Exploiting Polyhedral Symmetries

Achill Schürmann
(Delft University of Technology)

on joint work with David Bremner, Thomas Rehn, Mathieu Dutour Sikirić and Frank Vallentin
Symmetric Polyhedra
Symmetric Polyhedra

- “Beautiful” polyhedra have symmetries
Symmetric Polyhedra

- “Beautiful” polyhedra have symmetries
- Many Polyhedra in Optimization have symmetries
  - TSP Polytope, Matching Polytope, CUT Polytope, ...
Symmetric Polyhedra

- “Beautiful” polyhedra have symmetries
- Many Polyhedra in Optimization have symmetries
  - TSP Polytope, Matching Polytope, CUT Polytope, ...
  - Polyhedra of many MILPs
Symmetry Groups
Symmetry Groups

- Combinatorial, Linear, or Geometric Symmetries
Symmetry Groups

- Combinatorial, Linear, or Geometric Symmetries

\[ C_6 \times C_2 \]

\[ C_6 \times C_2 \]

\[ C_6 \times C_2 \]

\[ C_2 \times C_2 \]

\[ C_6 \times C_2 \]

\[ C_6 \times C_2 \]
Symmetry Groups

- Combinatorial, Linear, or Geometric Symmetries

DEF: A linear automorphism of \( \{v_1, \ldots, v_m\} \subset \mathbb{R}^n \) is a regular matrix \( A \in \mathbb{R}^{n \times n} \) with \( Av_i = v_{\sigma(i)} \) for some \( \sigma \in S_m \).
Detecting Linear Automorphisms
Detecting Linear Automorphisms

**THM:** The group of linear automorphisms is equal to the automorphism group of the complete graph $K_m$ with edge labels $v^t_i Q^{-1} v_j$, where $Q = \sum_{i=1}^{m} v_i v_i^t$.
**THM:** The group of linear automorphisms is equal to the automorphism group of the complete graph $K_m$ with edge labels $v_i^t Q^{-1} v_j$, where $Q = \sum_{i=1}^{m} v_i v_i^t$

$Q = \begin{pmatrix} 4 & -2 \\ -2 & 4 \end{pmatrix}$
Detecting Linear Automorphisms

**THM:** The group of linear automorphisms is equal to the automorphism group of the complete graph $K_m$ with edge labels $v_i^t Q^{-1} v_j$, where $Q = \sum_{i=1}^{m} v_i v_i^t$

\[
Q = \begin{pmatrix} 4 & -2 \\ -2 & 4 \end{pmatrix}
\]

=> use NAUTY by Brendan McKay
Exploiting Symmetries in LPs and IPs
For LPs one can intersect feasible polyhedron with invariant linear subspace.
Exploiting Symmetries in LPs and IPs

- For LPs one can intersect feasible polyhedron with invariant linear subspace
  (not possible for IPs)
Exploiting Symmetries in LPs and IPs

• For LPs one can intersect feasible polyhedron with invariant linear subspace (not possible for IPs)

• For IPs several new approaches have been proposed

=> see survey “Symmetry in Integer Linear Programming” by François Margot (2009)
Representation Conversion
up to symmetry
Representation Conversion

up to symmetry

Recent computational successes:
(with Mathieu Dutour Sikirić and Frank Vallentin)

Representative Conversion

up to symmetry

Recent computational successes:
(with Mathieu Dutour Sikirić and Frank Vallentin)

• Classification of eight dimensional perfect forms, Electron. Res. Announc. AMS, 13 (2007)
  • 1 orbit with 120 vertices in 35 dimensions
  • 25,075,566,937,584 facets in 83092 orbits
Recent computational successes:
(with Mathieu Dutour Sikirić and Frank Vallentin)

  - 1 orbit with 120 vertices in 35 dimensions
  - 25,075,566,937,584 facets in 83092 orbits

  - computation of vertices for many different Voronoi cells of lattices
  - verified that Leech Lattice cell has 307 vertex orbits (Conway, Borcherds, et. al.)
Recent computational successes:
(with Mathieu Dutour Sikirić and Frank Vallentin)

  - 1 orbit with 120 vertices in 35 dimensions
  - 25,075,566,937,584 facets in 83092 orbits

  - computation of vertices for many different Voronoi cells of lattices
  - verified that Leech Lattice cell has 307 vertex orbits (Conway, Borcherds, et. al.)

- The contact polytope of the Leech lattice, preprint at arXiv:0906.1427
  - 1 orbit with 196,560 vertices in 24 dimensions
  - 1,197,362,269,604,214,277,200 many facets in 232 orbits
Adjacency Decomposition Method
Adjacency Decomposition Method

- Find initial orbit(s) / representing vertex(s)
Adjacency Decomposition Method

- Find initial orbit(s) / representing vertex(s)
- For each new orbit representative
  - enumerate neighboring vertices
Adjacency Decomposition Method

- Find initial orbit(s) / representing vertex(s)
- For each new orbit representative
  - enumerate neighboring vertices
  - add as orbit representative if in a new orbit
Adjacency Decomposition Method

• Find initial orbit(s) / representing vertex(s)

• For each new orbit representative
  • enumerate neighboring vertices (up to symmetry)
  • add as orbit representative if in a new orbit

Representation conversion problem
**Adjacency Decomposition Method**

- Find initial orbit(s) / representing vertex(s)
- For each new orbit representative
  - enumerate neighboring vertices \((\text{up to symmetry})\)
  - add as orbit representative if in a new orbit

**Representation conversion problem**

**BOTTLENECK:** Stabilizer and In-Orbit computations
Adjacency Decomposition Method

- Find initial orbit(s) / representing vertex(s)
- For each new orbit representative
  - enumerate neighboring vertices \((\text{up to symmetry})\)
  - add as orbit representative if in a new orbit

**BOTTLENECK:** Stabilizer and In-Orbit computations

\[ \Rightarrow \text{Need of efficient data structures and algorithms for permutation groups: BSGS, (partition) backtracking} \]
A New Software Tool

(DFG-Project SCHU 1503/4-2)
A New Software Tool

(DFG-Project SCHU 1503/4-2)

• helps to compute linear automorphism groups
A New Software Tool

(DFG-Project SCHU 1503/4-2)

• helps to compute linear automorphism groups
• converts polyhedral representations using
  Recursive Decomposition Methods (Incidence/Adjacency)
  (also used by Christof/Reinelt, Deza/Fukuda, ... )
A New Software Tool

(DFG-Project SCHU 1503/4-2)

• helps to compute linear automorphism groups
• converts polyhedral representations using
  Recursive Decomposition Methods (Incidence/Adjacency)
  
  (also used by Christof/Reinelt, Deza/Fukuda, ...)

**EX: 4-dim. cube**
Integrated Methods?

• Can we exploit symmetries within known methods for Representation Conversion? (Vertex/Facet enumeration)
Integrated Methods?

• Can we exploit symmetries within known methods for Representation Conversion? (Vertex/Facet enumeration)
Integrated Methods?

- Can we exploit symmetries within known methods for Representation Conversion? (Vertex/Facet enumeration)

  Incremental: Add vertices incrementally, recompute facets at every step
  (Double Description Method, cdd by Komei Fukuda)
Integrated Methods?

- Can we exploit symmetries within known methods for Representation Conversion? (Vertex/Facet enumeration)

  **Incremental:** Add vertices incrementally, recompute facets at every step (Double Description Method, *cdd* by Komei Fukuda)

  **Pivoting:** Using Simplex pivots (lexicographic reverse search, *lrs* by David Avis)
Integrated Methods?

- Can we exploit symmetries within known methods for Representation Conversion? (Vertex/Facet enumeration)

  **Incremental:** Add vertices incrementally, recompute facets at every step (Double Description Method, cdd by Komei Fukuda)

  **Pivoting:** Using Simplex pivots (lexicographic reverse search, lrs by David Avis)

- Run through adjacent **bases** (sets of affinely independent vertices spanning a facet)
- **Lex-pos bases** are part of a fixed triangulation of the boundary
Bases and Symmetry

- Symmetry group acts on set of all bases
- **but** usually not on lex-pos bases
Bases and Symmetry

- Symmetry group acts on set of all bases
- **but** usually not on lex-pos bases

- lex-pos bases are obtained by a ("symbolic") perturbation
- if $0 \not\in \text{int} P$, vertex $v_i$ may be thought of being scaled by $1 - \varepsilon_i$
  \[
  1 \gg \varepsilon_1 \gg \varepsilon_2 \gg \cdots > 0
  \]
  => destroys symmetry
Adjacency Decomposition for Bases

- Enumeration of all bases up to symmetry works well only in special cases

**EX:** If facets are regular simplices and crosspolytopes
Adjacency Decomposition for Bases

- Enumeration of all bases up to symmetry works well only in special cases

**EX:** If facets are regular simplices and crosspolytopes

- **but** works not so well in general

<table>
<thead>
<tr>
<th>dimension</th>
<th># lex-pos bases</th>
<th># basis orbits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>496</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>186636</td>
<td>6300</td>
</tr>
<tr>
<td>4</td>
<td>48</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>240</td>
<td>17</td>
</tr>
<tr>
<td>Cubes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1440</td>
<td>237</td>
</tr>
<tr>
<td>7</td>
<td>10080</td>
<td>9892</td>
</tr>
<tr>
<td>8</td>
<td>80640</td>
<td>&gt; 209000</td>
</tr>
</tbody>
</table>
IDEA: Orbitwise Perturbation

- Consider some subgroup $G$ of the symmetry group
- Identify orbits $O_1, \ldots, O_k$ of vertices with respect to $G$
- Scale vertices of $O_i$ by $1 \pm \varepsilon_i$ (push/pull), $1 \gg \varepsilon_1 \gg \cdots \gg \varepsilon_k > 0$
IDEA: Orbitwise Perturbation

- Consider some subgroup $G$ of the symmetry group
- Identify orbits $O_1, \ldots, O_k$ of vertices with respect to $G$
- Scale vertices of $O_i$ by $1 \pm \varepsilon_i$ (push/pull), $1 \gg \varepsilon_1 \gg \cdots \gg \varepsilon_k > 0$

$O_1 = \{1, 8\}, O_2 = \{2, \ldots, 7\}$
IDEA: Orbitwise Perturbation

- Consider some **subgroup** $G$ of the symmetry group
- Identify orbits $O_1, \ldots, O_k$ of vertices with respect to $G$
- Scale vertices of $O_i$ by $1 \pm \varepsilon_i$ (push/pull), $1 \gg \varepsilon_1 \gg \cdots \gg \varepsilon_k > 0$

- can be implemented symbolically
- may **not** yield a triangulation
- **but** a set of bases on which $G$ acts

$O_1 = \{1, 8\}, O_2 = \{2, \ldots, 7\}$
Works better...

<table>
<thead>
<tr>
<th>dimension</th>
<th># lex-pos bases</th>
<th># basis orbits</th>
<th># pert. basis orbits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cut</td>
<td>10</td>
<td>496</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>186636</td>
<td>6300</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>48</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>240</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>Cubes</td>
<td>6</td>
<td>1440</td>
<td>237</td>
</tr>
<tr>
<td>7</td>
<td>10080</td>
<td>9892</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>80640</td>
<td>&gt; 209000</td>
<td>1</td>
</tr>
</tbody>
</table>

...but there is still a lot to do!
Conclusion

- Polyhedral Symmetries can be exploited!
Conclusion

• Polyhedral Symmetries can be exploited!
• in Integer Programming and MILPs
Conclusion

- Polyhedral Symmetries can be exploited!
  - in Integer Programming and MILPs
  - for Polyhedral Representation Conversion
Conclusion

• Polyhedral Symmetries can be exploited!
  • in Integer Programming and MILPs
  • for Polyhedral Representation Conversion
  • for other polyhedral computations... like Lattice Point Enumeration / Counting
Conclusion

- Polyhedral Symmetries can be exploited!
  - in Integer Programming and MILPs
  - for Polyhedral Representation Conversion
  - for other polyhedral computations... like Lattice Point Enumeration / Counting

ToDo

- Create efficient computational tools
Conclusion

• Polyhedral Symmetries can be exploited!
  • in Integer Programming and MILPs
  • for Polyhedral Representation Conversion
  • for other polyhedral computations... like Lattice Point Enumeration / Counting

ToDo

• Create efficient computational tools
• Integrate tools from Computational Group Theory
Conclusion

- Polyhedral Symmetries can be exploited!
- in Integer Programming and MILPs
- for Polyhedral Representation Conversion
- for other polyhedral computations... like Lattice Point Enumeration / Counting

ToDo

- Create efficient computational tools
- Integrate tools from Computational Group Theory
- Collect Benchmark Problems

=> Please, send me your data!
Merci!

Recent Survey:
International Congress on Mathematical Software 2010
ICMS, Kobe (Japan), Sep. 13-17

Section on Optimization and polyhedral computation
organized by Komei Fukuda, Michael Joswig and Achill Schürmann

http://www.mathsoftware.org/