Computing covering density
  - Computing Voronoi polytope
  - Computing Delaney symbol (flag system combinatorial description)
  - Computing quantization constant (in information theory)

- Extensions of this program have been done to
  - Periodic structure in crystallography
  - Non rational Gram matrix
Polyhedral tessellations

- By polyhedral tessellation we mean a face-to-face tiling of a linear space by a set of polytopes or polyhedral cone. The group acting on it is supposed to be infinite but the number of orbit up to equivalence is finite.

- Examples:
  - Computing perfect forms in dimension $n$. This is related to the theory of lattice sphere packing.
  - Computing tessellations by $L$-types (decomposition of the space of quadratic form by
    - Computing Minkovski domain tessellations.
  - There is also functionalities for modifying existing polyhedral tessellations in order to change the stabilizers.
  - An interesting research subject is to consider convex bodies, which are not polytopes. As long as we can build a coherent face structure, all is ok.
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 $\text{conv}(G.v)$.

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: $M_{24}.v = \text{Sym}(24).v$, whose face-lattice is given by the Wythoff construction.
VII. Other functionalities of polyhedral
Specific computations

For polytopes we can additionally (up to isomorphism)

- Compute the face-lattice.
- Compute the set of flags.
- Compute belt system: cycles of centrally symmetric facets \((F_1, F_2, \ldots, F_m)\) with \(F_{i-1} \cap F_i\) and \(F_i \cap F_{i+1}\) being antipodal.
- Enumeration of specific subpolytope inside one (only combinatorial).
- For very large polytope, it is possible to use linear programming to get the skeleton.

Other functions:

- Compute the orbits of sublattices of rank \(k\) of fixed determinant of a lattice.
- Compute the Wythoff construction of a polytope.
Volume computations

- All known methods for computing integrals over a polytope $P$ rely on decomposing it into an union (signed or not) of simplices.


We use lrs for computing a tessellation with many triangles in a fast way.

- The method to compute volume is to use the decomposition into facets and then to use a cone decomposition. All the stories about banking system applies but not the ones about Balinski theorem.

- This has been implemented also for the second moment.
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 $\text{conv}(G \cdot v)$ 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.
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.
THANK YOU