Computing symmetry groups of polyhedra

David Bremner

New Brunswick

May 12, 2014

Joint work with Mathieu Dutour Sikirić, Dmitrii V. Pasechnik, Thomas Rehn and Achill Schürmann

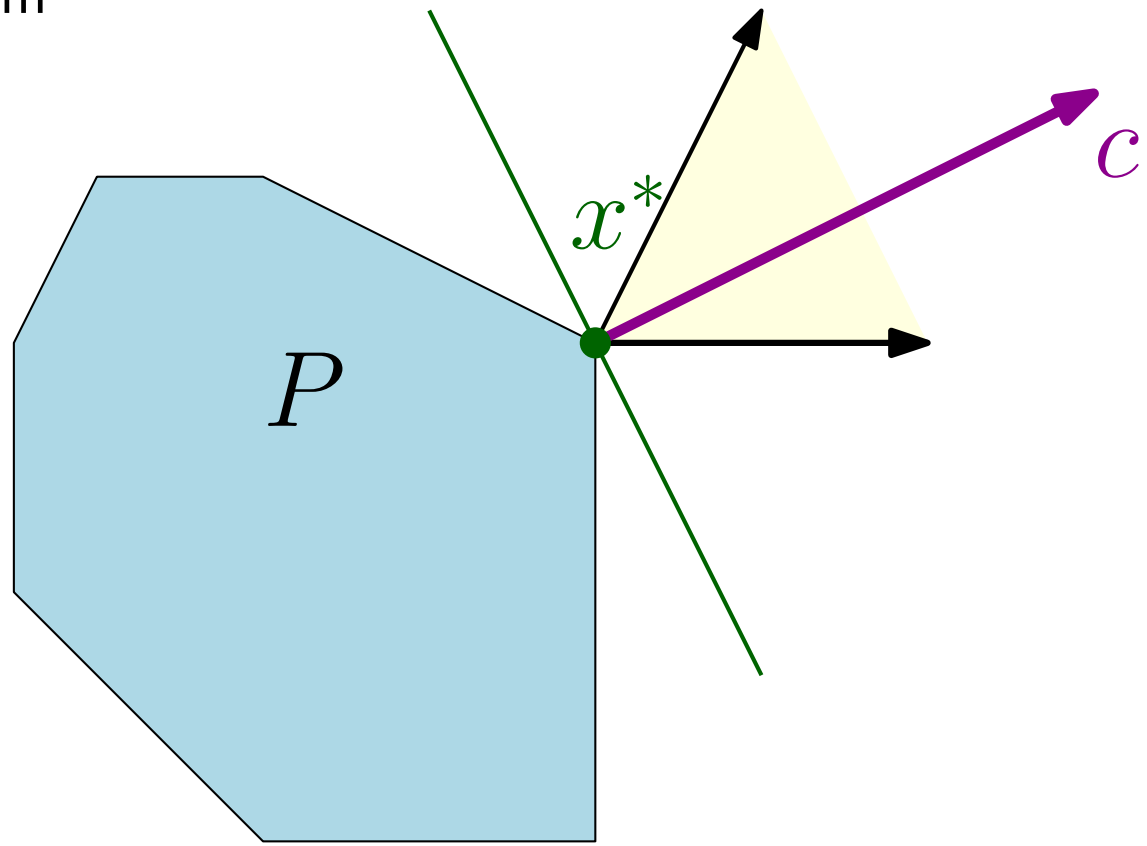
Outline

Linear Programs

Let $P = \{ x \mid Ax \leq b \}, A \in \mathbb{R}^{m \times d}, b \in \mathbb{R}^m$ For $c \in \mathbb{R}^d$, the optimization problem

$$\max c^T x$$
$$x \in P$$

is called a *Linear Program*



Convexity

positive combination

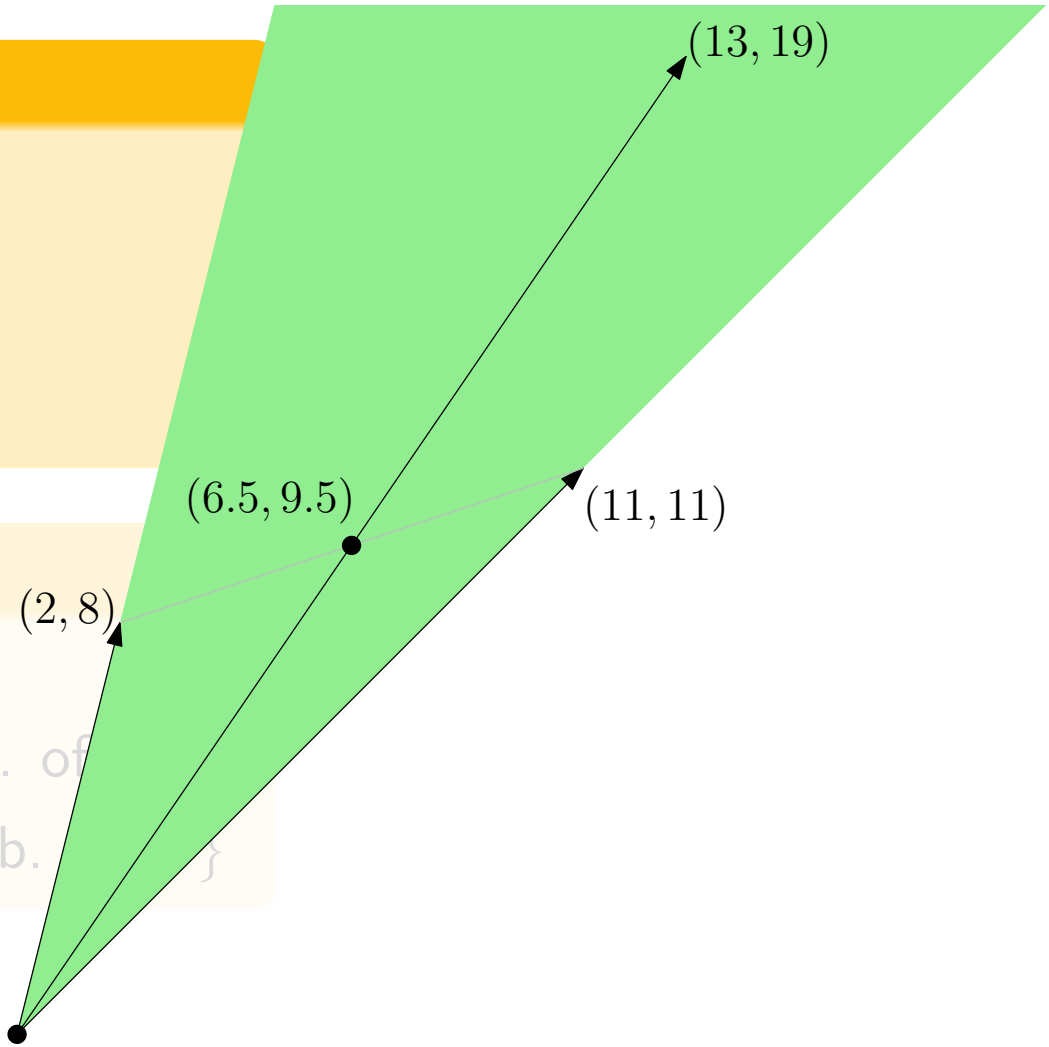
$$y = \sum_i \lambda_i x_i$$

$$\lambda_i \geq 0$$

positive/convex hull

$$\mathrm{pos}(X) = \{\, y \in \mathbb{R}^n \mid y \text{ pos. comb. of } \}$$

$$\mathrm{conv}(X) = \{\, y \in \mathbb{R}^n \mid y \text{ conv. comb. } \}$$



Convexity

convex combination

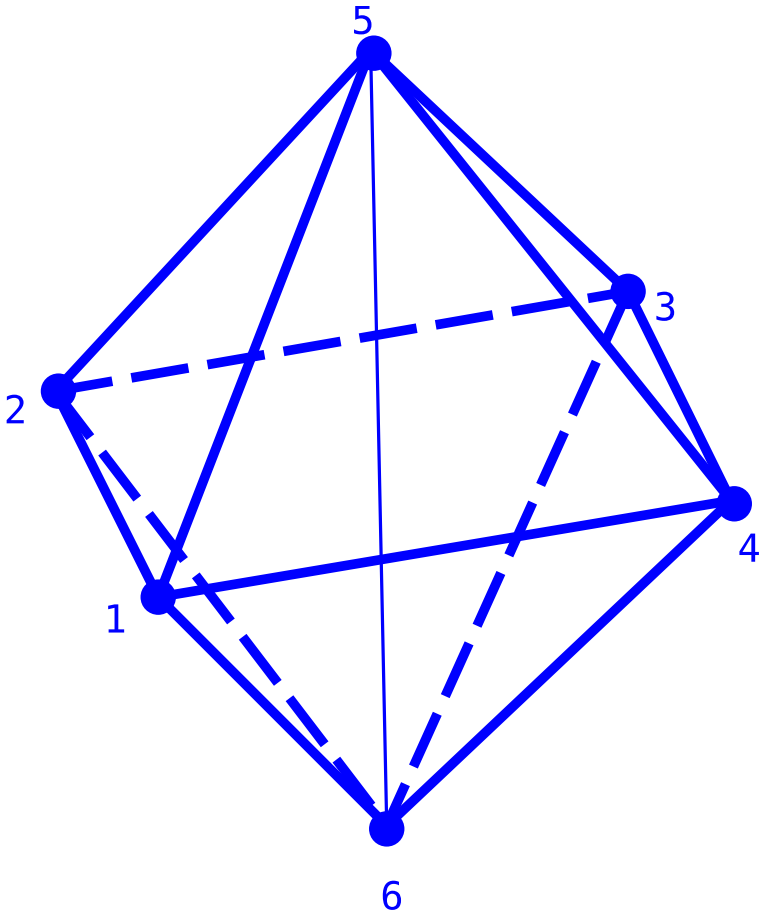
$$y = \sum_i \lambda_i x_i$$

$$\lambda_i \geq 0, \qquad \sum_i \lambda_i = 1$$

positive/convex hull

$$\mathrm{pos}(X) = \{\, y \in \mathbb{R}^n \mid y \text{ pos. comb. of } X \,\}$$

$$\mathrm{conv}(X) = \{\, y \in \mathbb{R}^n \mid y \text{ conv. comb. of } X \,\}$$



Convexity

convex combination

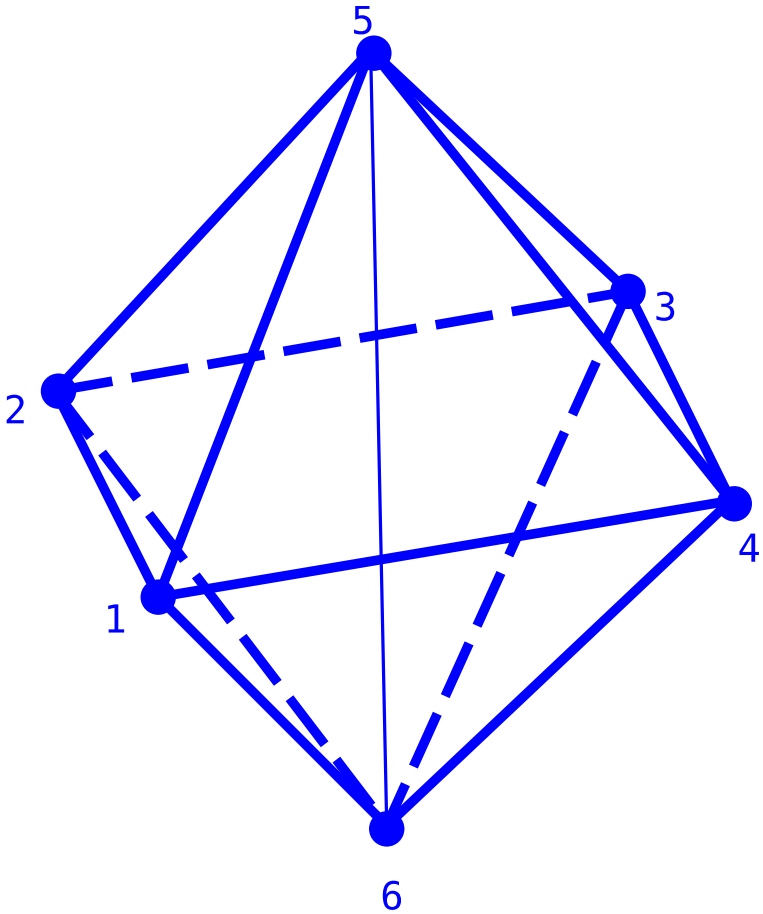
$$y = \sum_i \lambda_i x_i$$

$$\lambda_i \geq 0, \qquad \sum_i \lambda_i = 1$$

positive/convex hull

$$\text{pos}(X) = \{\, y \in \mathbb{R}^n \mid y \text{ pos. comb. of } X \,\}$$

$$\text{conv}(X) = \{\, y \in \mathbb{R}^n \mid y \text{ conv. comb. of } X \,\}$$



$\mathcal{H}$- and $\mathcal{V}$-representations

$\mathcal{V}$-representations

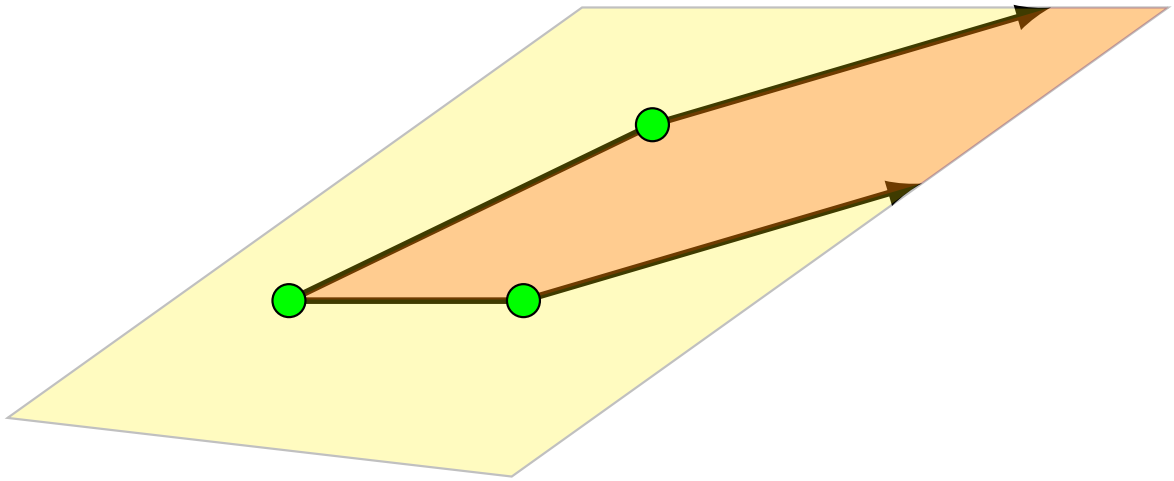
Polyhedron $\mathcal{V}(P) = \operatorname{conv} X + \operatorname{pos} Y$

Cone $\mathcal{V}(P) = \operatorname{pos} Y$

$\mathcal{H}$-representations

Polyhedron $\mathcal{H}(P) = \{ Ax \geq b \}$

Cone $\mathcal{H}(P) = \{ A'x' \geq 0 \}$



$\mathcal{H}$- and $\mathcal{V}$-representations

$\mathcal{V}$-representations

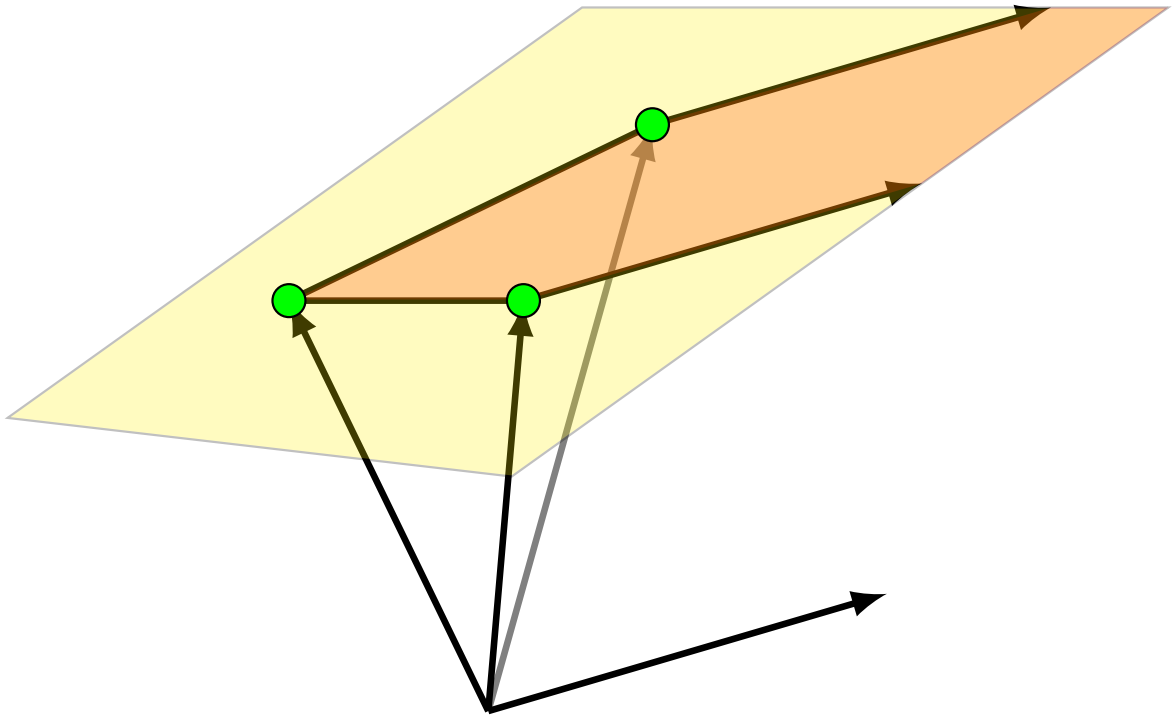
Polyhedron $\mathcal{V}(P) = \operatorname{conv} X + \operatorname{pos} Y$

Cone $\mathcal{V}(P) = \operatorname{pos} Y$

$\mathcal{H}$-representations

Polyhedron $\mathcal{H}(P) = \{ Ax \geq b \}$

Cone $\mathcal{H}(P) = \{ A'x' \geq 0 \}$



$\mathcal{H}$- and $\mathcal{V}$-representations

$\mathcal{V}$-representations

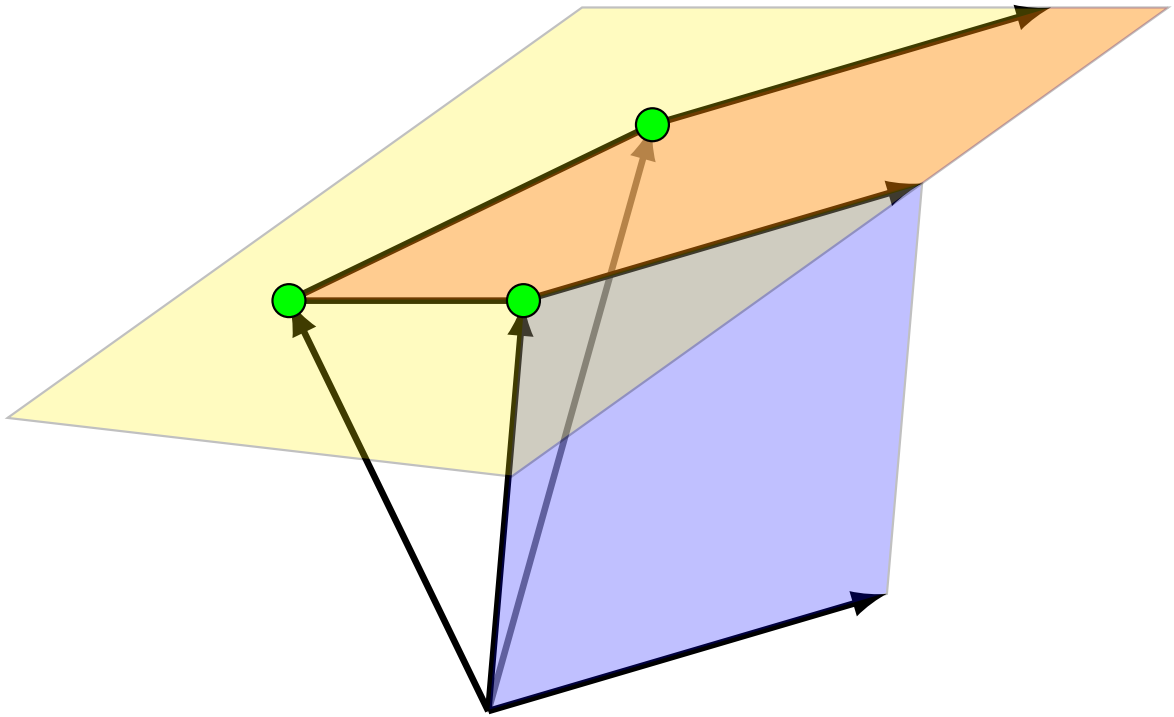
Polyhedron $\mathcal{V}(P) = \operatorname{conv} X + \operatorname{pos} Y$

Cone $\mathcal{V}(P) = \operatorname{pos} Y$

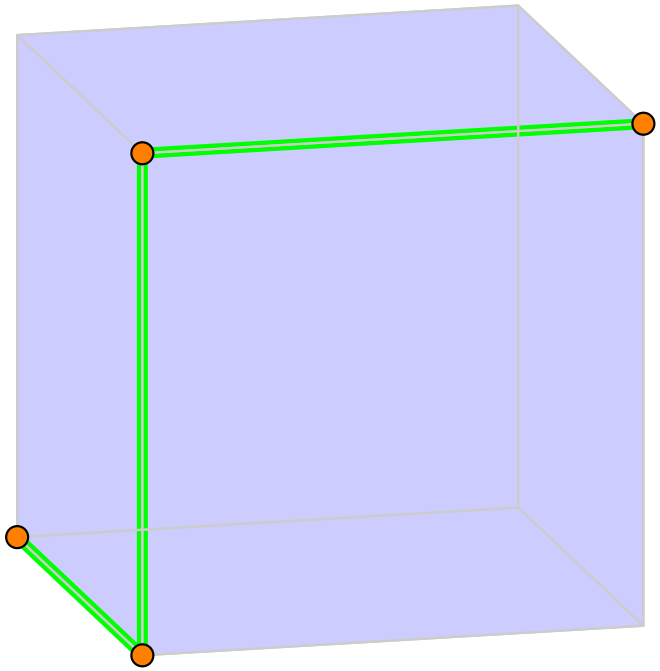
$\mathcal{H}$-representations

Polyhedron $\mathcal{H}(P) = \{ Ax \geq b \}$

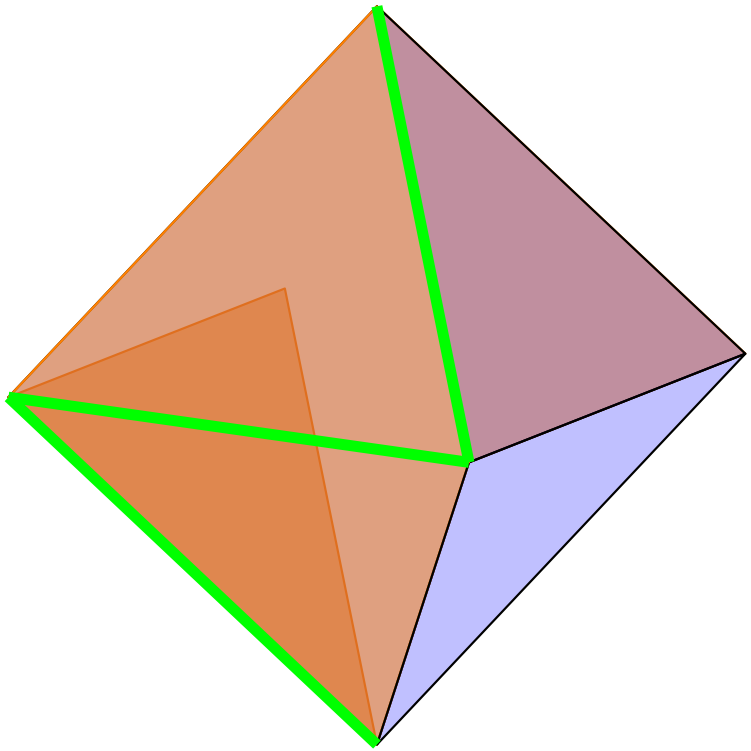
Cone $\mathcal{H}(P) = \{ A'x' \geq 0 \}$



Faces of polyhedra



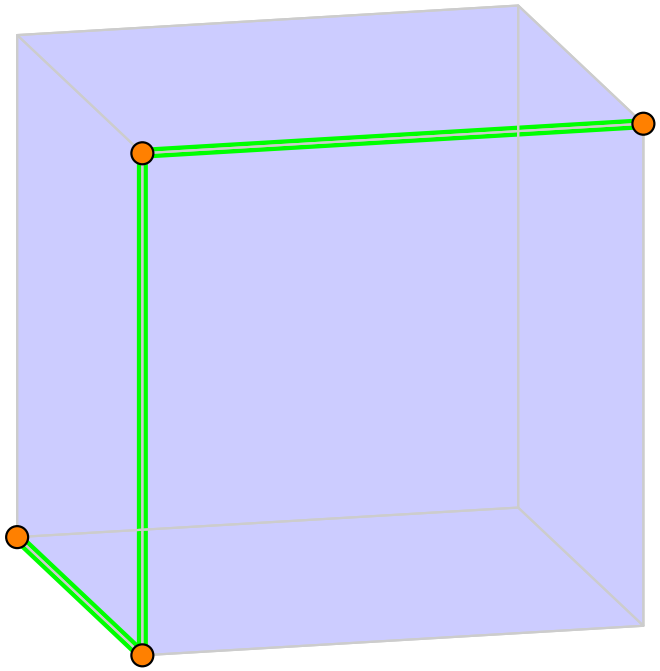
face lattice lattice of faces, ordered by inclusion



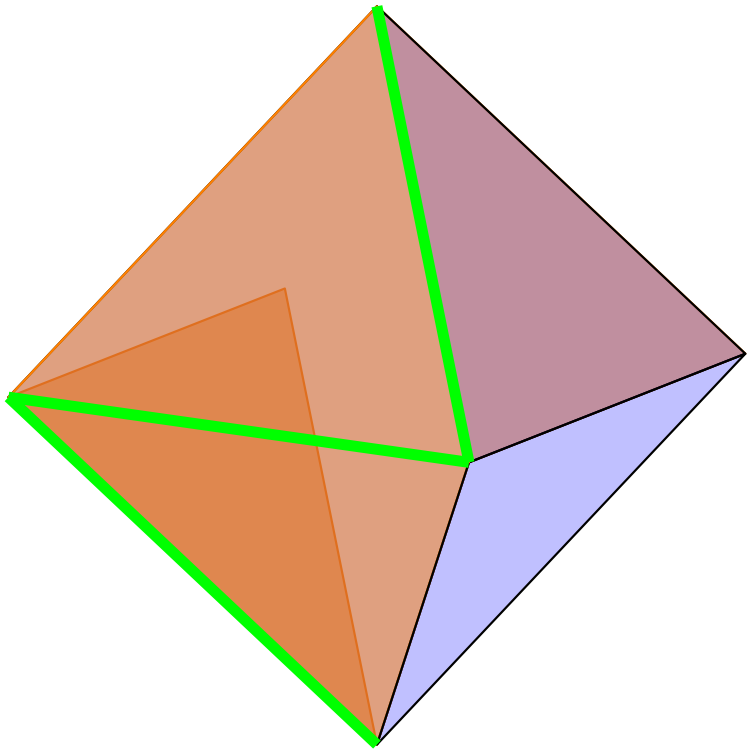
k-face intersection of P and supporting hyperplace, of dimension k.

k-skeleton All k-faces

Faces of polyhedra



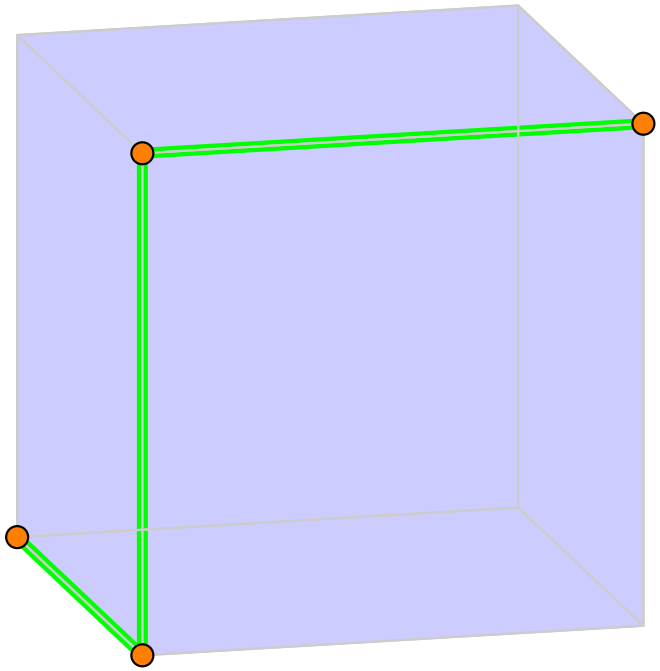
face lattice lattice of faces, ordered by inclusion



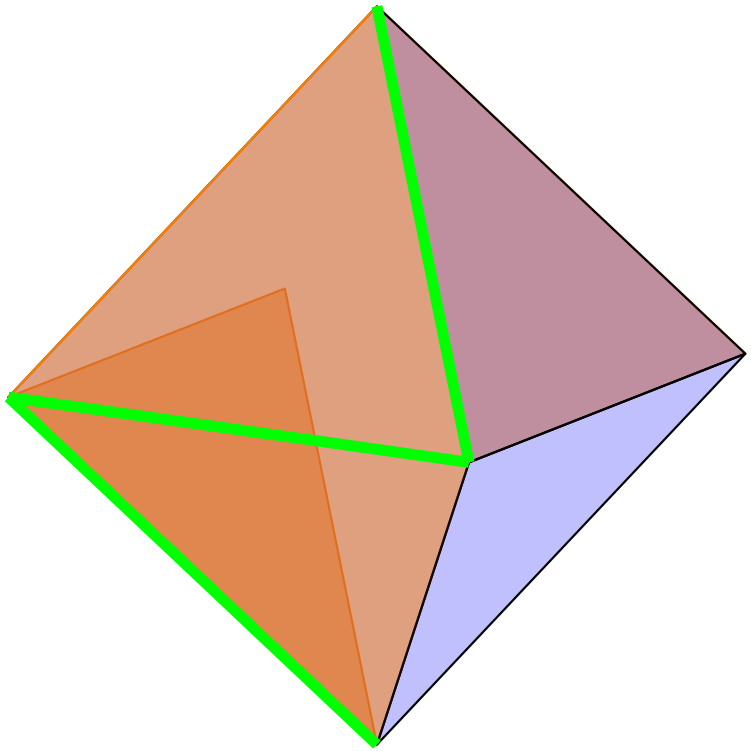
k-face intersection of P and supporting hyperplace, of dimension k.

k-skeleton All *k*-faces

Faces of polyhedra



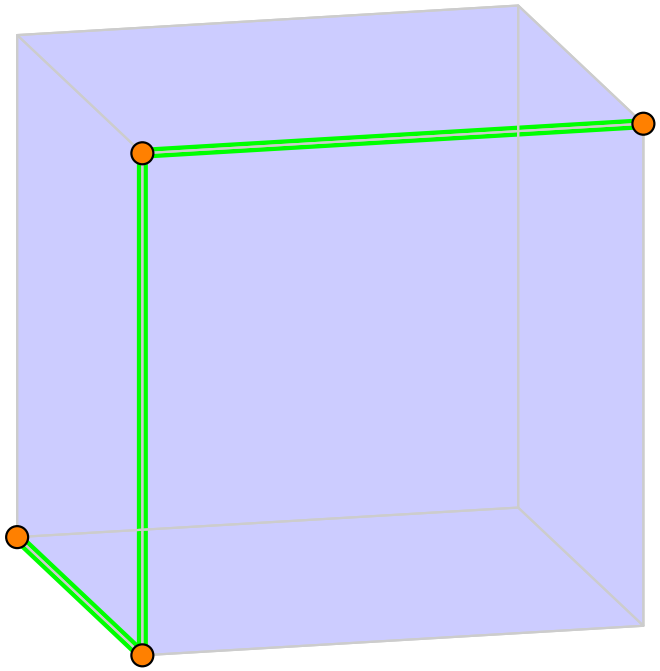
face lattice lattice of faces, ordered by inclusion



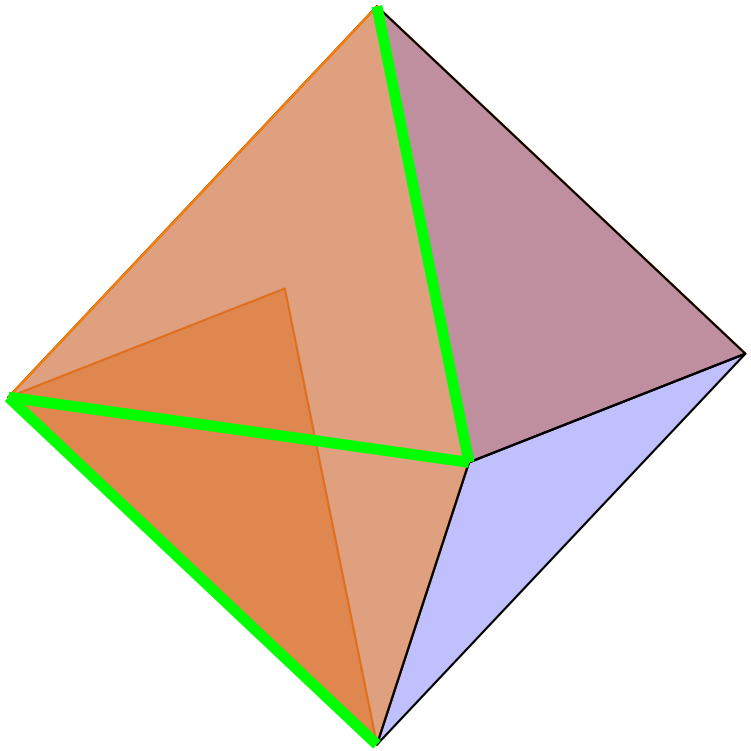
k-face intersection of P and supporting hyperplace, of dimension k.

k-skeleton All k-faces

Faces of polyhedra



face lattice lattice of faces, ordered by inclusion



k-face intersection of P and supporting hyperplace, of dimension k.

k-skeleton All k-faces

Outline

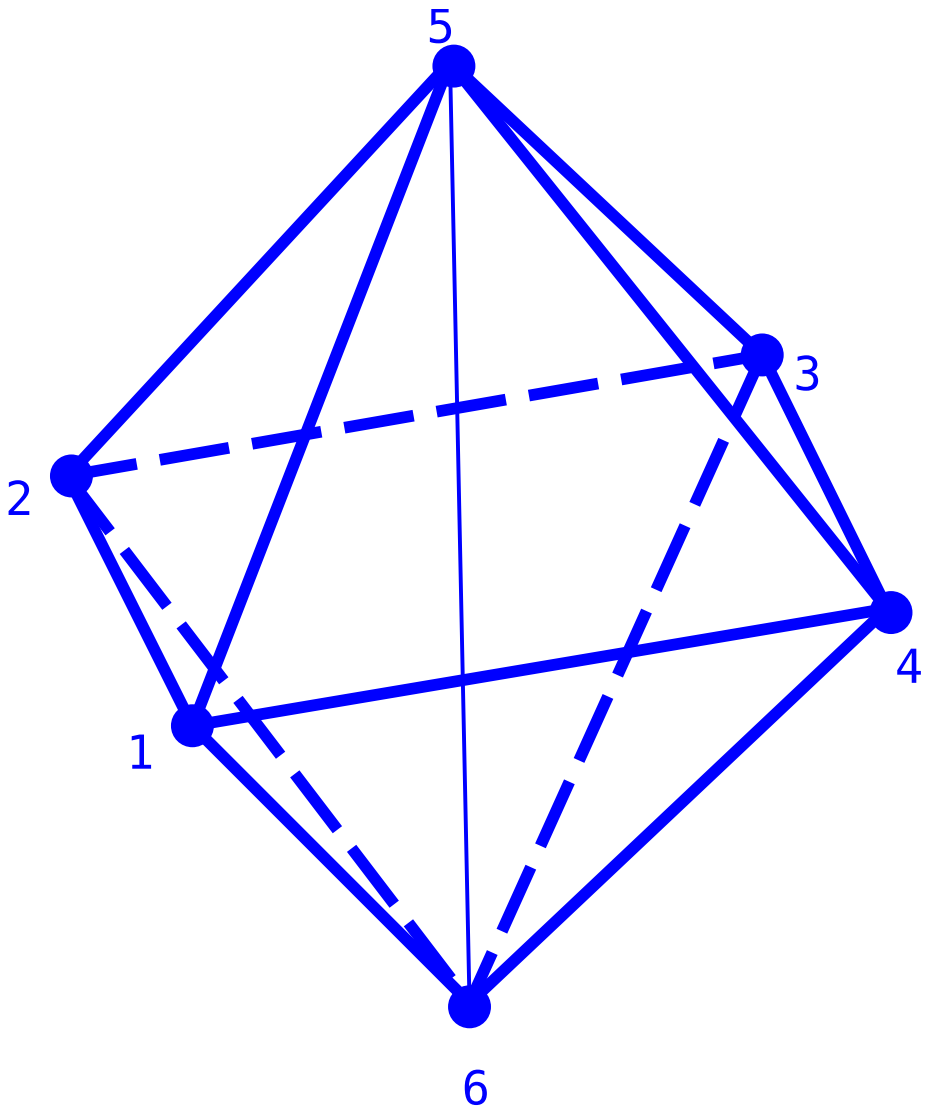
Permutation Groups

Symmetries as permutations $(5,6)$, $(1,2,3,4)$, $(1,2)(3,4)$, not $(4,5)$

Composing symmetries $(1,2,3,4) \circ (1,2)(3,4) = (1,4)(2,3)$

Groups $\circ$ is *associative*, permutations are *invertible*, $()$ is *identity*, so we have a *group*

Sym(k) all permutations of k elements



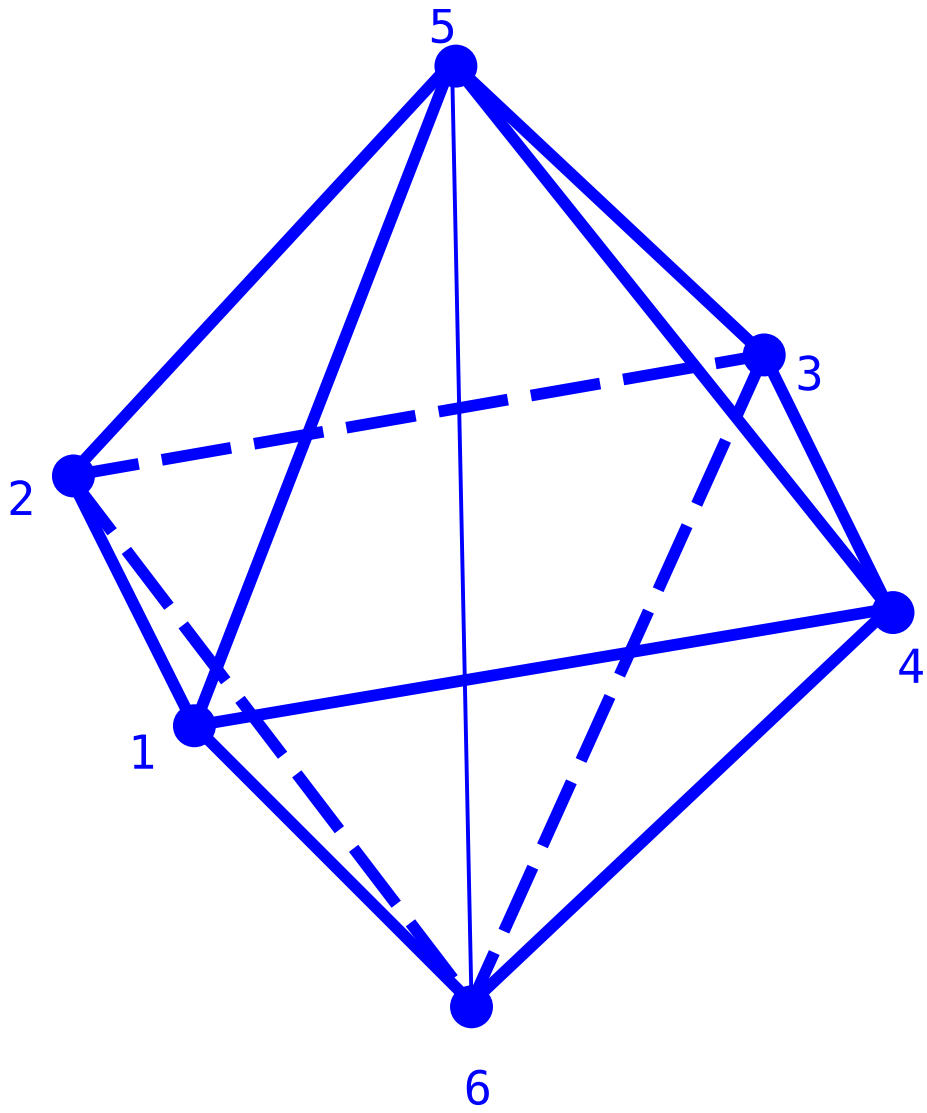
Permutation Groups

Symmetries as permutations $(5,6)$, $(1,2,3,4)$, $(1,2)(3,4)$, not $(4,5)$

Composing symmetries $(1,2,3,4) \circ (1,2)(3,4) = (1,4)(2,3)$

Groups $\circ$ is *associative*, permutations are *invertible*, $()$ is *identity*, so we have a *group*

Sym(k) all permutations of k elements



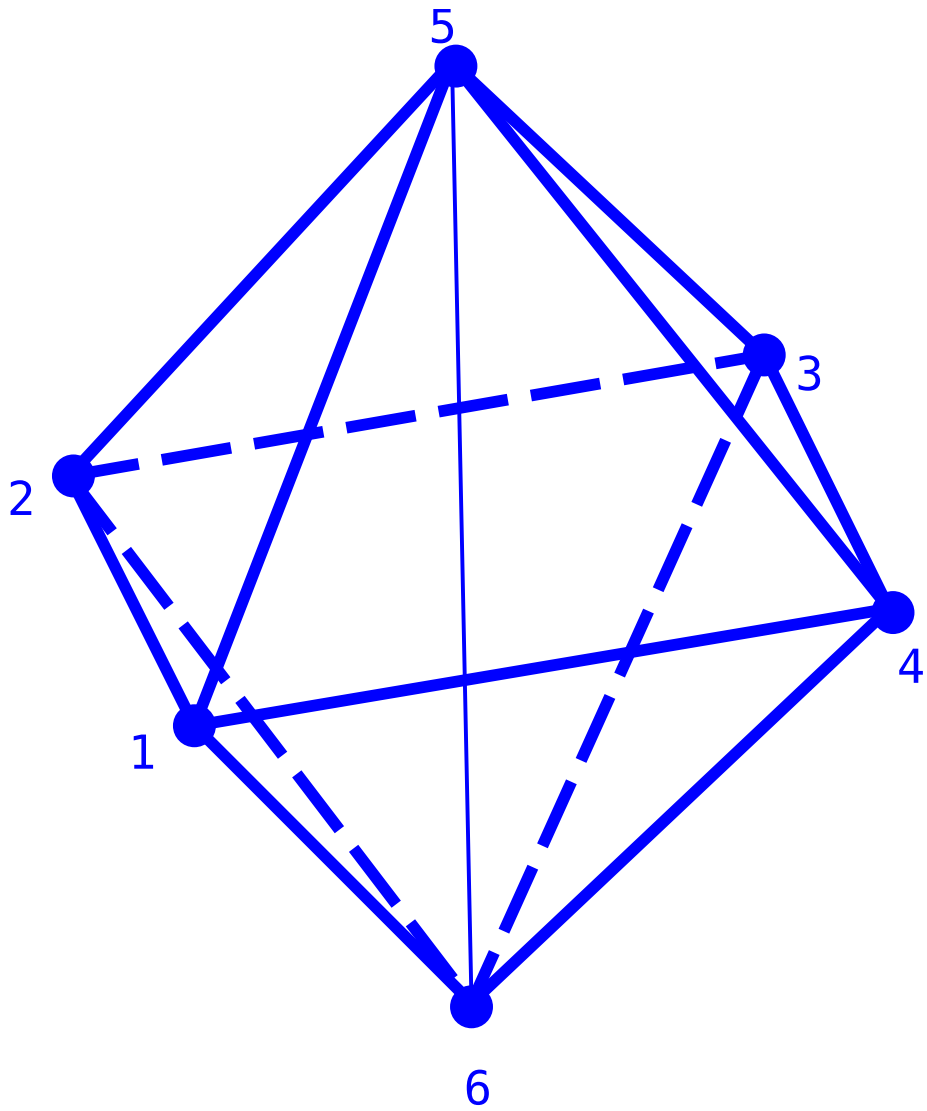
Permutation Groups

Symmetries as permutations $(5,6)$, $(1,2,3,4)$, $(1,2)(3,4)$, not $(4,5)$

Composing symmetries $(1,2,3,4) \circ (1,2)(3,4) = (1,4)(2,3)$

Groups $\circ$ is *associative*, permutations are *invertible*, $()$ is *identity*, so we have a *group*

Sym(k) all permutations of k elements



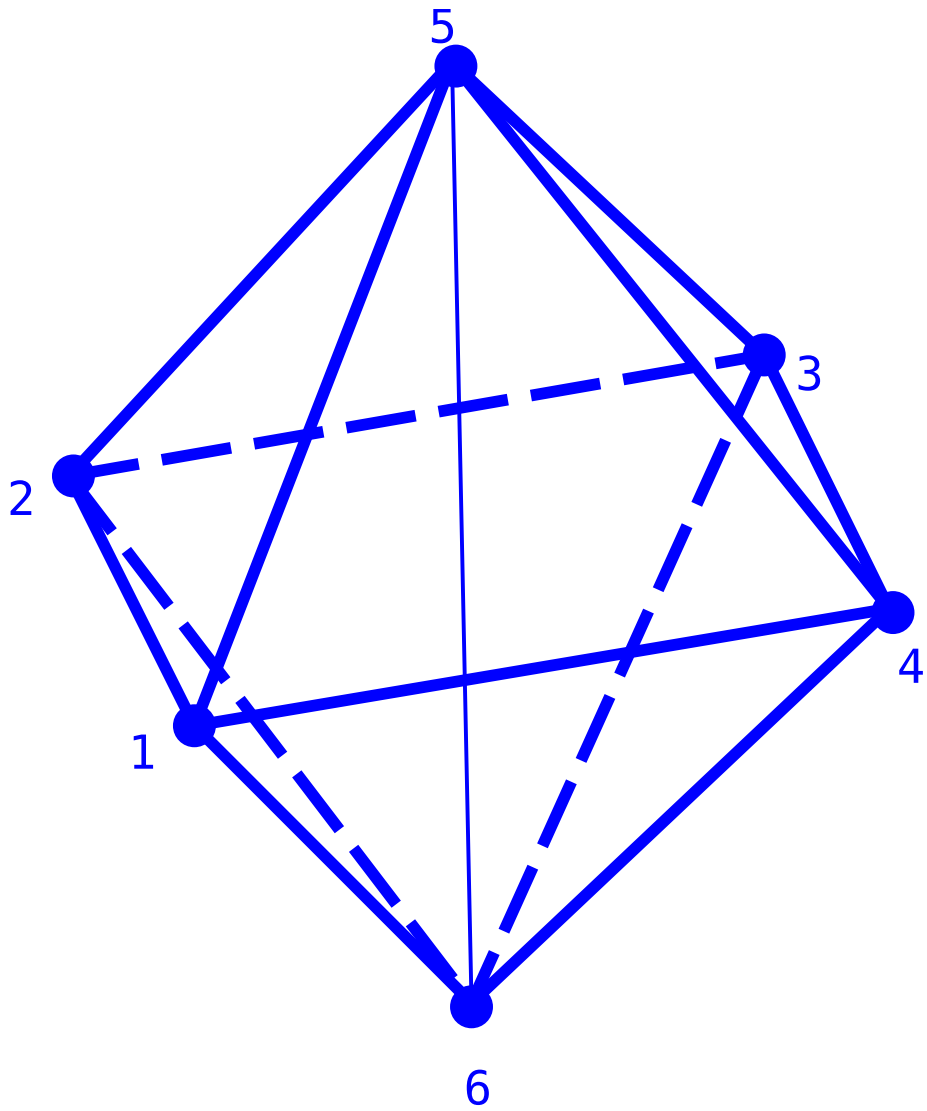
Permutation Groups

Symmetries as permutations $(5,6)$, $(1,2,3,4)$, $(1,2)(3,4)$, not $(4,5)$

Composing symmetries $(1,2,3,4) \circ (1,2)(3,4) = (1,4)(2,3)$

Groups $\circ$ is *associative*, permutations are *invertible*, $()$ is *identity*, so we have a *group*

Sym(k) all permutations of k elements



Group Generators

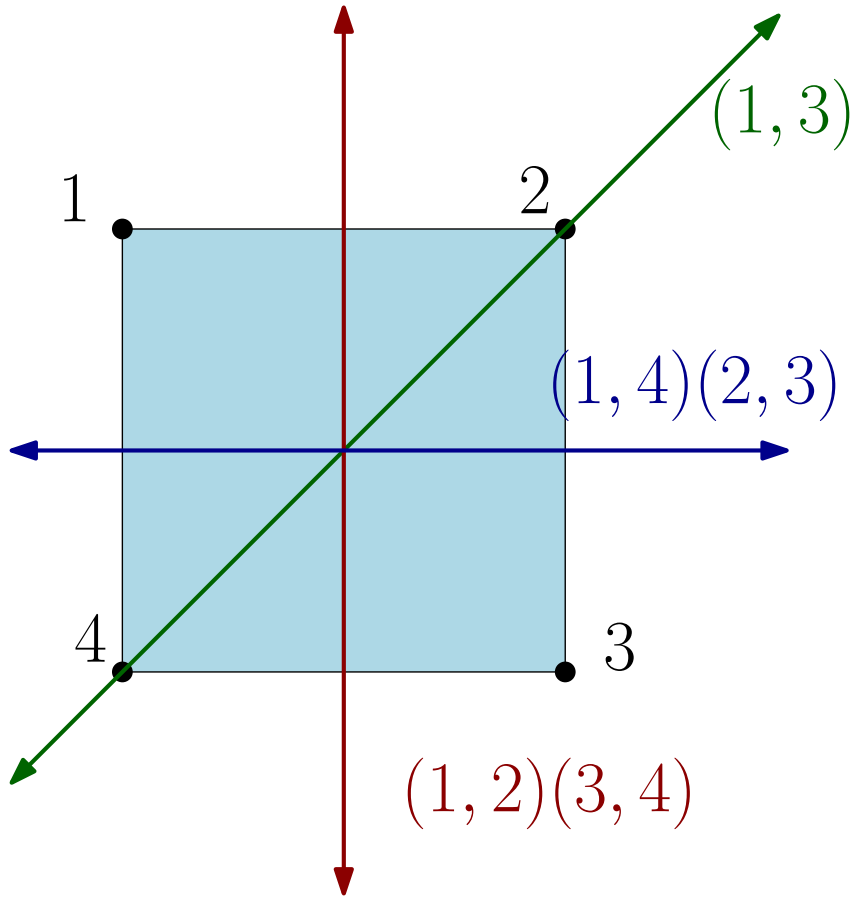
generated subgroup

$$\langle \gamma_1, \ldots \gamma_k \rangle = \prod_{\substack{g_j \in \{\gamma_i, \gamma_i^{-1}\} \\ j \in \{1 \ldots m\}}} g_j$$

generators

Γ *generates* group G if

$$\langle \Gamma \rangle = G$$



Group Generators

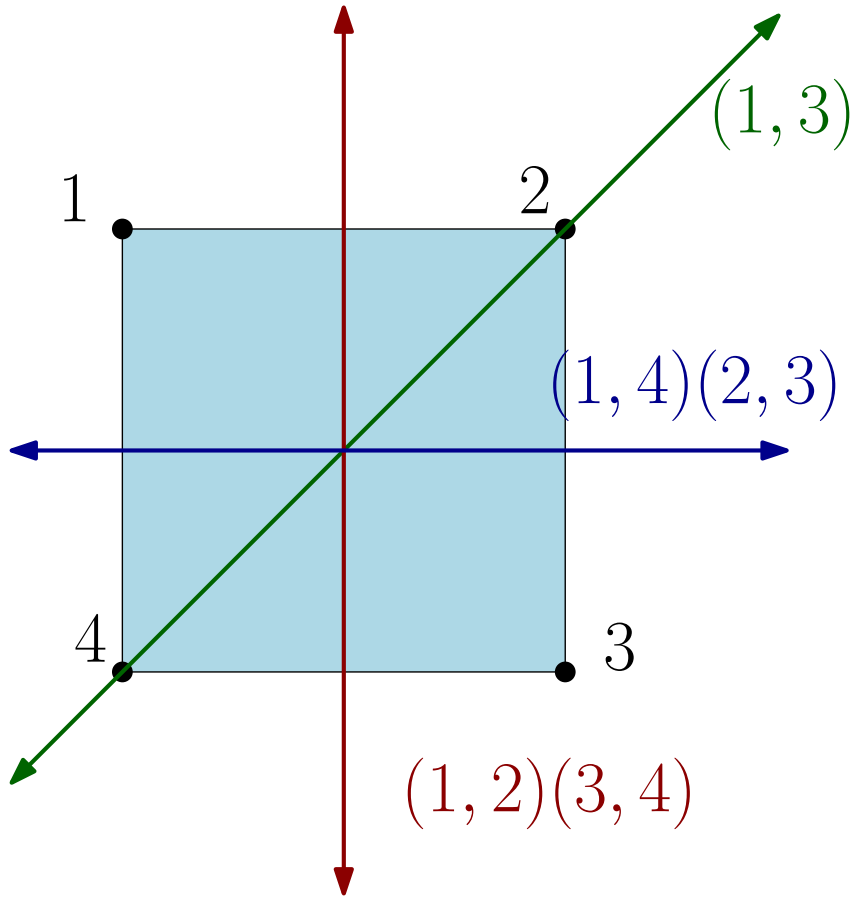
generated subgroup

$$\langle \gamma_1, \ldots \gamma_k \rangle = \prod_{\substack{g_j \in \{\gamma_i, \gamma_i^{-1}\} \\ j \in \{1 \ldots m\}}} g_j$$

generators

Γ *generates* group G if

$$\langle \Gamma \rangle = G$$



Matrix Groups

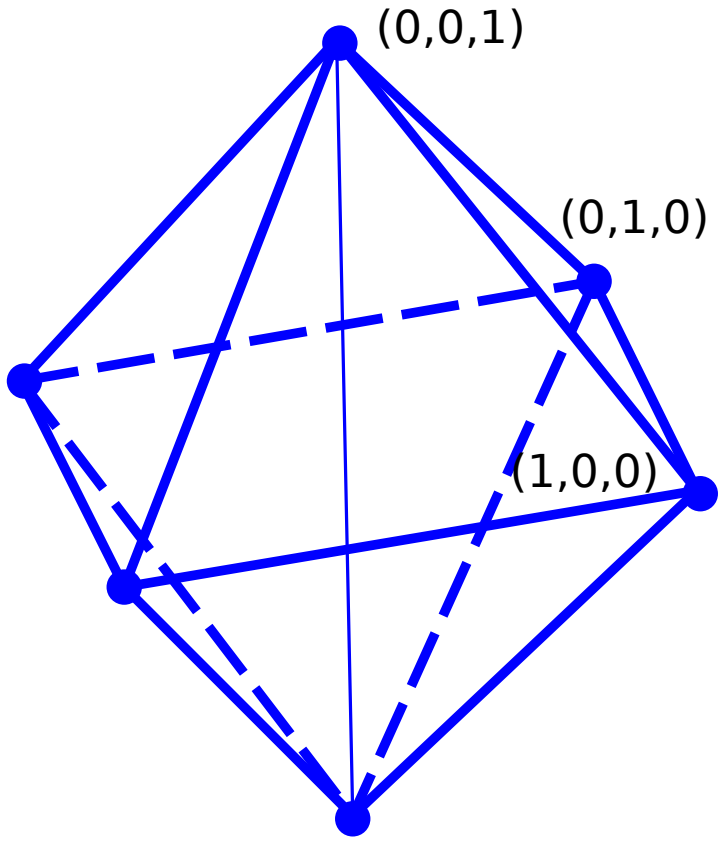
Symmetries as matrices

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Composing symmetries

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{GL}_n(\mathbb{R}) = \{ \text{invertable } n \times n \text{ matrices} \}$$



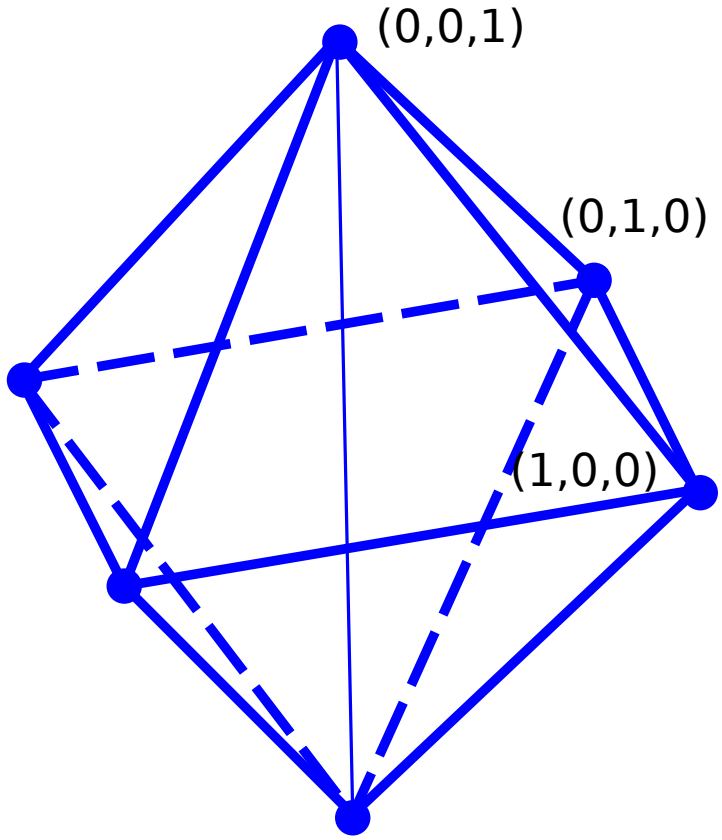
Matrix Groups

Symmetries as matrices

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Composing symmetries

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\mathrm{GL}_n(\mathbb{R}) = \{ \text{invertable } n \times n \text{ matrices} \}$$

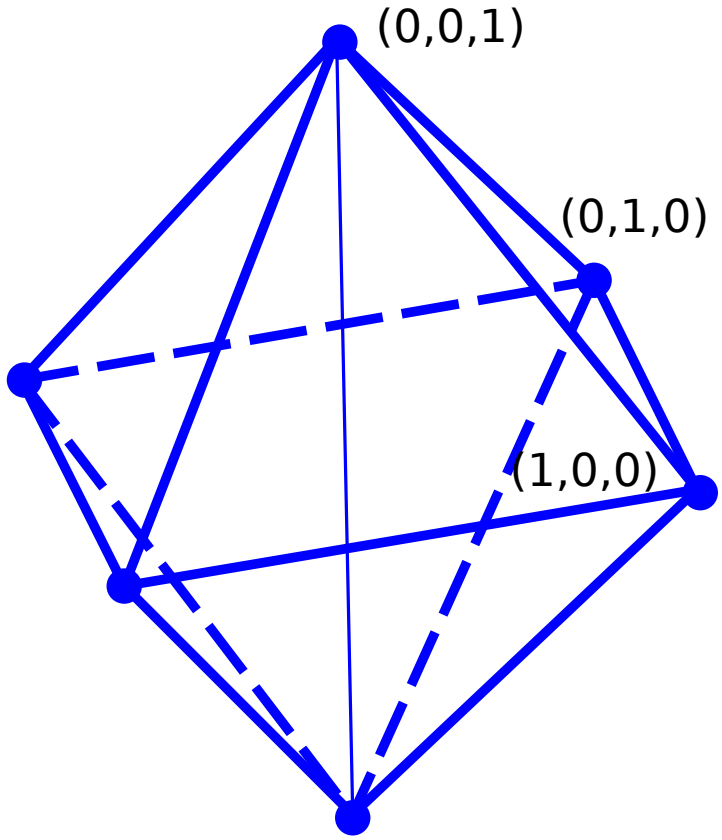
Matrix Groups

Symmetries as matrices

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Composing symmetries

$$\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

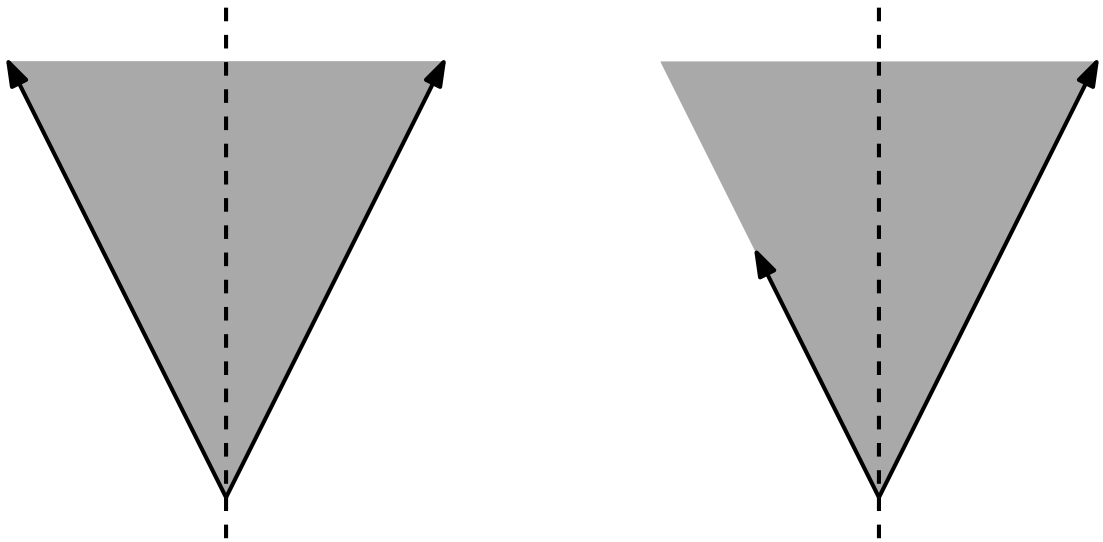


$$\mathrm{GL}_n(\mathbb{R}) = \{ \text{invertable } n \times n \text{ matrices} \}$$

Polyhedral Symmetries

$$\mathrm{Lin}_v(P) \subset \mathrm{Proj}(P) \subset \mathrm{Comb}(P) \subset \mathrm{Skel}_k(P)$$

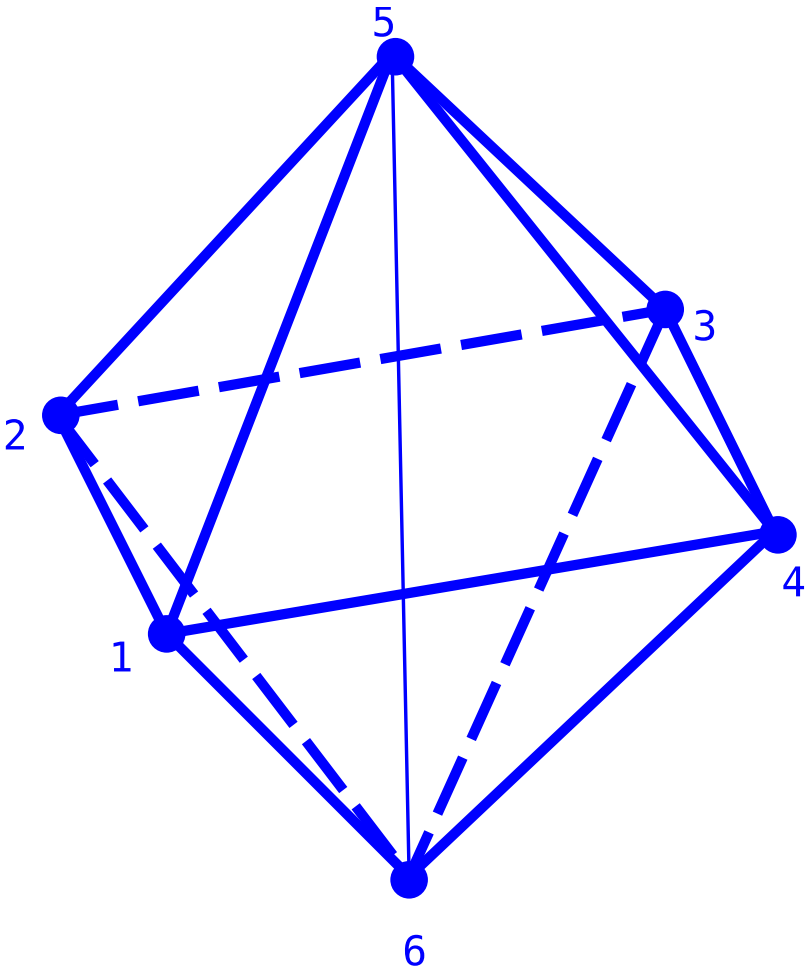
Type	**Preserves**
$\mathrm{Skel}_k(P)$	k-skeleton
$\mathrm{Comb}(P)$	Face Lattice
$\mathrm{Proj}(P)$	Cone
$\mathrm{Lin}_v(P)$	Generators v

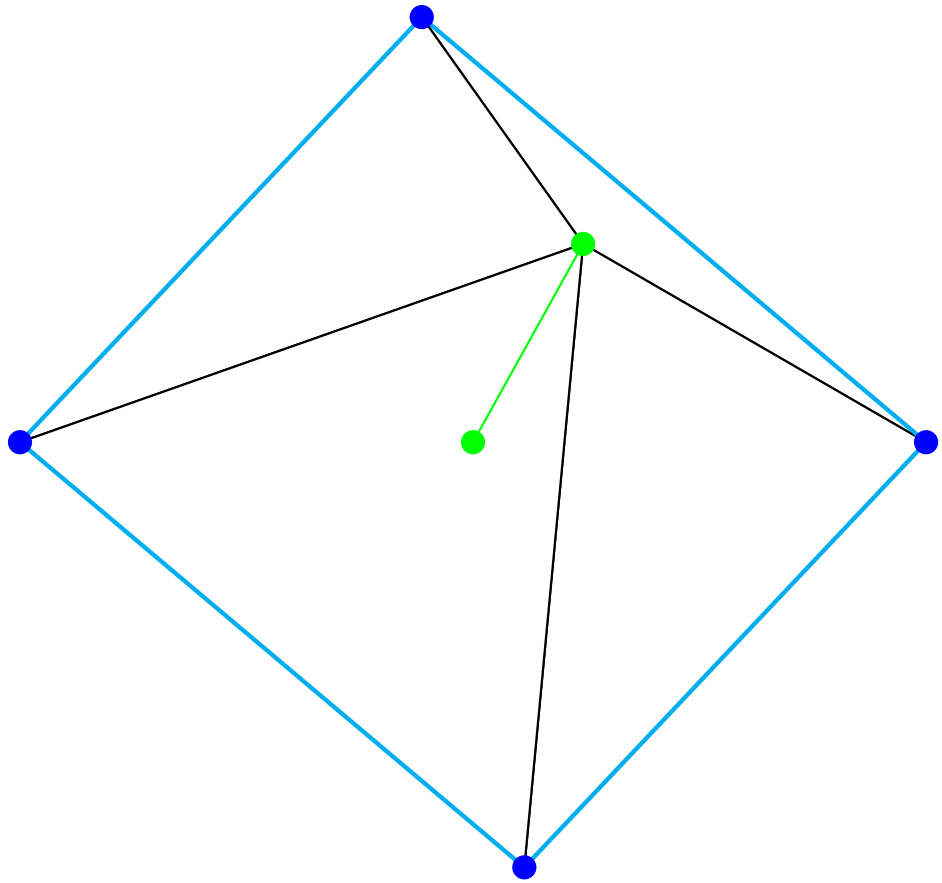


Orbits and Fixed Sets

orbit $\mathrm{Orbit}_G(x) = \{\, g(x) \mid \text{ symmetry } g \in G \,\}$

fixed set $\mathrm{Fix}_G(S) = \{\, x \in S \mid \mathrm{Orbit}_G(x) = \{\, x \,\} \,\}$



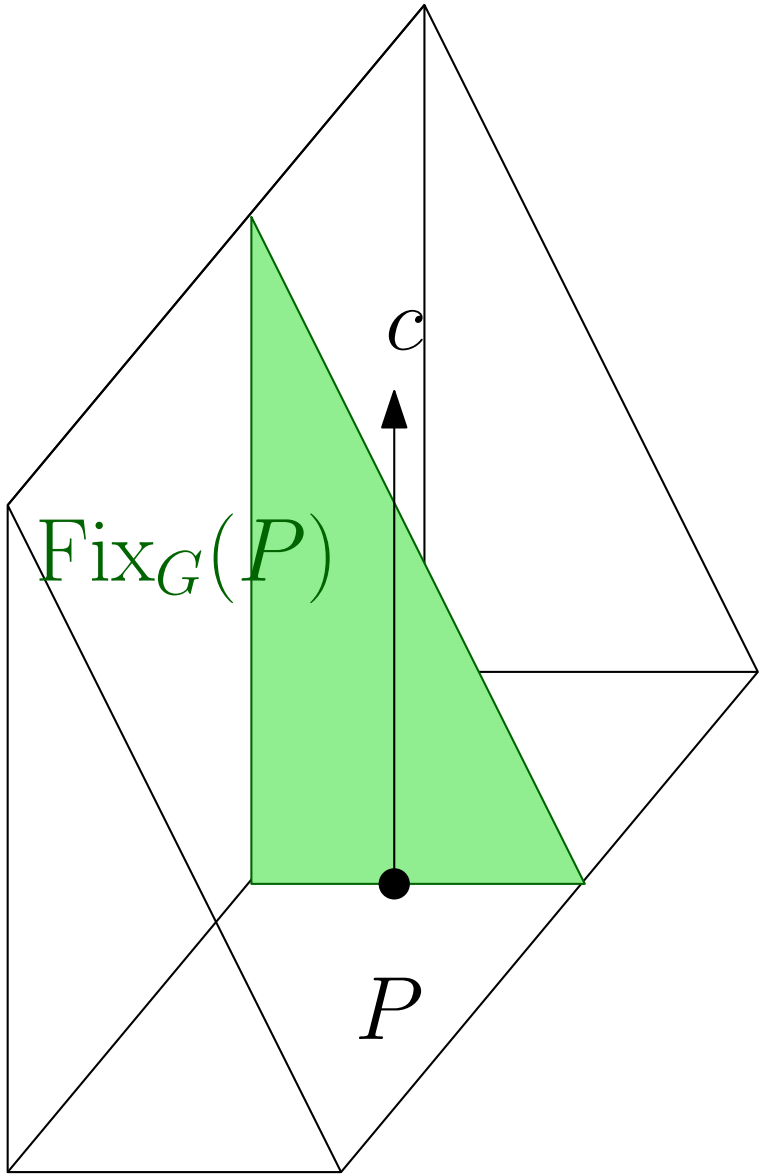


Symmetry in Linear Programming

- Let G be the group of symmetries of $P \cup \{ c \}$

$$\max_{x \in P} c^T x = \max_{x \in \mathrm{Fix}_G(P)} c^T x$$

- The fixed set can be computed efficiently given the generators of G (see e.g. Bödi, Herr, Joswig, 2013)

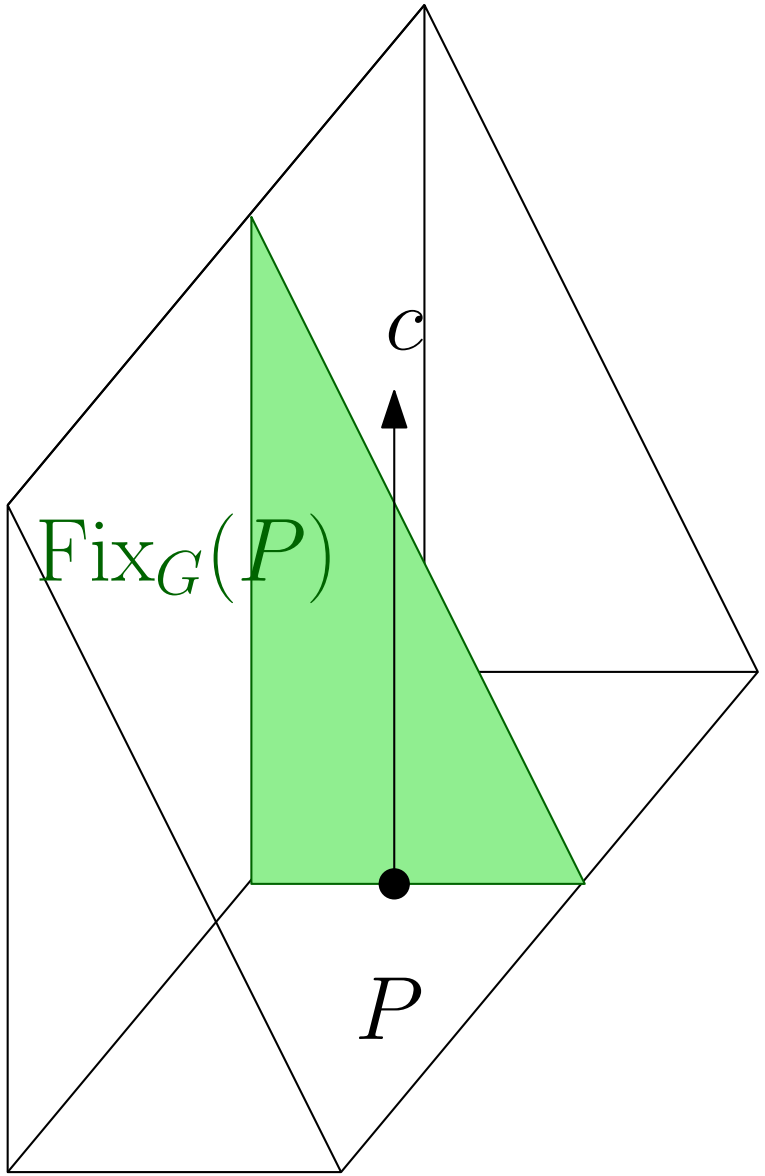


Symmetry in Linear Programming

- Let G be the group of symmetries of $P \cup \{ c \}$

$$\max_{x \in P} c^T x = \max_{x \in \mathrm{Fix}_G(P)} c^T x$$

- The fixed set can be computed efficiently given the generators of G (see e.g. Bödi, Herr, Joswig, 2013)



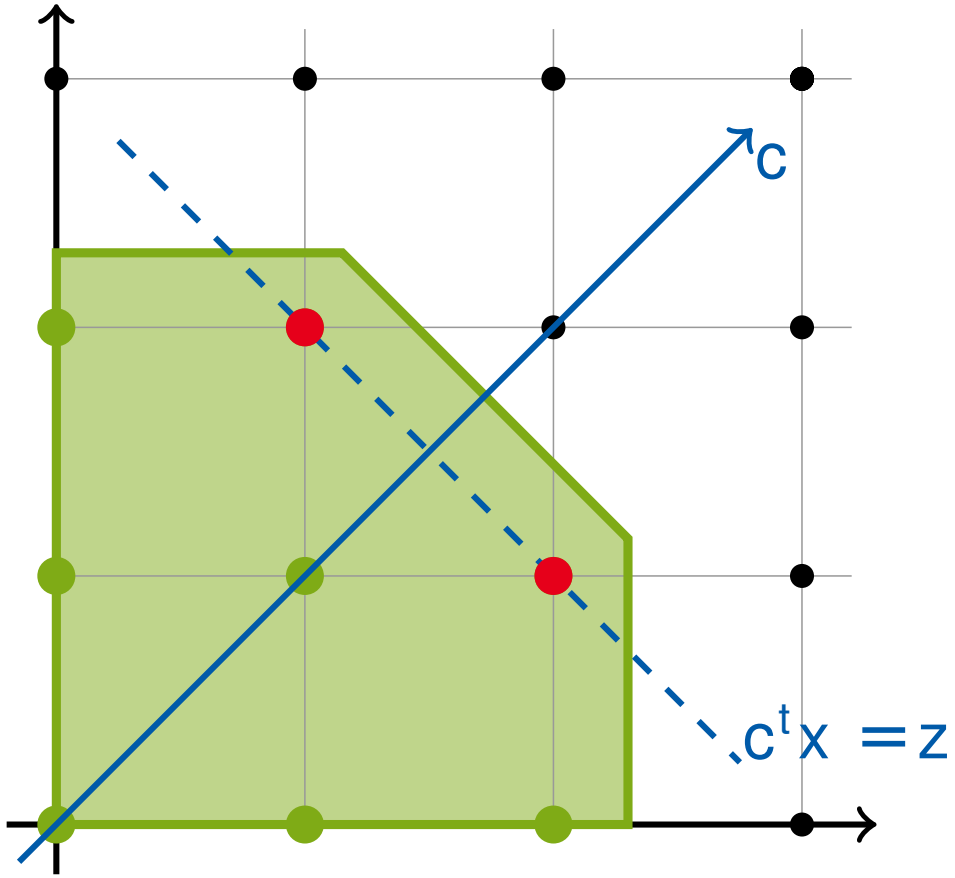
Integer Programming: Core Set Methods

c-layer

Hyperplane orthogonal to c containing integer points.

For sufficiently small number of orbits on the elementary vectors

- The number c-layers can be bounded by the dimension
- Each c layer can be tested by testing the closest point to $c\mathbb{R}$



Bödi, Herr, Joswig, 2013

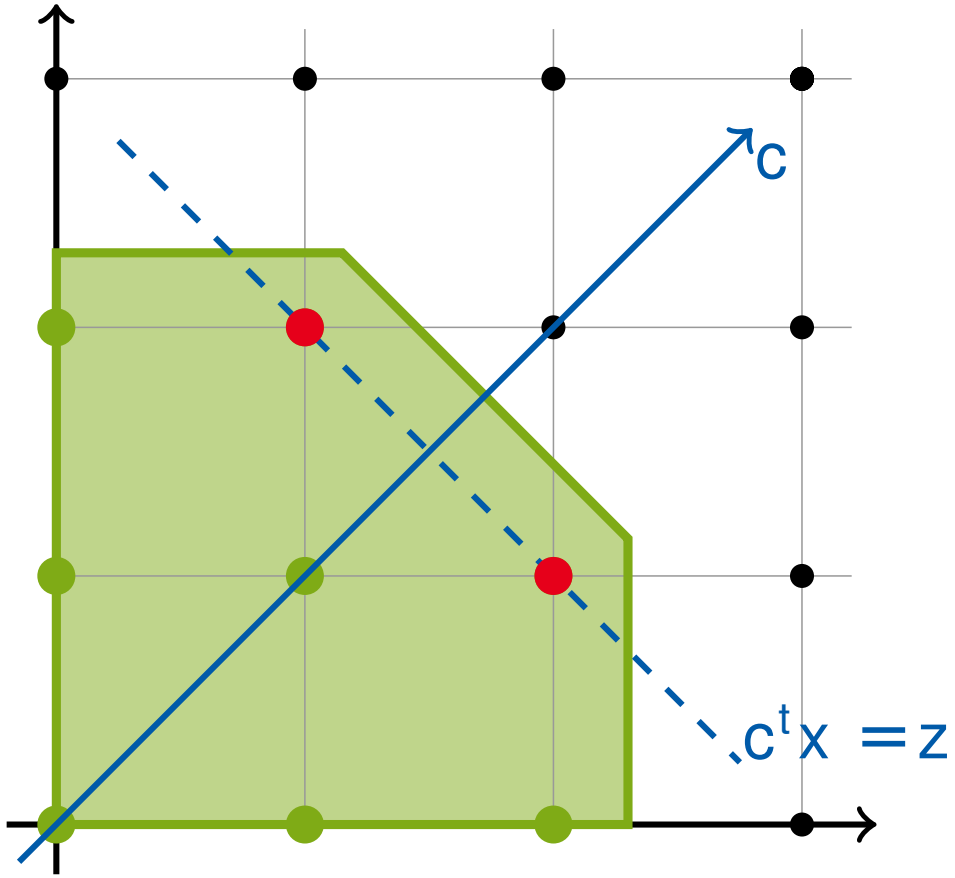
Integer Programming: Core Set Methods

c-layer

Hyperplane orthogonal to c containing integer points.

For sufficiently small number of orbits on the elementary vectors

- The number c-layers can be bounded by the dimension
- Each c layer can be tested by testing the closest point to $c\mathbb{R}$



Bödi, Herr, Joswig, 2013

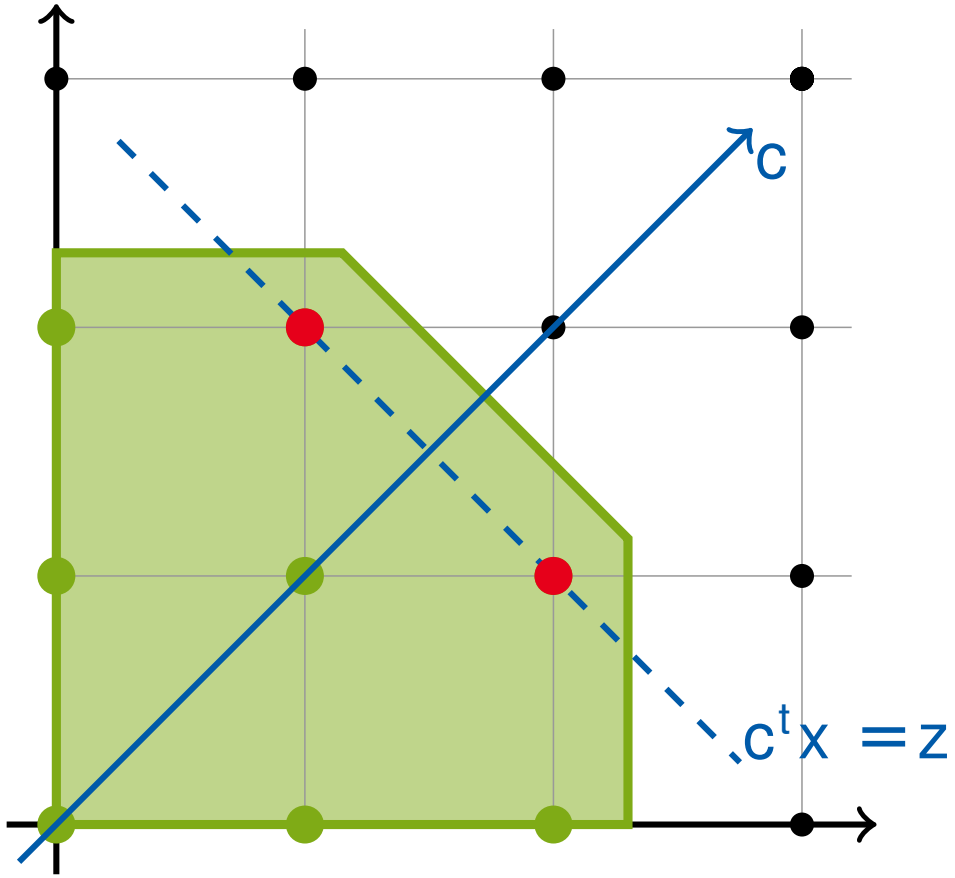
Integer Programming: Core Set Methods

c-layer

Hyperplane orthogonal to c containing integer points.

For sufficiently small number of orbits on the elementary vectors

- The number c-layers can be bounded by the dimension
- Each c layer can be tested by testing the closest point to $c\mathbb{R}$



Bödi, Herr, Joswig, 2013

Symmetric Problems

Combinatorial Optimization

- metric polytope 8 (dimension 28)
- 1 orbit of 336 facets
- 1550825600 vertices in 533 orbits

Lattice Packings

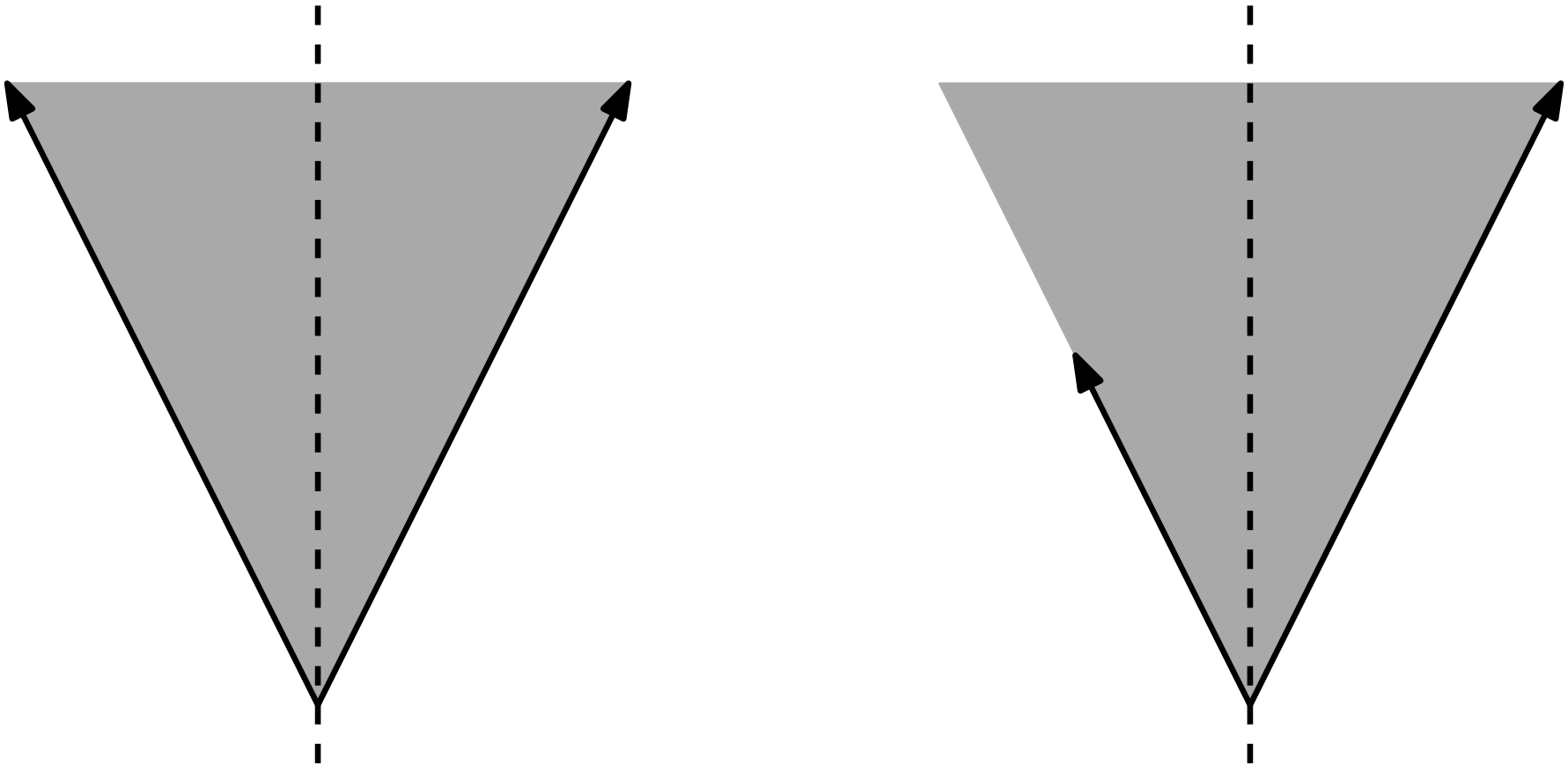
- Λ_9 (dimension 9)
- 49874 vertices in 3? orbits.

Outline

Linear Symmetry Group

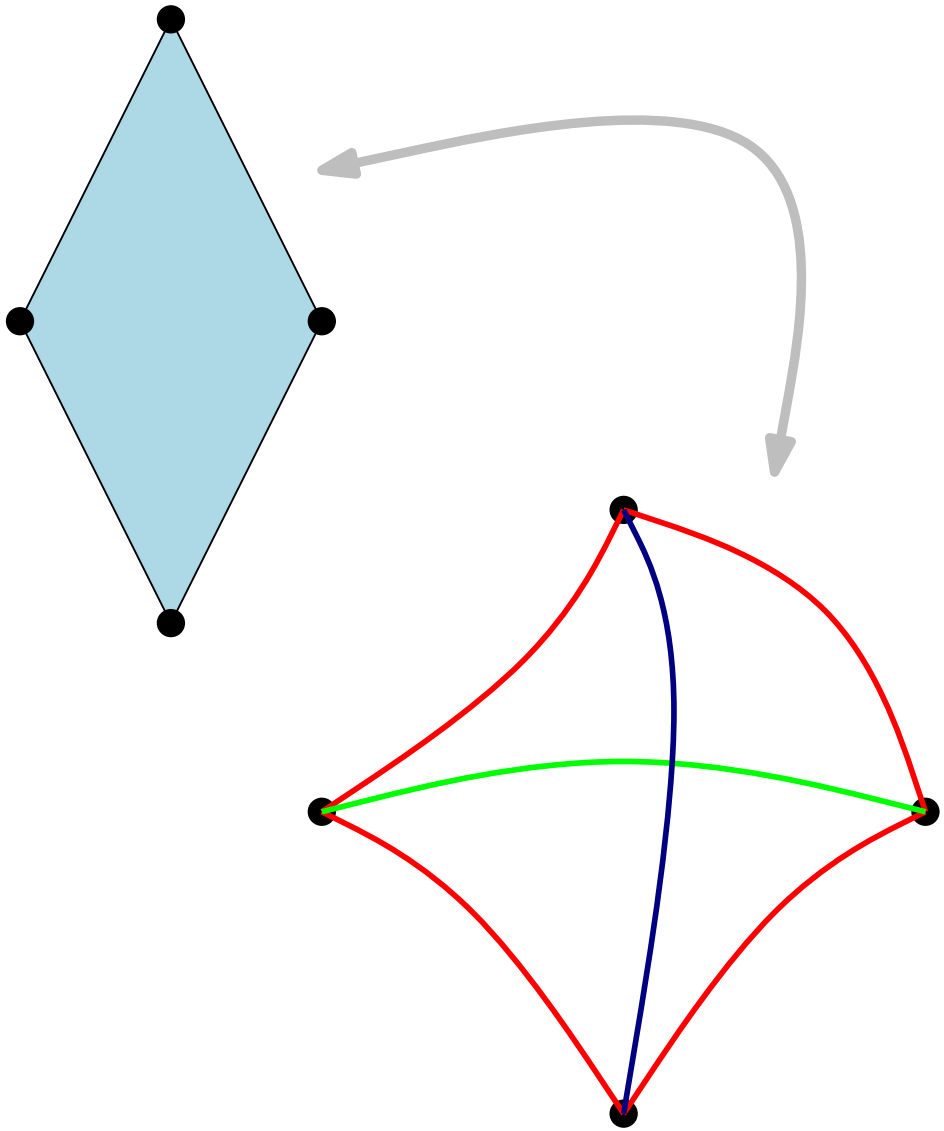
Let $C = \mathrm{pos}(v_1, v_2, v_3, \ldots v_p)$

$$\mathrm{Lin}_v(C) = \left\{ \sigma \in \mathrm{Sym}(p) \mid \exists A \in \mathrm{GL}_n(\mathbb{R}), Av_i = v_{\sigma(i)} \right\}$$



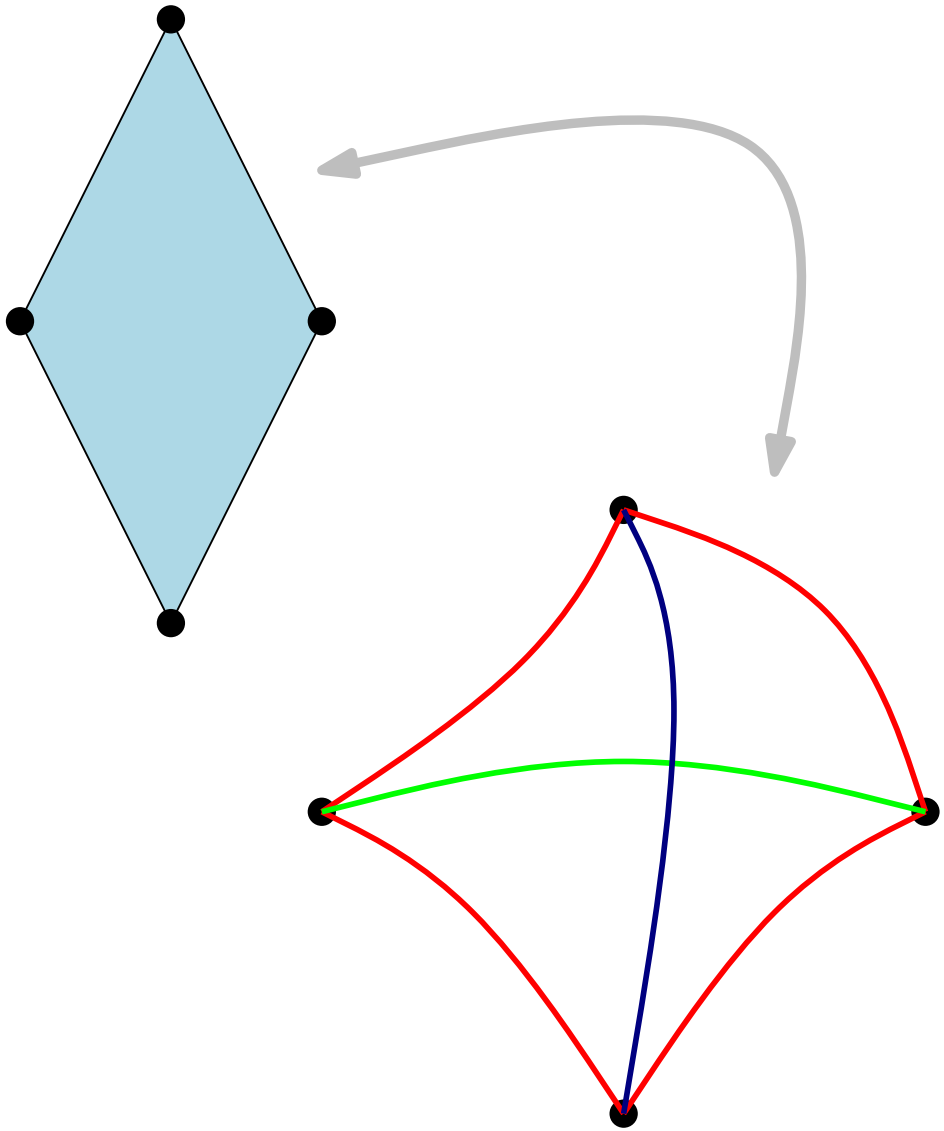
Edge Coloured Graphs

- Consider finding isometries (distance preserving transforms) of affine point sets.
- points become nodes in a graph
- Edges are coloured according to distance.
- Now we "just" need to find the automorphism group of an edge coloured graph.



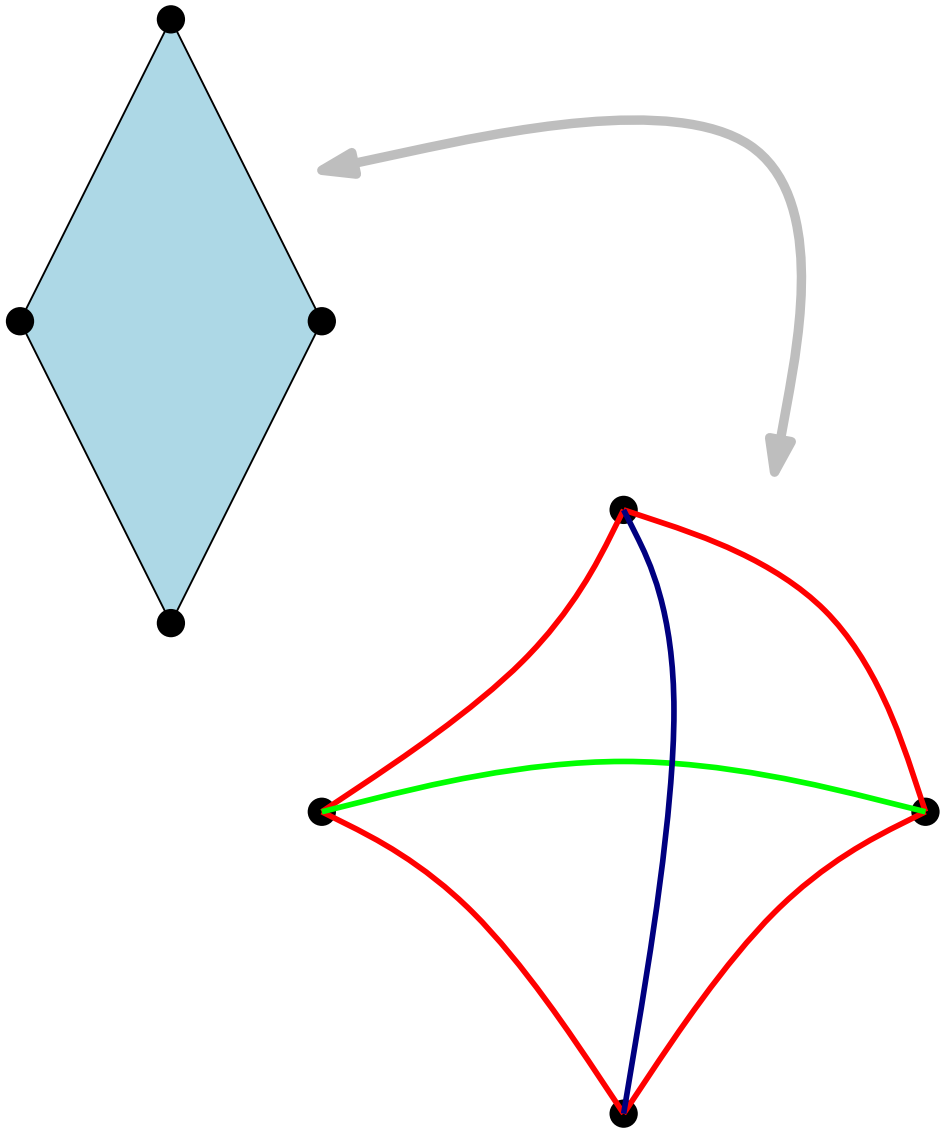
Edge Coloured Graphs

- Consider finding isometries (distance preserving transforms) of affine point sets.
- points become nodes in a graph
- Edges are coloured according to distance.
- Now we "just" need to find the automorphism group of an edge coloured graph.



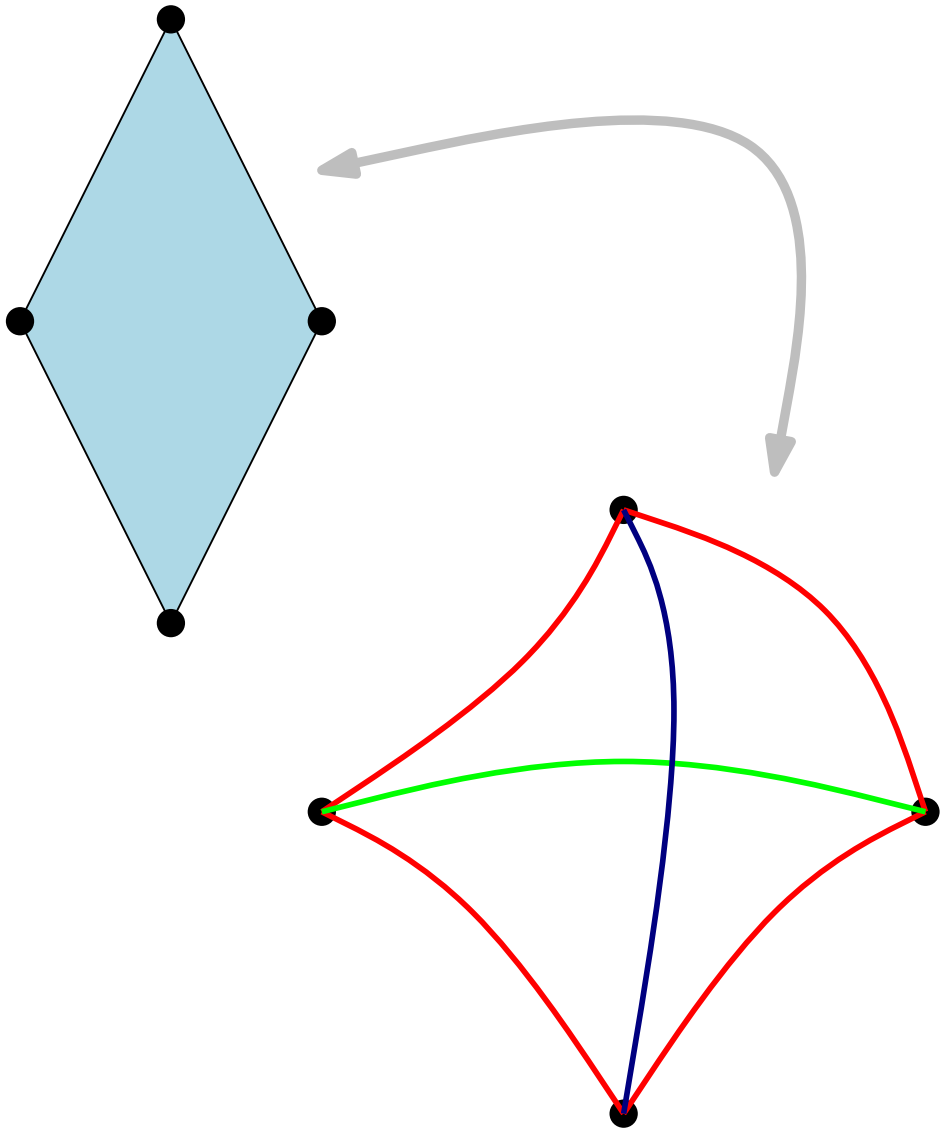
Edge Coloured Graphs

- Consider finding isometries (distance preserving transforms) of affine point sets.
- points become nodes in a graph
- Edges are coloured according to distance.
- Now we "just" need to find the automorphism group of an edge coloured graph.



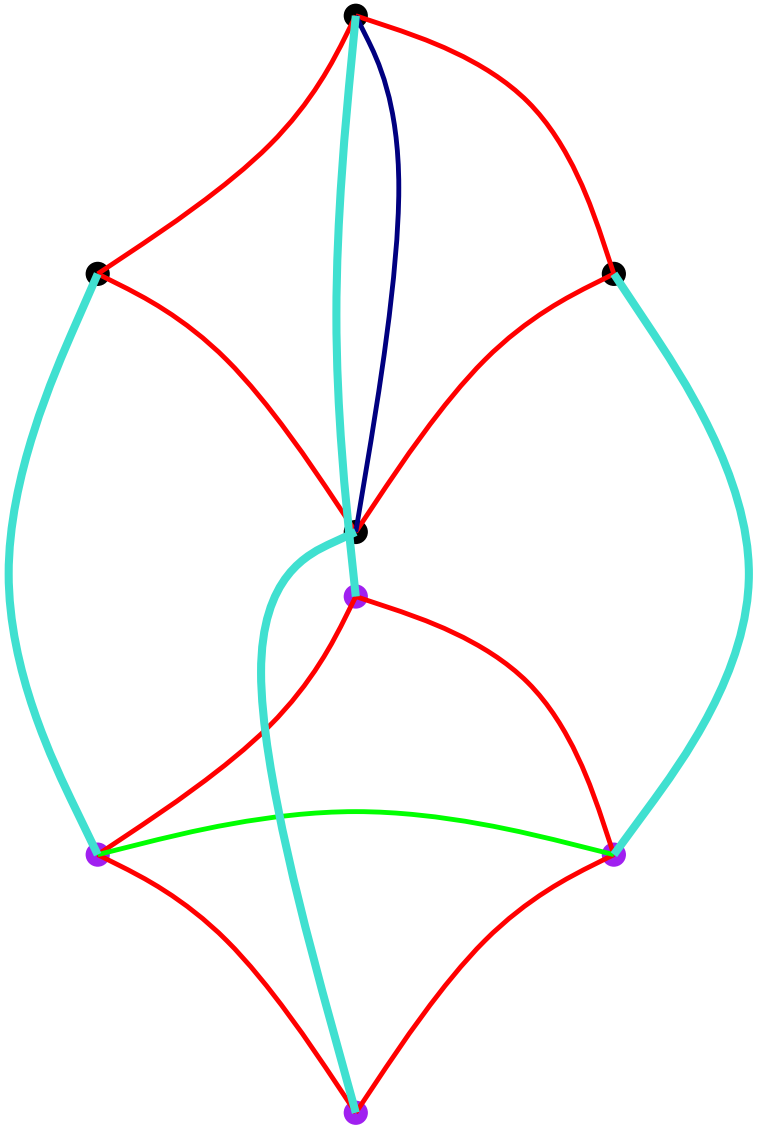
Edge Coloured Graphs

- Consider finding isometries (distance preserving transforms) of affine point sets.
- points become nodes in a graph
- Edges are coloured according to distance.
- Now we "just" need to find the automorphism group of an edge coloured graph.



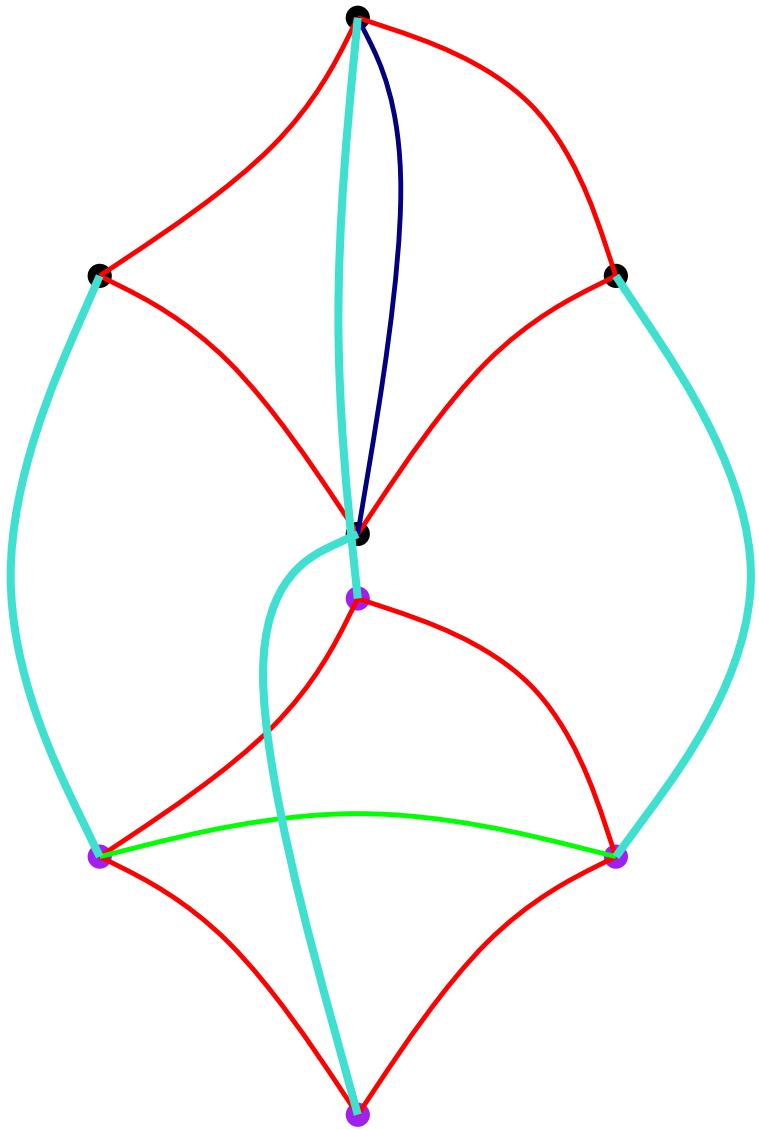
Finding Graph Automorphisms

- Finding automorphism (symmetries) of *vertex* coloured graphs is practical for moderate sized instances using partition backtracking software like *nauty*, *bliss*, and *permlib*
- To encode k edge colours, use $\lceil \log k \rceil$ copies of the graph, and edges in each layer as "bits".



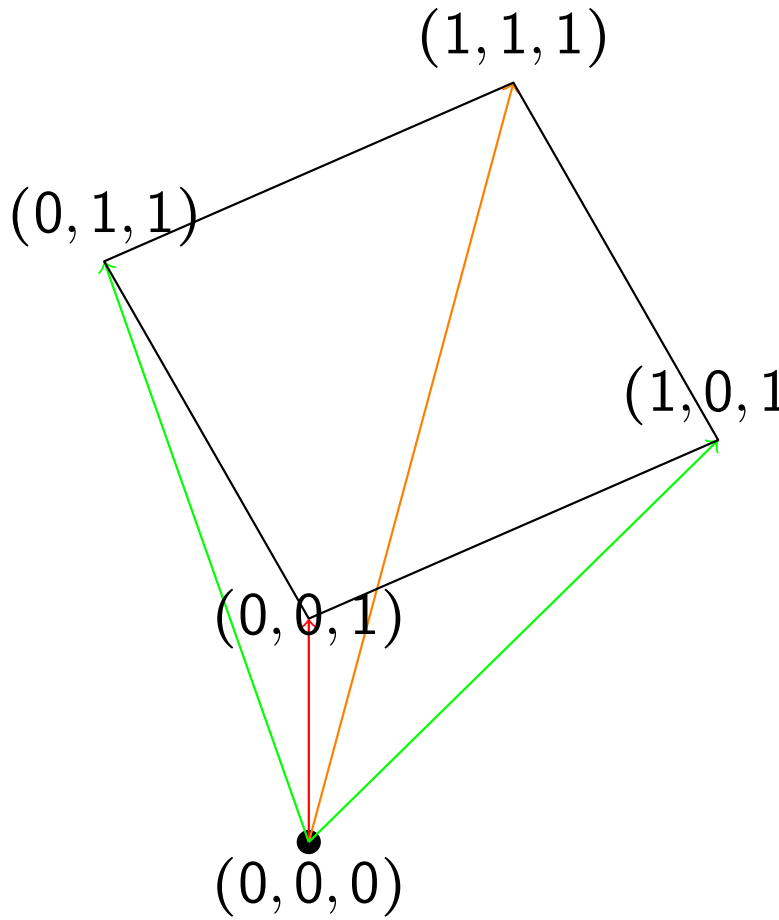
Finding Graph Automorphisms

- Finding automorphism (symmetries) of *vertex* coloured graphs is practical for moderate sized instances using partition backtracking software like *nauty*, *bliss*, and *permlib*
- To encode k edge colours, use $\lceil \log k \rceil$ copies of the graph, and edges in each layer as "bits".



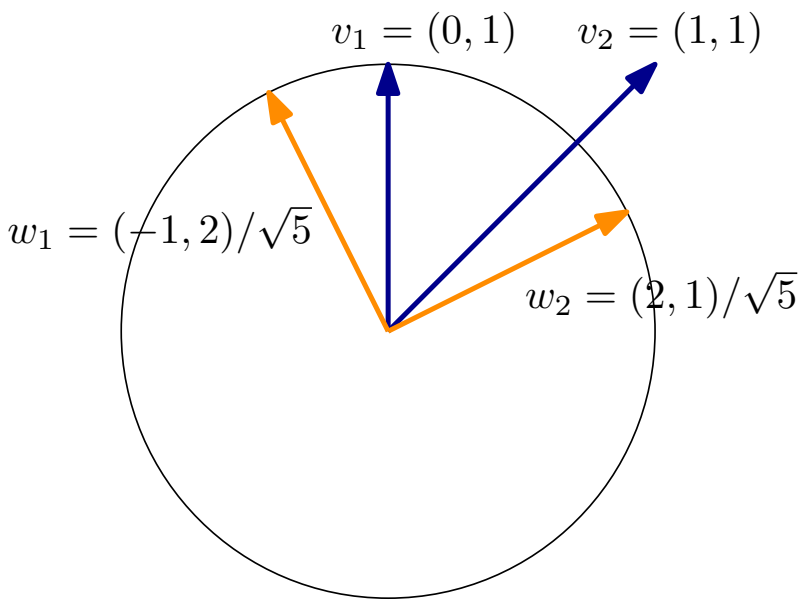
From affine to linear symmetries

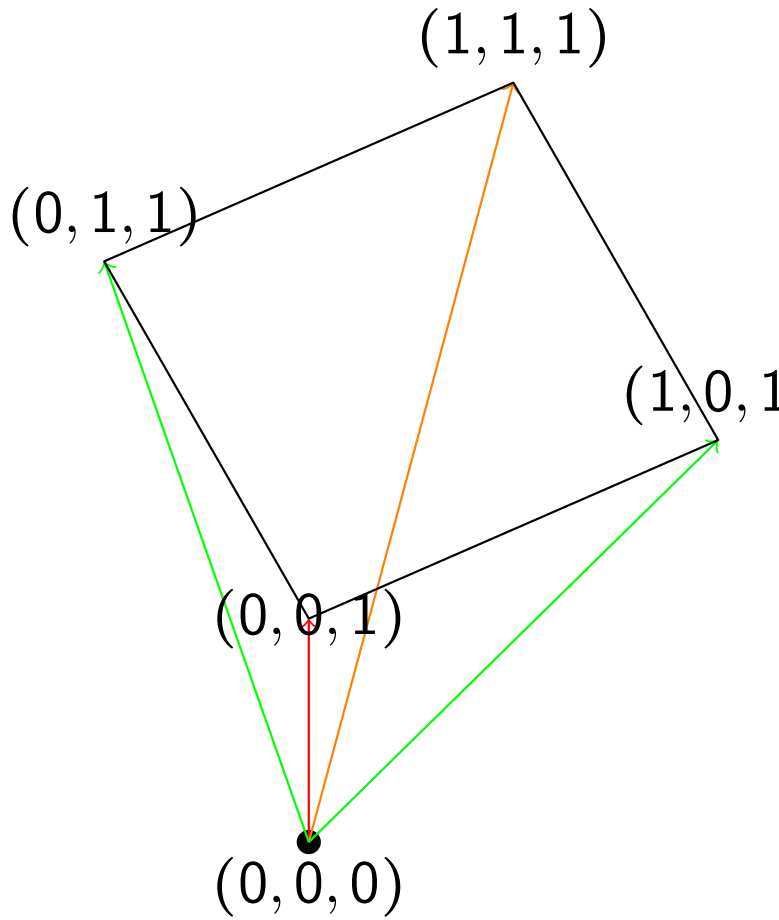
- Our naive approach is insufficient, e.g. when dealing with non centrally symmetric (lifted) polytopes.
- We can do better by normalizing the input



From affine to linear symmetries

- Our naive approach is insufficient, e.g. when dealing with non centrally symmetric (lifted) polytopes.
- We can do better by normalizing the input





Finding linear symmetries

Proposition (BDS09)

Given generators v_i for cone P, $\mathrm{Lin}_v(P)$ is the automorphism group of an edge coloured graph with colours $c_{ij} = v_i^T Q^{-1} v_j$ where $Q = \sum_{i=1}^n v_i v_i^t$.

$$Q^{-1} = R^T R$$
$$w_i = R v_i$$
$$v_i^T Q^{-1} v_j = \langle w_i, w_j \rangle$$

Finding linear symmetries

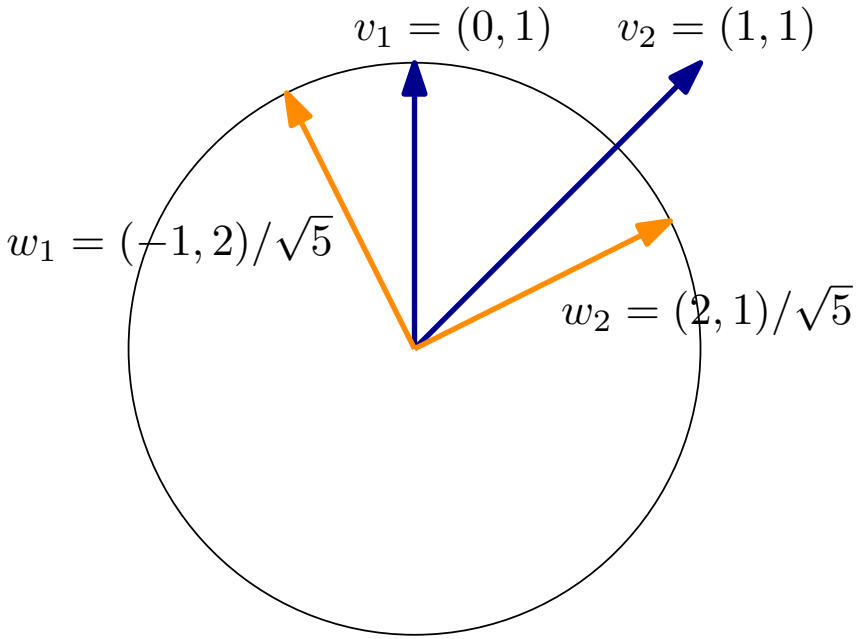
Proposition (BDS09)

Given generators v_i for cone P, $\mathrm{Lin}_v(P)$ is the automorphism group of an edge coloured graph with colours $c_{ij} = v_i^T Q^{-1} v_j$ where $Q = \sum_{i=1}^n v_i v_i^t$.

$$Q^{-1} = R^T R$$

$$w_i = R v_i$$

$$v_i^T Q^{-1} v_j = \langle w_i, w_j \rangle$$

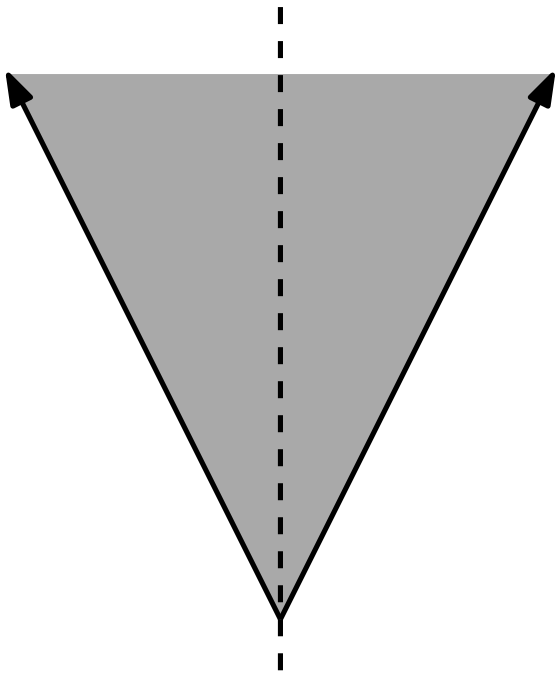


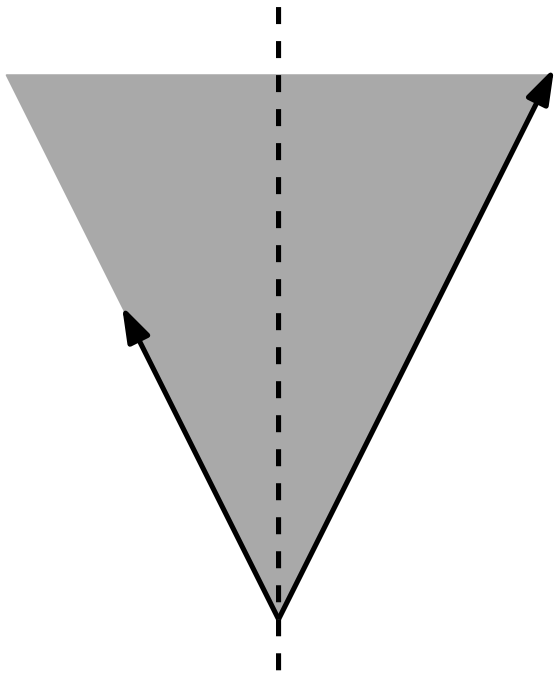
Projective Symmetries

Let $C = \text{pos}(v_1, v_2, v_3, \ldots v_p)$, $R_i = \mathbb{R}_{\geq 0} v_i$

$$\text{GL}(C) = \{ A \in \text{GL}_n(\mathbb{R}) \mid AC = C \}$$

$$\text{Proj}(C) = \{ \sigma \in \text{Sym}(p) \mid \exists A \in \text{GL}(C), AR_i = R_{\sigma(i)} \}$$



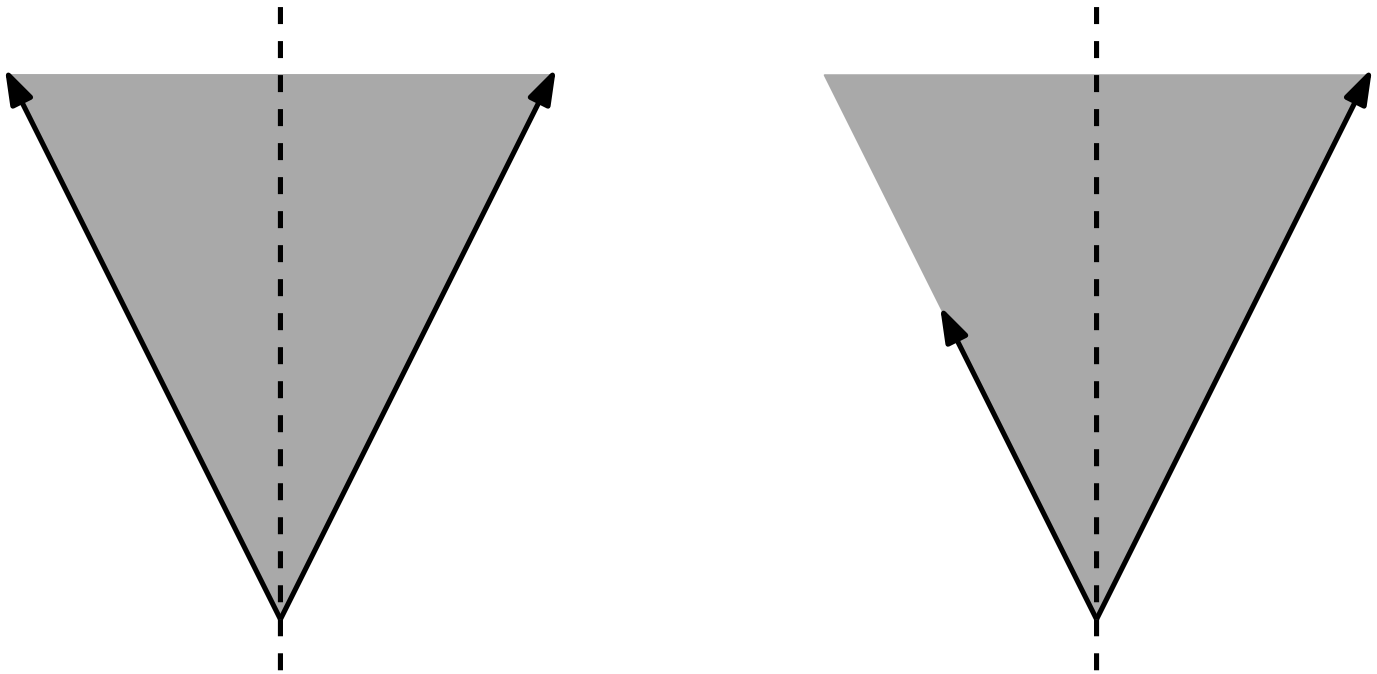


Projective Symmetries

Let $C = \text{pos}(v_1, v_2, v_3, \ldots v_p)$, $R_i = \mathbb{R}_{\geq 0} v_i$

$$\text{GL}(C) = \{ A \in \text{GL}_n(\mathbb{R}) \mid AC = C \}$$

$$\text{Proj}(C) = \{ \sigma \in \text{Sym}(p) \mid \exists A \in \text{GL}(C), AR_i = R_{\sigma(i)} \}$$

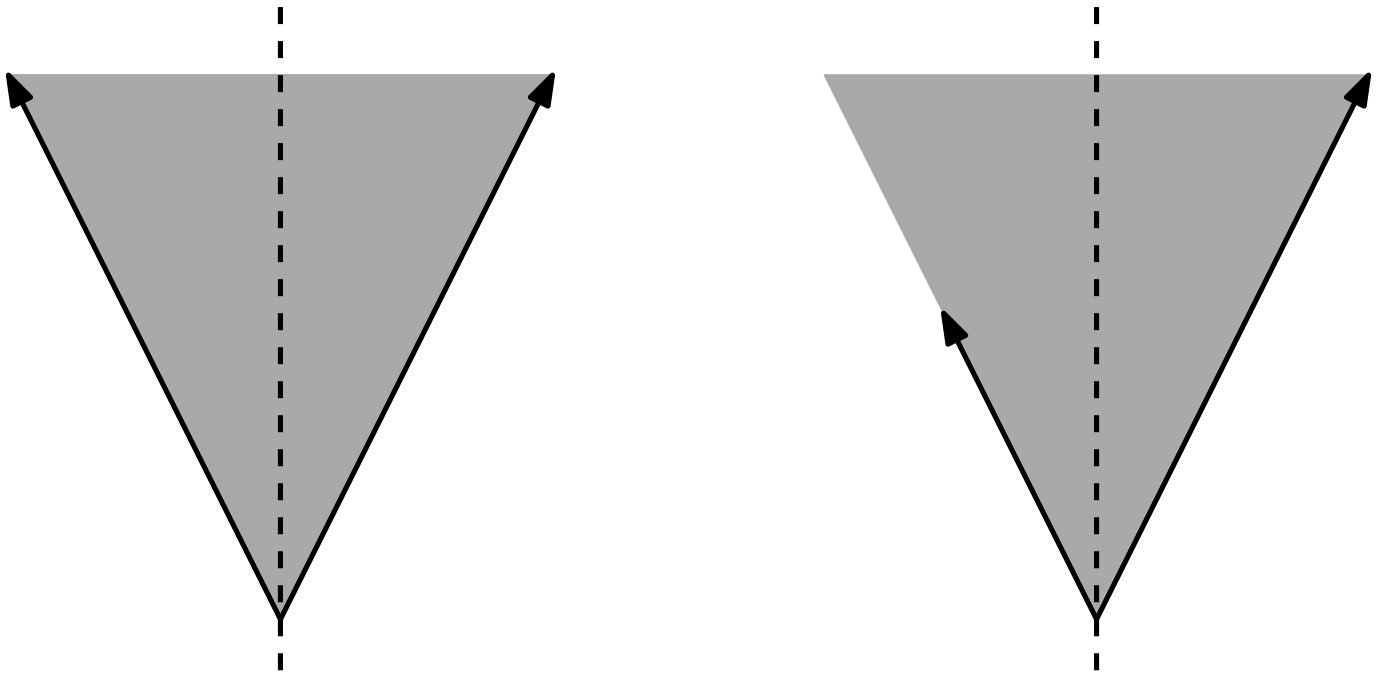


Projective Symmetries

Let $C = \text{pos}(v_1, v_2, v_3, \ldots v_p)$, $R_i = \mathbb{R}_{\geq 0} v_i$

$$\text{GL}(C) = \{ A \in \text{GL}_n(\mathbb{R}) \mid AC = C \}$$

$$\text{Proj}(C) = \left\{ \sigma \in \text{Sym}(p) \mid \exists A \in \text{GL}(C), AR_i = R_{\sigma(i)} \right\}$$

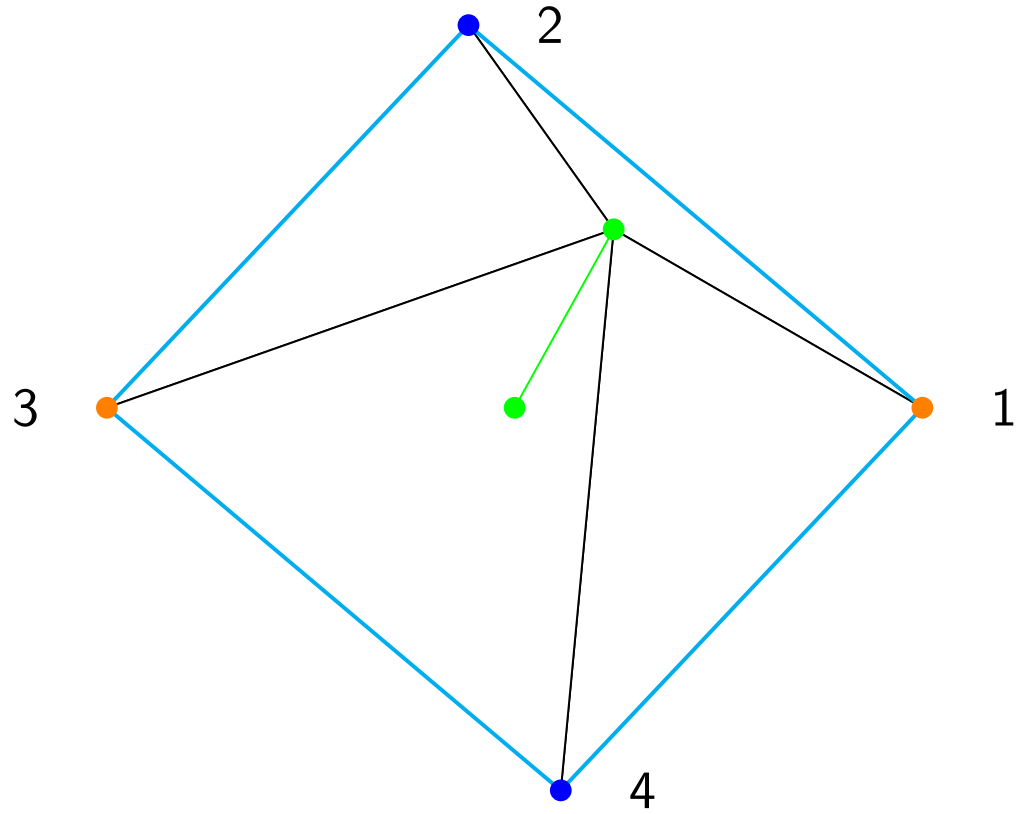


Decomposing Cones

Proposition (BDPRS14)

Given rays $R \subseteq \mathbb{R}^n$ there is a unique decomposition $\mathbb{R}^n = \oplus_{1 \le i \le p} X_i$ with $R = \cup_{1 \le i \le p} R \cap X_i$ and the cones $\mathrm{pos}(R \cap X_i)$ being non-decomposable.

1	1	0	0
1	0	1	0
1	−1	0	0
1	0	−1	0
1	0	0	1



In time $O\left(pn^2\right)$, we can

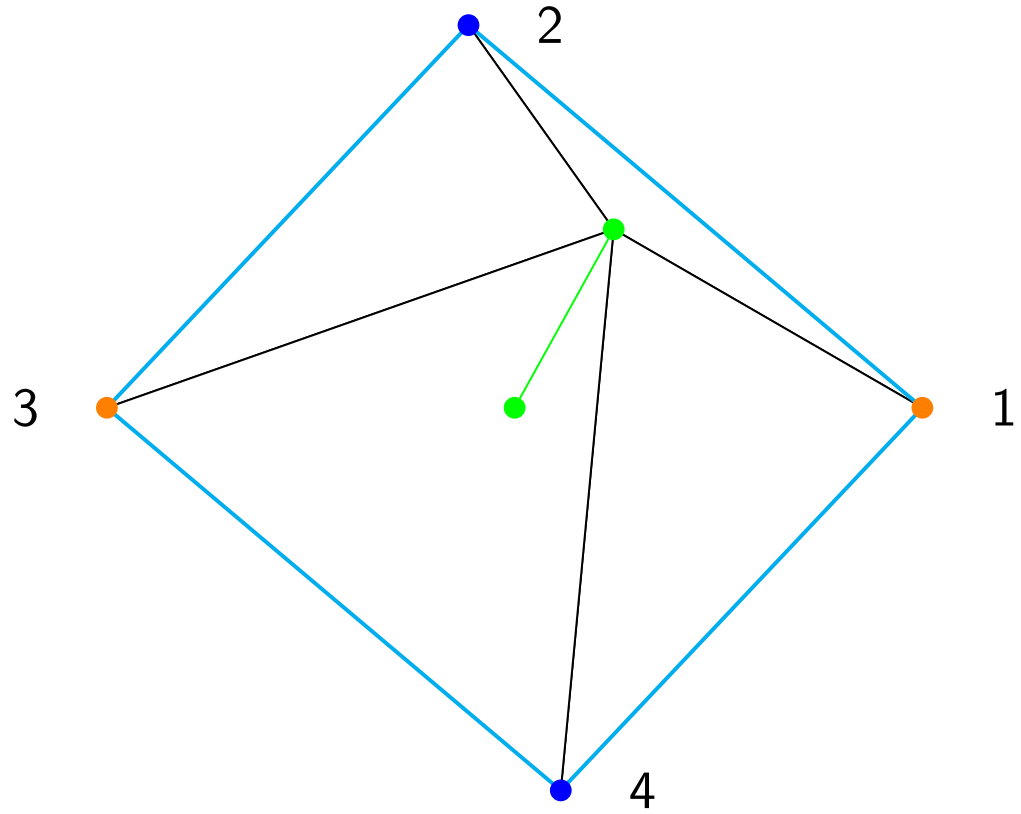
- Find vector generators
- Find a basis
- Analyze support

Decomposing Cones

Proposition (BDPRS14)

Given rays $R \subseteq \mathbb{R}^n$ there is a unique decomposition $\mathbb{R}^n = \oplus_{1 \le i \le p} X_i$ with $R = \cup_{1 \le i \le p} R \cap X_i$ and the cones $\mathrm{pos}(R \cap X_i)$ being non-decomposable.

1	1	0	0
1	0	1	0
1	−1	0	0
1	0	−1	0
1	0	0	1



In time $O\left(pn^2\right)$, we can

- Find vector generators
- Find a basis
- Analyze support

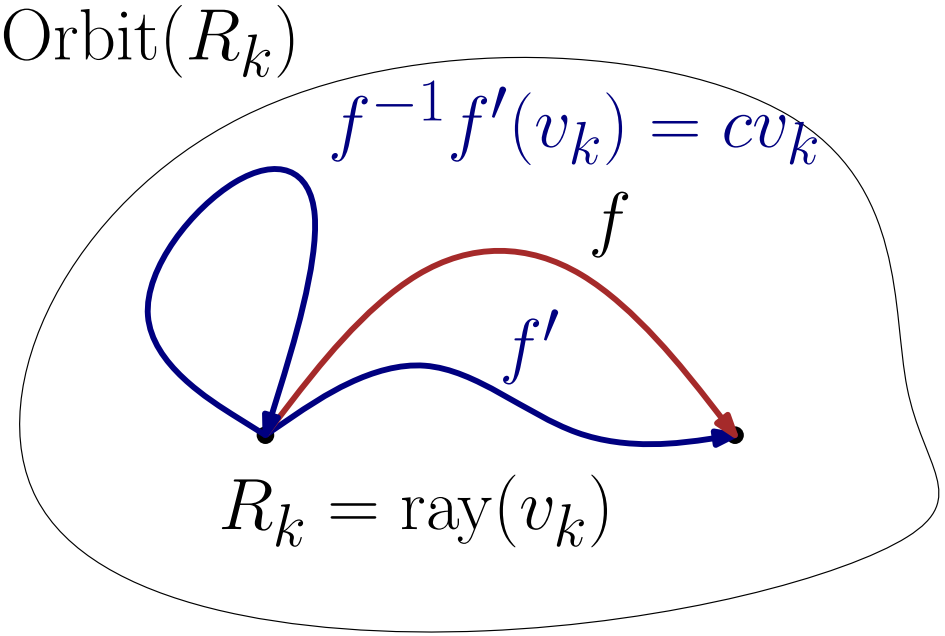
Projective groups for non-decomposible cones

Theorem

Suppose $\mathcal{C}$ is a non-decomposable polyhedral cone generated by rays $\{R_i \mid 1 \le i \le p\}$. Then there exist vectors v_i such that $R_i = \mathbb{R}_+ v_i$ and

$$\mathsf{Lin}_v(\mathcal{C}) = \mathsf{Proj}(\mathcal{C})\,.$$

- Identify $\mathsf{Proj}(C)$ with the finite order elements ($\det(g) = 1$) of $\mathsf{GL}(C)$.
- Choosing one vector generator then uniquely defines the whole orbit.



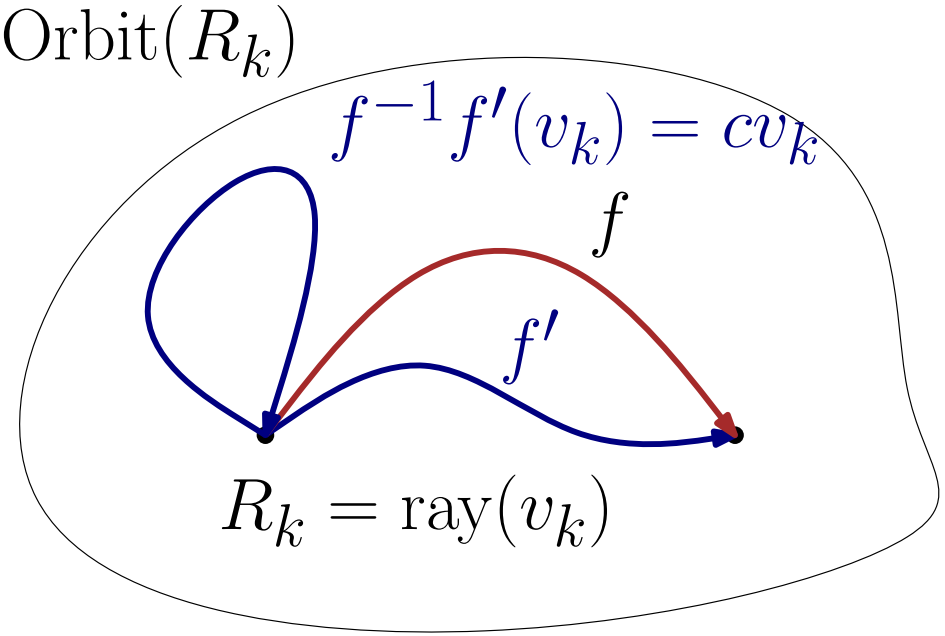
Projective groups for non-decomposible cones

Theorem

Suppose $\mathcal{C}$ is a non-decomposable polyhedral cone generated by rays $\{R_i \mid 1 \le i \le p\}$. Then there exist vectors v_i such that $R_i = \mathbb{R}_+ v_i$ and

$$\mathsf{Lin}_v(\mathcal{C}) = \mathsf{Proj}(\mathcal{C})\,.$$

- Identify $\mathsf{Proj}(C)$ with the finite order elements ($\det(g) = 1$) of $\mathsf{GL}(C)$.
- Choosing one vector generator then uniquely defines the whole orbit.



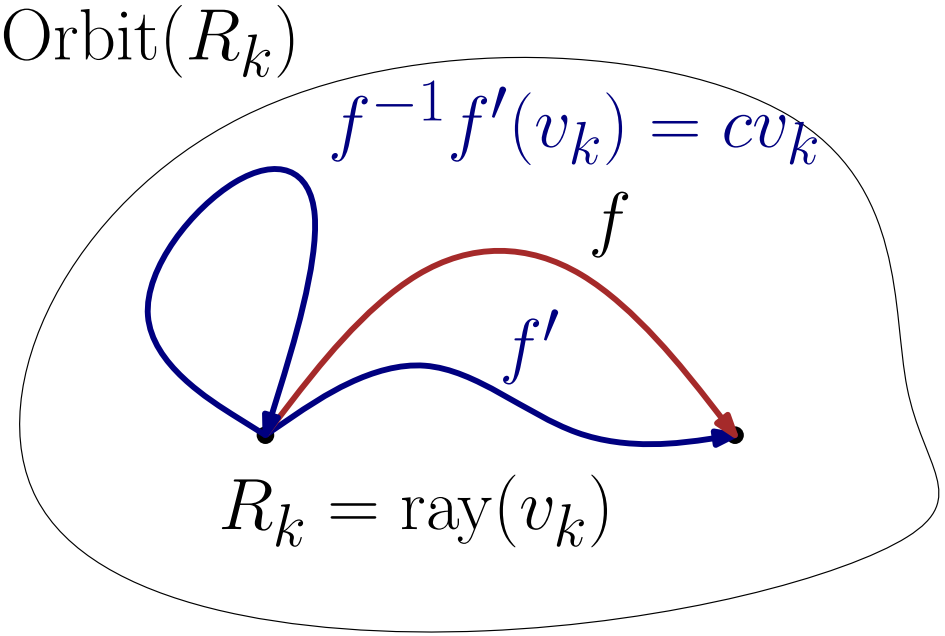
Projective groups for non-decomposible cones

Theorem

Suppose $\mathcal{C}$ is a non-decomposable polyhedral cone generated by rays $\{R_i \mid 1 \le i \le p\}$. Then there exist vectors v_i such that $R_i = \mathbb{R}_+ v_i$ and

$$\mathsf{Lin}_v(\mathcal{C}) = \mathsf{Proj}(\mathcal{C})\,.$$

- Identify $\mathsf{Proj}(C)$ with the finite order elements ($\det(g) = 1$) of $\mathsf{GL}(C)$.
- Choosing one vector generator then uniquely defines the whole orbit.



Putting the projective groups together

Theorem

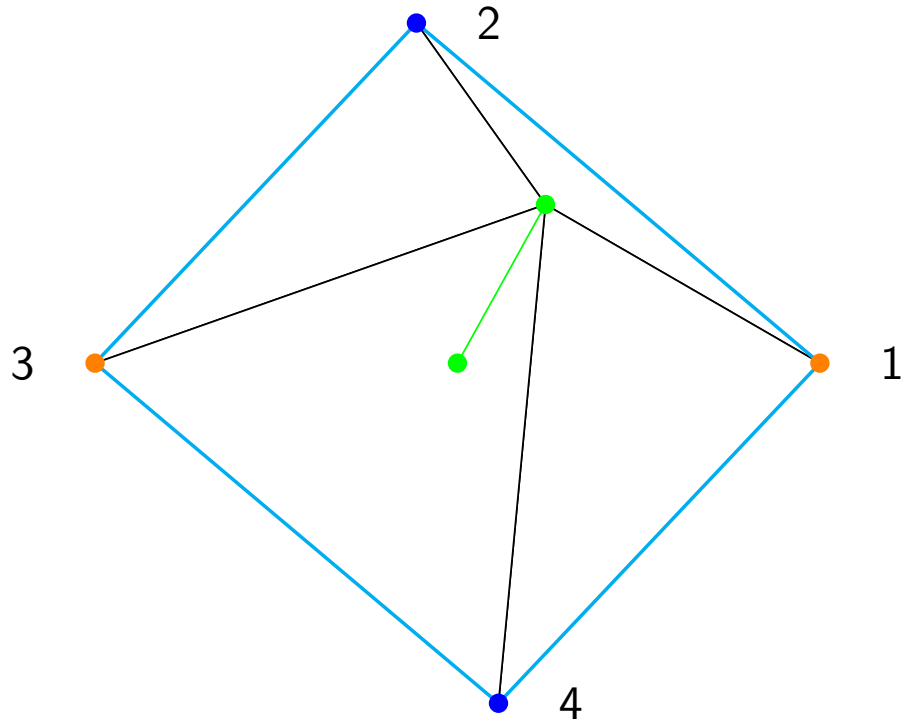
Let $C_1 \dots C_h$ be the decomposition above for cone C. Let $\hat{C}_k$ be a representative of the k-th projective type obtained, and let m_k be its multiplicity.

$$\mathrm{Proj}(C) = \prod \mathrm{Proj}(\hat{C}_k) \wr \mathrm{Sym}(m_k)$$

wreath product

$G \wr H$ captures the idea of permuting the C_j and acting within them at the same time.

$$\begin{aligned}\mathrm{Proj}(C) &= \mathrm{Sym}(2) \wr \mathrm{Sym}(2) \\ &= \langle (1,3), (2,4), (1,4)(2,3) \rangle \\ &= \mathrm{Lin}(\square)\end{aligned}$$



Putting the projective groups together

Theorem

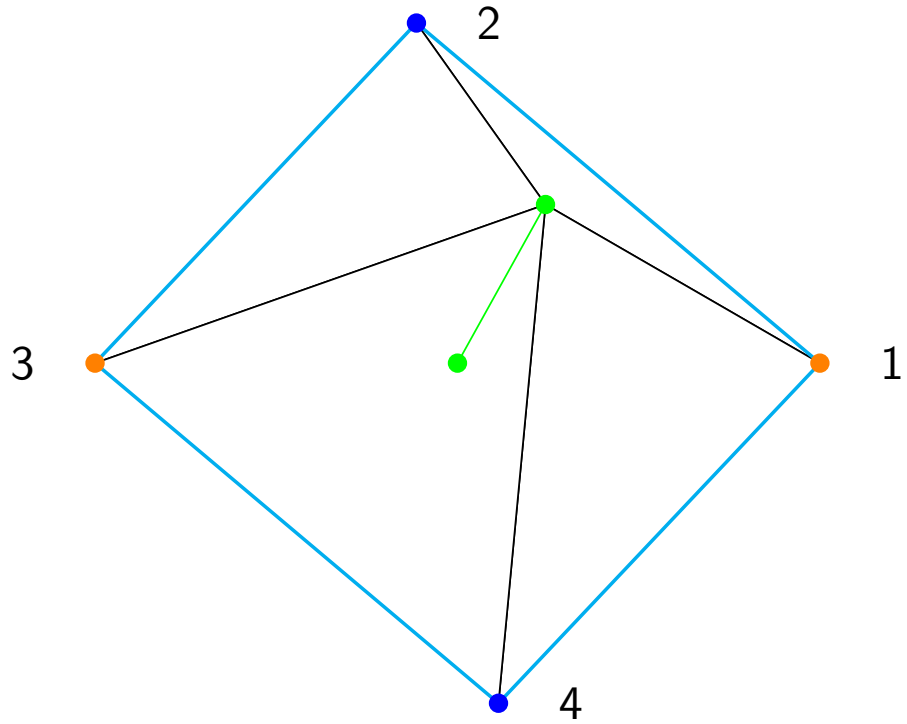
Let $C_1 \dots C_h$ be the decomposition above for cone C. Let $\hat{C}_k$ be a representative of the k-th projective type obtained, and let m_k be its multiplicity.

$$\mathrm{Proj}(C) = \prod \mathrm{Proj}(\hat{C}_k) \wr \mathrm{Sym}(m_k)$$

wreath product

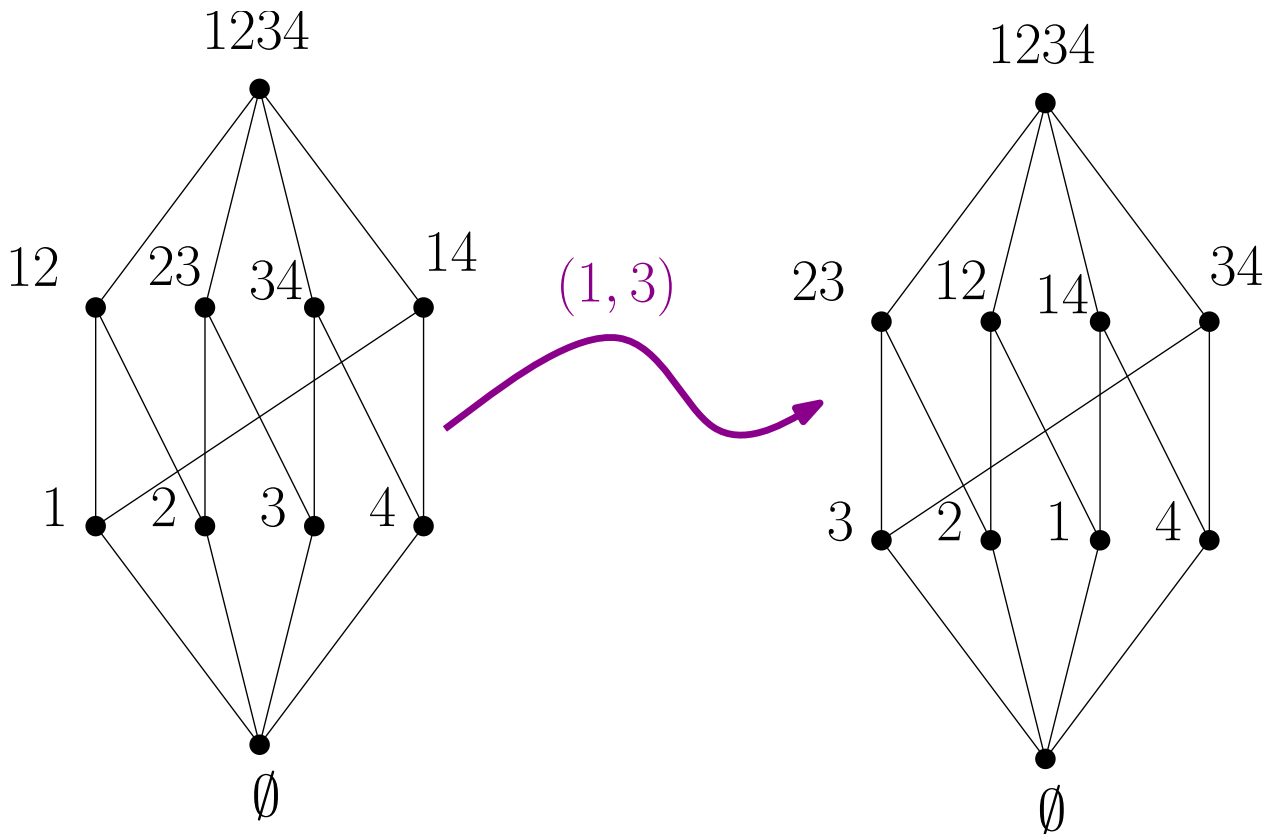
$G \wr H$ captures the idea of permuting the C_j and acting within them at the same time.

$$\begin{aligned}\mathrm{Proj}(C) &= \mathrm{Sym}(2) \wr \mathrm{Sym}(2) \\ &= \langle (1,3), (2,4), (1,4)(2,3) \rangle \\ &= \mathrm{Lin}(\square)\end{aligned}$$



Combinatorial Symmetries

$$\mathsf{Lin}_v(P) \subseteq \mathsf{Comb}(C) \subseteq \mathsf{Skel}_k(C) \subseteq \mathsf{Skel}_{k-1}(C)$$



Sandwiching Groups

Intermediate Subgroup Computation

Given G_1, G_2 such that $G_1 \subset H \subset G_2$, and an oracle ϕ for H, find H.

- for $g \in G_2$ either $G_1 g G_1 \subset H$ or $G_1 g G_1 \cap H = \varnothing$.
- if $g \in H \setminus G_1$, $G_1 \leftarrow G_1 \cup \{ g \}$, recompute cosets

Double Coset Decomposition

Given $H \subset G$, $a \in G$,

$$HaH := \{ h_1 a h_2 \mid h_1, h_2 \in H \} \qquad \text{(double coset)}$$

$$G = \bigsqcup_{i=1}^{s} H a_i H \qquad \text{(double coset decomposition)}$$

Sandwiching Groups

Intermediate Subgroup Computation

Given G_1, G_2 such that $G_1 \subset H \subset G_2$, and an oracle ϕ for H, find H.

- for $g \in G_2$ either $G_1 g G_1 \subset H$ or $G_1 g G_1 \cap H = \varnothing$.
- if $g \in H \setminus G_1$, $G_1 \leftarrow G_1 \cup \{ g \}$, recompute cosets

Double Coset Decomposition

Given $H \subset G$, $a \in G$,

$$HaH := \{ h_1 a h_2 \mid h_1, h_2 \in H \} \qquad \text{(double coset)}$$

$$G = \bigsqcup_{i=1}^{s} H a_i H \qquad \text{(double coset decomposition)}$$

Symmetries for integer programming

- We want to preserve not only the lattice, but also $\mathbb{Z}^n$
- One necessary condition for matrix symmetry A is that $AQA^T = Q$, for $Q = \sum_{i=1}^{n} v_i v_i^t$.
- This stabilization condition can be added to backtracking algorithms for computing lattice automorphisms
- The intermediate subgroup algorithm is useful here as well.

Software and Experiments

- Implemented in GAP by Dutour Sikirić
  `http://drobilica.irb.hr/~mathieu/Polyhedral/`
- Computed $\mathrm{Lin}_V(P)$, $\mathrm{Proj}(P)$ and $\mathrm{Comb}(P)$ for 4313 polytopes of
  `http://ehrhart.math.fu-berlin.de/people/paffenho/`
- 1 example where $\mathrm{Lin}_v(P) \subsetneq \mathrm{Proj}(P)$
- 74 examples where $\mathrm{Proj}(P) \subsetneq \mathrm{Comb}(P)$

Software and Experiments

- Implemented in GAP by Dutour Sikirić
  `http://drobilica.irb.hr/~mathieu/Polyhedral/`
- Computed $\mathrm{Lin}_V(P)$, $\mathrm{Proj}(P)$ and $\mathrm{Comb}(P)$ for 4313 polytopes of
  `http://ehrhart.math.fu-berlin.de/people/paffenho/`
- 1 example where $\mathrm{Lin}_v(P) \subsetneq \mathrm{Proj}(P)$
- 74 examples where $\mathrm{Proj}(P) \subsetneq \mathrm{Comb}(P)$

stop

Orbital Branching

A symmetric packing problem

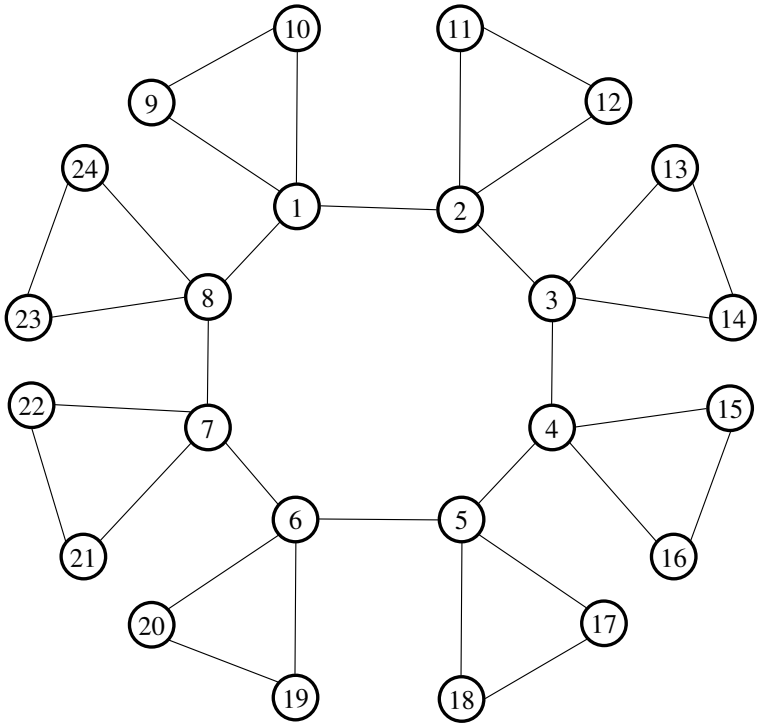
$$\max \sum x_i$$
$$x_i + x_j \leq 1 \quad \forall (i,j) \in E$$
$$x_i \in \{0,1\}$$

Two Orbits of vertices

- $O_1 = \{x_1 \ldots x_8\}$
- $O_2 = \{x_9 \ldots x_{24}\}$

Orbital Branching

- Solution from choosing $x_i = 1$, will be the same, for all $x_i \in O_2$



Example from "Symmetry in Integer Programming", Jim Ostrowski

Orbital Branching

A symmetric packing problem

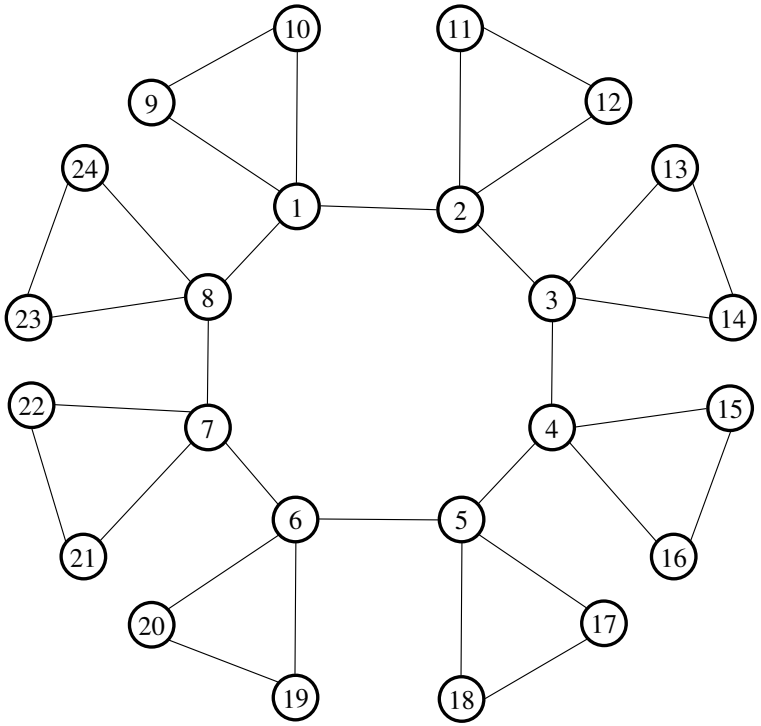
$$\max \sum x_i$$
$$x_i + x_j \leq 1 \quad \forall (i,j) \in E$$
$$x_i \in \{0,1\}$$

Two Orbits of vertices

- $O_1 = \{x_1 \ldots x_8\}$
- $O_2 = \{x_9 \ldots x_{24}\}$

Orbital Branching

- Solution from choosing $x_i = 1$, will be the same, for all $x_i \in O_2$



Example from "Symmetry in Integer Programming", Jim Ostrowski

Orbital Branching

A symmetric packing problem

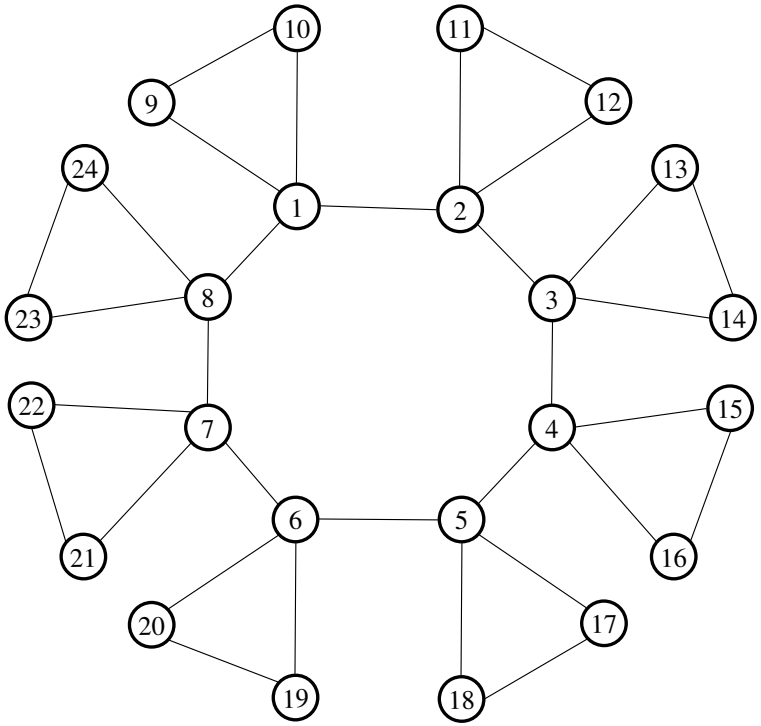
$$\max \sum x_i$$
$$x_i + x_j \leq 1 \quad \forall (i,j) \in E$$
$$x_i \in \{0,1\}$$

Two Orbits of vertices

- $O_1 = \{x_1 \ldots x_8\}$
- $O_2 = \{x_9 \ldots x_{24}\}$

Orbital Branching

- Solution from choosing $x_i = 1$, will be the same, for all $x_i \in O_2$



Example from "Symmetry in Integer Programming", Jim Ostrowski

Orbital Branching

A symmetric packing problem

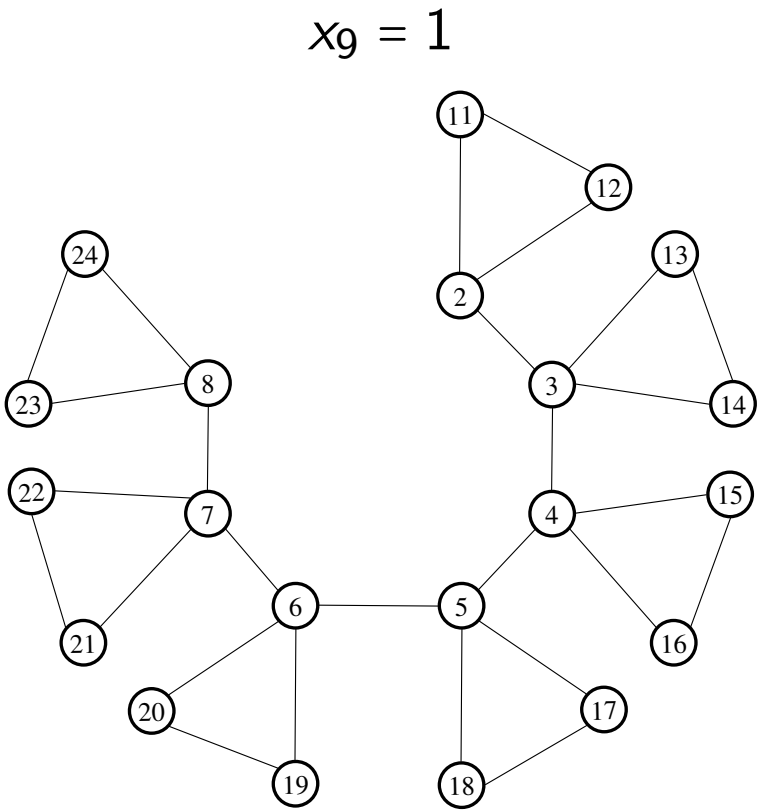
$$\max \sum x_i$$
$$x_i + x_j \leq 1 \quad \forall (i,j) \in E$$
$$x_i \in \{0,1\}$$

Two Orbits of vertices

- $O_1 = \{x_1 \ldots x_8\}$
- $O_2 = \{x_9 \ldots x_{24}\}$

Orbital Branching

- Solution from choosing $x_i = 1$, will be the same, for all $x_i \in O_2$



Example from "Symmetry in Integer Programming", Jim Ostrowski

Orbital Branching

A symmetric packing problem

$$\max \sum x_i$$
$$x_i + x_j \leq 1 \quad \forall (i,j) \in E$$
$$x_i \in \{0,1\}$$

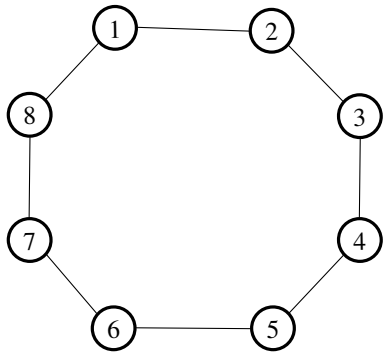
Two Orbits of vertices

- $O_1 = \{x_1 \ldots x_8\}$
- $O_2 = \{x_9 \ldots x_{24}\}$

Orbital Branching

- Solution from choosing $x_i = 1$, will be the same, for all $x_i \in O_2$

$\sum_{i=9}^{26} x_i = 0$



Example from "Symmetry in Integer Programming", Jim Ostrowski