LMS J. Comput. Math. 17 (1) (2014) 565–581
doi:10.1112/S1461157014000400

Computing symmetry groups of polyhedra

David Bremner, Mathieu Dutour Sikirić, Dmitrii V. Pasechnik,
Thomas Rehn and Achill Schürmann

Abstract

Knowing the symmetries of a polyhedron can be very useful for the analysis of its structure as well as for practical polyhedral computations. In this note, we study symmetry groups preserving the linear, projective and combinatorial structure of a polyhedron. In each case we give algorithmic methods to compute the corresponding group and discuss some practical experiences. For practical purposes the linear symmetry group is the most important, as its computation can be directly translated into a graph automorphism problem. We indicate how to compute integral subgroups of the linear symmetry group that are used, for instance, in integer linear programming.

1. *Introduction*

Symmetric polyhedra occur frequently in diverse contexts of mathematics. Polyhedra in general are central to the theory of mathematical optimization (mathematical programming), and the main objects of study in linear and integer linear programming. In applications such as transportation logistics or machine scheduling, symmetric polyhedra are frequently studied, notably the *Travelling Salesman*, *Assignment*, and *Matching* polyhedra. For these and further examples we refer to [**35**] and the numerous references therein. Polyhedra also play prominent roles in other parts of mathematics, for example in algebraic geometry, and in particular in the theory of toric varieties [**6**].

For the analysis of high-dimensional polyhedra it is important to know their symmetries. Furthermore, for many important tasks in polyhedral computations, such as linear and integer linear programming, the representation conversion problem, or volume computations, symmetry exploiting techniques are available (see [**4**, **27**, **37**]). Even commercial optimization software like [**39**, **43**] includes some techniques for symmetry exploitation by now. To a large extent, the methods used depend on the kind of symmetry that is available and how it is presented. For instance, if we know the group of all affine symmetries of a polyhedron coming from a linear programming problem, we can reduce the problem dimension in case the utility function is also invariant under the group. As we discuss in this paper, the group of affine symmetries has the advantage that it can be practically computed using only partial information, for instance, from a description of the polyhedron by linear inequalities.

The main purpose of this paper is to provide a comprehensive overview of existing techniques to practically compute different types of symmetry groups of a polyhedron. We provide a collection of computational recipes for the main polyhedral symmetry groups of interest, which grew out of our own computational experience. Our general approach is the translation of a polyhedral symmetry finding problem into a problem of determining all the combinatorial

Received 21 October 2013; revised 9 July 2014.

2010 Mathematics Subject Classification 20B25, 52B15 (primary).

The authors acknowledge the hospitality of Mathematisches Forschungsinstitut Oberwolfach (MFO) and Hausdorff Research Institute for Mathematics (HIM) in Bonn. Mathieu Dutour Sikirić was supported by the Humboldt Foundation. Dmitrii Pasechnik was supported by Singapore Ministry of Education ARF Tier 2 Grant MOE2011-T2-1-090.

automorphisms of a colored graph. Although these graph automorphism problems are not completely understood from a complexity theoretical point of view (see [**19**]), there exist sophisticated software tools for their practical solution [**40**, **44**, **46**, **47**]. All algorithms explained here are available within the `GAP` package `polyhedral` [**41**]. The group of all affine symmetries, or the *linear symmetry group*, of a polyhedron can also be computed with the `C++` tool `SymPol` [**49**]. Note that similar computational tasks for special classes of lattice polytopes are performed by `PALP` [**22**, **45**], which arose from works on the classification of so-called reflexive polyhedra (see [**20**, **21**]).

The paper is organized as follows. In § 2 we define the three most important polyhedral symmetry groups and explain some of the relations among them. They are: the linear symmetry group, the projective symmetry group and the combinatorial symmetry group. In the following sections we consider each of them separately. We start with the linear symmetry group in § 3, which from a practical point of view is probably the most interesting one. Depending on the context of application, certain subgroups of the linear symmetry group have been shown to be important. In §§ 3.1–3.3 we describe specific computational tools for subgroups that we encountered in integer linear programming and in problems arising in the computational geometry of positive definite quadratic forms. In § 3.4 we provide some practical pointers for the necessary computations with vertex and edge colored graphs. Section 4 deals with the projective symmetry group, which has not received much practical attention so far, as an algorithm for its computation has been missing. Here, as a main theoretical contribution of our paper, we give a new characterization of the projective symmetry group and from it derive an algorithm for its computation. Finally, in § 5, we give practical recipes for the computation of the full combinatorial symmetry group of a polyhedron.

2. *Polyhedral symmetry groups*

In this section we define the basic objects that we study in this article: polyhedral cones and their symmetry groups. We note that our study of polyhedral cones also covers polytopes by the use of homogeneous coordinates, as we explain below.

2.1. *Polyhedral cones*

A *polyhedral cone* $\mathcal{C}$ in the vector space $\mathbb{R}^n$ is defined as the set of vectors satisfying a finite number of linear (homogeneous) inequalities. By the Farkas–Minkowski–Weyl theorem there exists a second (dual) description

$$\mathcal{C} = \{\lambda_1 v_1 + \ldots + \lambda_p v_p \mid \lambda_i \in \mathbb{R}_{\geqslant 0}\}$$

with a minimal set of *generating vectors* $v_1, \ldots, v_p$ and *extreme rays*

$$R_i = \mathbb{R}_{\geqslant 0} v_i.$$

Without loss of generality, we assume that $\mathcal{C}$ is full-dimensional, that is it has dimension n, and that it does not contain a non-trivial vector space. Note that, while the extreme rays of $\mathcal{C}$ are uniquely determined under this assumption, the generating vectors are not.

A *face* of a polyhedral cone $\mathcal{C}$ is the intersection of $\mathcal{C}$ with a *supporting hyperplane*, that is, with a hyperplane $\{x \in \mathbb{R}^n \mid a^T x = 0\}$ for some $a \in \mathbb{R}^n$ such that C is contained in the halfspace $\{x \in \mathbb{R}^n \mid a^T x \geqslant 0\}$. Faces are partially ordered by setwise inclusion, which gives a combinatorial lattice that is called the *face lattice of* $\mathcal{C}$. The *dimension of a face* is defined as the dimension of the smallest linear subspace containing it.

Facets of $\mathcal{C}$ are faces of dimension $n-1$. The set of facets and the set of extreme rays are uniquely determined by $\mathcal{C}$ and the problem of passing from one description to the other is called the *dual description* or *representation conversion problem*.

A *polytope* P is the convex hull of a finite set of vectors $\{v_i \mid 1 \leqslant i \leqslant M\}$ in $\mathbb{R}^n$. By using *homogeneous coordinates* for the v_i, that is the vectors $v_i' = (1, v_i^T)^T$, and considering the polyhedral cone $\mathcal{C}$ defined by $\{v_i' \mid 1 \leqslant i \leqslant M\}$, we can actually embed P into $\mathcal{C}$ and translate the notions introduced for polyhedral cones to polytopes. Note that the passage to homogeneous coordinates gives us a canonical identification of a given polytope P with a uniquely defined polyhedral cone $\mathcal{C} = \mathcal{C}(P)$. For more information and background on polyhedra and polytopes we refer to [**38**].

2.2. *Symmetry groups*

In this section we define three different symmetry groups of polyhedral cones and discuss relations among them.

Combinatorial symmetries. Let $\mathcal{F}_k$ denote the set of k-dimensional faces (k-faces) of $\mathcal{C}$. Such k-faces are identified with the set of extreme rays contained in them.

DEFINITION 1. The *combinatorial symmetry group* $\mathrm{Comb}(\mathcal{C})$ of $\mathcal{C}$ is the group of all permutations of extreme rays that preserve $\mathcal{F}_k$ for all $0 \leqslant k \leqslant n-1$.

In particular, $\mathrm{Comb}(\mathcal{C})$ is a subgroup of the symmetric group $\mathrm{Sym}(p)$ on p elements, where p is the number of extreme rays. The combinatorial symmetry group is precisely the permutation group in $\mathrm{Sym}(p)$ which preserves the face lattice of $\mathcal{C}$.

It is well known that $\mathrm{Comb}(\mathcal{C})$ is actually determined by $\mathcal{F}_{n-1}$ alone: $\mathrm{Comb}(\mathcal{C})$ is isomorphic to the automorphism group of the bipartite facet-ray-incidence graph. Indeed, there is generally no simpler way to compute $\mathrm{Comb}(\mathcal{C})$, as this problem is *graph isomorphism complete* even for simple or simplicial polytopes (see [**17**]). For the construction of this bipartite graph we must know both representations of $\mathcal{C}$, namely the extreme rays and the facets. However, in practice, usually only one of these descriptions is known.

Projective symmetries. Let $\mathrm{GL}(\mathcal{C})$ be the group of invertible matrices $A \in \mathrm{GL}_n(\mathbb{R})$ which preserve the cone $\mathcal{C}$ setwise: $A\mathcal{C} = \mathcal{C}$. These transformations induce a projective symmetry of the cone by permuting its extreme rays.

DEFINITION 2. The *projective symmetry group* $\mathrm{Proj}(\mathcal{C})$ of $\mathcal{C}$ consists of all permutations $\sigma \in \mathrm{Sym}(p)$ such that there exists a matrix $A \in \mathrm{GL}(\mathcal{C})$ with $AR_i = R_{\sigma(i)}$ for $1 \leqslant i \leqslant p$.

It is easy to see that the projective symmetry group is a subgroup of the combinatorial symmetry group. Note, however, that $\mathrm{GL}(\mathcal{C})$ and $\mathrm{Proj}(\mathcal{C})$ are not isomorphic since the kernel K of the homomorphism $\mathrm{GL}(\mathcal{C}) \to \mathrm{Proj}(\mathcal{C})$ is non-trivial. It contains, for instance, all dilations. In group-theoretic terms, $\mathrm{GL}(\mathcal{C}) \cong K \times \mathrm{Proj}(\mathcal{C})$ (cf. Theorem 8).

We note that in a more general setting of symmetries of a configuration of points in a projective space over a field (for example over $\mathbb{C}$), we can define $\mathrm{GL}(\mathcal{C})$ and $\mathrm{Proj}(\mathcal{C})$ as above, but K does not need to be split from $\mathrm{GL}(\mathcal{C})$. See Remark 9 for a simple example of the latter, and [**15**, **16**] for the related group-theoretic notions.

Linear symmetries. Let $v = \{v_i \mid 1 \leqslant i \leqslant p\}$ be a set of generators for the extreme rays $\{R_i \mid 1 \leqslant i \leqslant p\}$ of a polyhedral cone $\mathcal{C}$.

DEFINITION 3. The *linear symmetry group* $\mathrm{Lin}_v(\mathcal{C})$ of $\mathcal{C}$ (with respect to v) is the set of all permutations $\sigma \in \mathrm{Sym}(p)$ such that there exists a matrix $A \in \mathrm{GL}_n(\mathbb{R})$ where $Av_i = v_{\sigma(i)}$ for $1 \leqslant i \leqslant p$.

The group defined by such matrices A is denoted by $\mathrm{GL}_v(\mathcal{C})$ and is isomorphic to $\mathrm{Lin}_v(\mathcal{C})$.

Symmetries of polytopes. In the particular case of a polytope P in $\mathbb{R}^n$ with an associated cone $\mathcal{C} = \mathcal{C}(P)$ in $\mathbb{R}^{n+1}$ generated by $\{v_i' = (1, v_i^T)^T \mid 1 \leqslant i \leqslant p\}$ any $A \in \mathrm{GL}_{v'}(\mathcal{C})$ is of the form

$$A = \begin{pmatrix} 1 & 0 \\ b & B \end{pmatrix},$$

that is, it describes an affine transformation $x \mapsto Bx + b$ of $\mathbb{R}^n$ preserving P. If we speak of the *linear symmetry group of the polytope* P we mean the group $\mathrm{Lin}_{v'}(\mathcal{C})$ for the specific set of generators v' obtained from the coordinates of vertices v of P. This group is isomorphic to the group of all affine transformations of $\mathbb{R}^n$ that preserve P. If we speak of the *projective symmetry*, respectively *combinatorial symmetry group*, of the polytope P we simply mean $\mathrm{Proj}(\mathcal{C})$, respectively $\mathrm{Comb}(\mathcal{C})$.

Relations and differences among symmetries. For every polyhedral cone $\mathcal{C}$ and every set of generators v we have the subgroup relations

$$\mathrm{Lin}_v(\mathcal{C}) \leqslant \mathrm{Proj}(\mathcal{C}) \leqslant \mathrm{Comb}(\mathcal{C}).$$

Both inclusions can be strict as the example of Figure 1 shows. Further, for each of the three symmetry groups we can define a corresponding notion of equivalence. Two cones $\mathcal{C}$ and $\mathcal{C}'$ with generators v and v' are (linearly, projectively, combinatorially) equivalent if there is a (linear, projective, combinatorial) symmetry mapping $\mathcal{C}$ onto $\mathcal{C}'$, respectively v onto v'.

Remark 1. Our notion of projective equivalence should not be confused with the one used in projective geometry (cf. for example [**5**]). The latter would lead to much coarser equivalence of cones than is meaningful.

Example 1. In Figure 1 we give examples of those notions for the cube. Note that coordinates of vertices are given in $\mathbb{R}^3$; corresponding polyhedral cones are given by homogeneous coordinates in $\mathbb{R}^4$. We note that in $\mathbb{R}^2$ it is well known that any two n-gons are combinatorially equivalent. If $n = 4$ they are also projectively equivalent by Theorem 6 but this does not generalize to $n > 4$.

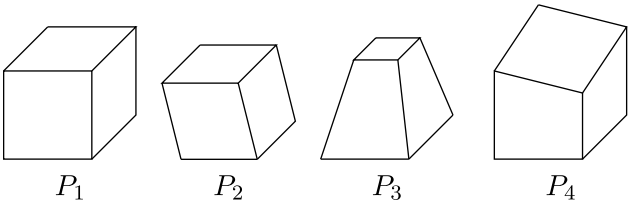
In § 4 we prove that the projective symmetry group $\mathrm{Proj}(\mathcal{C})$ can be realized as $\mathrm{Lin}_v(\mathcal{C})$ for a suitable choice of vectors v. However, in [**3**, **13**, **38**] some polytopes are given whose combinatorial symmetries cannot be realized as projective symmetries.

Using the implementation in [**41**] we have computed the linear, projective and combinatorial symmetry group of 4313 polytopes available from the web page of Paffenholz [**50**]. For these examples, there is only one case where the projective symmetry group is larger than the linear symmetry group. This example was the one obtained by applying the construction E_2 of [**29**] to the 4-simplex; it is projectively equivalent to the dual of the Johnson polytope $J(5, 2)$. For 75 of the 4313 examples, the combinatorial symmetry group is larger than the projective symmetry group. The additional symmetries are in most cases a factor of 2 but in two cases reached a factor of 36.

3. *Computing* $\mathrm{Lin}_v(\mathcal{C})$

In this section we give algorithms to compute the linear symmetry group $\mathrm{Lin}_v(\mathcal{C})$ and certain subgroups occurring in applications like combinatorial optimization.

The linear symmetry group is easier to compute in practice than either the projective or the combinatorial symmetry group; at least in the typical situation where we have only a generator representation (or only an inequality representation) for the input. Furthermore, the computation of the linear group is used as a subroutine in our algorithms to compute the



P_1 P_2 P_3 P_4

$$\operatorname{vert} P_1 = \begin{pmatrix} 2 & 2 & 2 & 2 & -2 & -2 & -2 & -2 \\ 2 & 2 & -2 & -2 & 2 & 2 & -2 & -2 \\ 2 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \end{pmatrix},$$

$$\operatorname{vert} P_2 = \begin{pmatrix} 2 & 2 & 2 & 2 & -2 & -2 & -2 & -2 \\ 1 & 1 & -3 & -3 & 2 & 2 & -2 & -2 \\ 1 & -3 & 1 & -3 & 2 & -2 & 2 & -2 \end{pmatrix},$$

$$\operatorname{vert} P_3 = \begin{pmatrix} 2 & 2 & 2 & 2 & -2 & -2 & -2 & -2 \\ 1 & 1 & -1 & -1 & 2 & 2 & -2 & -2 \\ 1 & -1 & 1 & -1 & 2 & -2 & 2 & -2 \end{pmatrix},$$

$$\operatorname{vert} P_4 = \begin{pmatrix} 1 & 2 & 2 & 3 & -2 & -2 & -2 & -2 \\ 2 & 2 & -2 & -2 & 2 & 2 & -2 & -2 \\ 2 & -2 & 2 & -2 & 2 & -2 & 2 & -2 \end{pmatrix}$$

FIGURE 1. *Four polytopes in* $\mathbb{R}^3$. *The classes of equivalence under linear, projective and combinatorial equivalence are* $(\{P_1, P_2\}, \{P_3\}, \{P_4\})$, $(\{P_1, P_2, P_3\}, \{P_4\})$ *and* $(\{P_1, P_2, P_3, P_4\})$ *respectively.*

projective and the combinatorial symmetry groups. A practical method to compute the linear symmetry group is based on the following theorem.

THEOREM 2 [4]. *Let* $v = \{v_i \mid 1 \leqslant i \leqslant p\}$ *be a set of generators for the extreme rays of a cone* $\mathcal{C}$. *Let* Q *be the following positive definite matrix*

$$Q = \sum_{i=1}^{p} v_i v_i^T. \tag{1}$$

Further, let $G(v)$ *be the complete undirected graph on* p *vertices* $\{1, \ldots, p\}$ *with edge colors* $w_{i,j} = v_i^T Q^{-1} v_j$. *We obtain the linear group* $\mathrm{Lin}_v(\mathcal{C})$ *as automorphism group of the colored graph* $G(v)$.

This theorem reduces the determination of $\mathrm{Lin}_v(\mathcal{C})$ to the computation of the automorphism group of an edge colored graph $G(v)$. At the end of this section, in § 3.4, we give some general recipes to deal with such graphs.

In the particular case of a polytope $P \subset \mathbb{R}^n$ generated by $\{v_i \mid 1 \leqslant i \leqslant p\}$ with associated polyhedral cone $\mathcal{C}$ generated by $\{v_i' = (1, v_i^T)^T \mid 1 \leqslant i \leqslant p\}$, the matrix Q^{-1} allows one to define a Euclidean scalar product on $\mathbb{R}^n$ for which $\mathrm{GL}_{v'}(\mathcal{C})$ is the group of affine isometries of P.

Let us note that one can compute $\mathrm{Lin}_v(\mathcal{C})$ for polytopes with a few thousand vertices using Theorem 2 together with the methods to work with colored graphs presented in § 3.4. For polytopes with a large vertex set some reduction may be necessary. For instance, one idea used in [**25**], for which the polytope of interest has about 10^8 vertices, is to compute the stabilizer of a vertex.

In high dimensions a key bottleneck is the computation of the inverse of the matrix Q. One approach to the problem is to compute the inverse using double precision floating point

numbers. A tolerance number tol has to be chosen and values of Q_{ij}^{-1} which are within tol have to be grouped. One then computes the automorphism group of the colored graph for the grouped colors and checks if the obtained graph automorphisms can actually be represented by matrices of $\mathrm{GL}_v(\mathcal{C})$. If they cannot, then tol has to be decreased or double precision is not enough.

3.1. $\mathrm{GL}_n(\mathbb{Z})$ *symmetries*

In some applications, like integer linear programming, the goal is to find some $\mathrm{GL}_n(\mathbb{Z})$ subgroup of $\mathrm{GL}_v(\mathcal{C})$, rather than the full linear symmetry group. We define $\mathrm{GL}_v(\mathcal{C},\mathbb{Z}) = \mathrm{GL}_v(\mathcal{C}) \cap \mathrm{GL}_n(\mathbb{Z})$. We assume $v_i \in \mathbb{Z}^n$ in this subsection. If the $\{v_i \mid 1 \leqslant i \leqslant p\}$ span $\mathbb{Z}^n$ as a $\mathbb{Z}$-lattice, then $\mathrm{GL}_v(\mathcal{C},\mathbb{Z}) = \mathrm{GL}_v(\mathcal{C})$. However, in general this equality does not hold and we need additional ideas for computing $\mathrm{GL}_v(\mathcal{C},\mathbb{Z})$. Below we list some possible strategies.

Using an auxiliary lattice. For a positive definite symmetric matrix Q the automorphism group G_Q is defined as

$$G_Q = \{A \in \mathrm{GL}_n(\mathbb{Z}) \mid AQA^T = Q\}$$

and can be interpreted as the automorphism group of a lattice, that is of a discrete additive subgroup of the integral vectors. The group G_Q can thus be computed with the algorithm of Plesken and Souvignier [**31**] implemented in `ISOM/AUTO`.

If $A \in \mathrm{GL}_v(\mathcal{C},\mathbb{Z})$ then one gets easily that $AQA^T = Q$ for the positive definite matrix $Q = \sum_{i=1}^{p} v_i v_i^T$. So $\mathrm{GL}_v(\mathcal{C},\mathbb{Z}) \subset G_Q$. We can obtain $\mathrm{GL}_v(\mathcal{C},\mathbb{Z})$ from G_Q by computing the setwise stabilizer. In principle, the Plesken–Souvignier algorithm can be adapted to include this stabilization: as the matrices A are generated row by row in a backtrack algorithm, we can check whether the current set of rows of A stabilizes a projection of $\{v_1,\ldots,v_p\}$ accordingly. If it violates this stabilization property, we may discard the entire candidate branch.

The Plesken–Souvignier algorithm computes a set $S \subset \mathbb{Z}^n$ of short lattice vectors, which may itself be a difficult task. Then a backtrack search on S is employed to compute G_Q. If we briefly ignore the computational cost of S, the lattice approach has the advantage of working with an $n \times n$ matrix instead of a $p \times p$ matrix. Thus for polyhedra which are generated by many rays and for which S is not too difficult to compute this may be a viable alternative.

Example 2. A particular class of polytopes that we encountered with this property are so-called *consecutive ones polytopes* (see, for example, [**28**]). These arise as the convex hull of $m \times n$ matrices with $0,1$ entries satisfying a consecutive ones property. Because these polytopes have about $2^{m\cdot n}$ vertices in dimension mn, the graph construction for computing symmetries is infeasible even for small m and n. The linear symmetries can nevertheless be obtained very quickly in small dimensions by using the strategy described above.

Iterating over group elements. If the group $\mathrm{GL}_v(\mathcal{C})$ is known then $\mathrm{GL}_v(\mathcal{C},\mathbb{Z})$ can be obtained by iterating over all group elements and selecting the integral elements.

This method is actually quite efficient if $\mathrm{GL}_v(\mathcal{C})$ is not too large. For large groups it can be made more efficient by an *intermediate subgroup algorithm*, which we now explain.

Given three groups $G_1 \subset H \subset G_2$, let G_1 and G_2 be fully known and H only be described by an oracle. That is, for every $g \in G_2$ we can decide whether $g \in H$ or not. Our goal is to compute an explicit representation of the group H. We can do a double coset decomposition of G_2 using the subgroup G_1:

$$G_2 = \bigcup_{i=1}^{s} G_1 g_i G_1$$

with $g_i \in G_2$ and $G_1 g_i G_1 \cap G_1 g_{i'} G_1 \neq \emptyset$ if and only if $i = i'$. Suppose that $g \in H$, then since $G_1 \subset H$, for every $f, f' \in G_1$ we have $fgf' \in H$. So, for a given $g \in G_2$ either $G_1 g G_1 \subset H$

or $G_1 g G_1 \cap H = \emptyset$. This allows a reduction in the number of oracle calls. Additionally, if we found a $g \in G_2 - G_1$ that belongs to H then we can replace G_1 by the group generated by g and G_1 and recompute the double coset decomposition. The underlying assumption to get good performance using the intermediate subgroup algorithm is that the index $[G_2 : G_1]$ is not 'too large'.

EXAMPLE 3. This method was used with success in [9] for $v = \{\pm v_1, \ldots, \pm v_n\}$ with $\{v_i\}$ being a basis of $\mathbb{R}^n$. In that case $G_2 = \mathrm{GL}_v(\mathcal{C})$ and G_1 was chosen to be the group generated by transpositions of the v_i and sign changes inducing integral matrix transformations. In order for the intermediate subgroup algorithm to be efficient, one needs to find a sufficiently large group G_1; unfortunately there is no systematic method known to do that. Note that the technique used for computing $\mathrm{GL}_v(\mathcal{C})$ actually works for any finite set of vectors v, not necessarily generating a polyhedral cone.

Adding elements to $\mathcal{C}$. If the generators v_i of $\mathcal{C}$ integrally generate $\mathbb{Z}^n$ then $\mathrm{GL}_v(\mathcal{C}) = \mathrm{GL}_v(\mathcal{C}, \mathbb{Z})$. Hence, if one is able to add vectors $\mathcal{W} = \{w_1, \ldots, w_r\}$ to the generating set of $\mathcal{C}$ such that any elements of $\mathrm{GL}_v(\mathcal{C}, \mathbb{Z})$ will preserve $\mathcal{W}$ then this group can be obtained by the same method used for $\mathrm{GL}_v(\mathcal{C})$, this time applied to the set $\{v_1, \ldots, v_m, w_1, \ldots, w_r\}$. Although adding vectors $\mathcal{W}$ works quite well in specific examples, we do not know of a general algorithm to obtain such a set W.

EXAMPLE 4. One such case is considered in [11] when computing the Delaunay[†] tessellations of an n-dimensional lattice L. For more information on Delaunay tessellations of lattices in general we refer the interested reader to [36].

We denote by $\mathrm{Isom}(P)$ the group of isometries of a Delaunay polytope P and by $\mathrm{Isom}_L(P)$ the group of isometries of P that also preserve the lattice L. In order to obtain the group $\mathrm{Isom}_L(P)$ with help of the Plesken–Souvignier algorithm we set up a homogenized problem: we define an $(n+1)$-dimensional lattice L' spanned by all $(0, v)$ and $(1, v - c)$ with c the center of the Delaunay sphere around P for $v \in L$. Then the automorphism group of L' contains an index 2 subgroup isomorphic to $\mathrm{Isom}_L(P)$ and the involution $-\mathrm{Id}_L$. The Plesken–Souvignier algorithm can be used to compute the automorphism group of the lattice L' under the side constraint that a set S of vectors is kept invariant. In our case, we use the homogenized vertices of P and $-P$ as this set S, which gives us the desired group $\mathrm{Isom}_L(P)$. See [11] for more details and [41] for an implementation of this technique.

Stabilizer on integral embeddings. Let us consider $L = \mathbb{Z}^n$ as a lattice. The group $\mathrm{GL}_v(\mathcal{C})$ does not necessarily stabilize L and so it defines an orbit $O(L) = \{L_1 = L, L_2, \ldots, L_k\}$. The group $\mathrm{GL}_v(\mathcal{C}, \mathbb{Z})$ is then the stabilizer of L in $\mathrm{GL}_v(\mathcal{C})$.

Let $L' = \mathbb{Z}v_1 + \ldots + \mathbb{Z}v_m$ be the lattice spanned by the v_j. Since the vectors v_j are assumed to be integral we have $L' \subset L$. Any element $g \in \mathrm{GL}_v(\mathcal{C})$ preserves L' setwise as it permutes the v_j. Therefore we have the inclusion $L' = g(L') \subset g(L) = L_i$ for some i. On the other hand, for every index i there exists g in $\mathrm{GL}_v(\mathcal{C})$ with $g(L) = L_i$, showing $L' \subset L_i$ for every i. Let $d \in \mathbb{N}$ be the smallest integer such that $L_i \subset (1/d)L'$ for every i.

By choosing a basis $w = (w_1, \ldots, w_n)$ of $(1/d)L'$ we can conjugate $\mathrm{GL}_v(\mathcal{C})$ into a finite subgroup H of $\mathrm{GL}(n, \mathbb{Z})$. In the basis w, the sublattices L_i are expressed with integral coordinates. Since $L' \subset L_i$ for all i we can actually quotient out by L' and this corresponds to a mapping of H into a finite subgroup H_d of $\mathrm{GL}(n, \mathbb{Z}/d\mathbb{Z})$. Thus $\mathrm{GL}_v(\mathcal{C}, \mathbb{Z})$ can be obtained as a stabilizer of a finite set, that is, L/L'.

[†]We note that Delaunay is often spelled *Delone*, as the latter is a straightforward English transliteration of his name in Cyrillic.

This stabilization computation can be accelerated by using the divisors of d. For any divisor h of d we can map $\mathrm{GL}(n,\mathbb{Z}/d\mathbb{Z})$ to $\mathrm{GL}(n,\mathbb{Z}/h\mathbb{Z})$. By using a sequence of divisors $(d_i)_{1\leqslant i\leqslant l}$ with $d_1 = 1$, $d_l = d$ and $d_i | d_{i+1}$ we can obtain a sequence of stabilizers that converges to the desired stabilizer.

In the case of a Delaunay polytope P in a lattice L we can simplify those constructions a little. The finite group $\mathrm{Aut}(L)$ of isometries of L preserving 0 is identified with the group of isometries of the quotient $\mathbb{R}^n/L$. The isobarycenter c of P is expressed as $(1/m)v$ with $v \in L$ and $m \in \mathbb{N}$ and the group $\mathrm{Isom}_L(P)$ is identified with the stabilizer of $c \in \mathbb{R}^n/L$ by $\mathrm{Aut}(L)$. For any divisor d of m we can consider the stabilizer in $\mathbb{R}^n/L'$ with $L' = (1/d)L$ and the factorization of m gives a sequence of stabilizers that converges to $\mathrm{Isom}_L(P)$. See [**11**, **41**] for more details.

3.2. *Symmetries of integer linear programming problems*

In integer linear programming, one optimizes over the intersection of a polyhedron with the integer lattice. From a mathematical point of view, the natural symmetries thus preserve the polyhedron and the integer lattice, that is, they are $\mathrm{GL}_n(\mathbb{Z})$-symmetries. As far as we know, these general symmetries are not used in existing integer optimization software. Instead, the common practice is to consider only *coordinate symmetries*, that is, permutations of coordinates that are automorphisms of the polyhedron. These symmetries turn out to be easier to compute, and also straightforward to work with in integer linear programming solvers.

Several authors have been concerned with ways to compute coordinate symmetries, by reducing the problem to a graph automorphism problem (see [**2**, **26**, **32**, **34**]). Coordinate symmetries are isomorphic to the automorphisms of the following complete bipartite graph. Its vertex set is the union $\{v_1,\ldots,v_p\} \cup \{x_1,\ldots,x_n\}$ of generators (coming from inequalities) and variables. Between each pair v_i and x_j we add an edge colored by the coefficient of variable x_j in generator v_i. This bipartite edge colored graph is simpler to handle than the complete colored graph required for more general symmetries. In integer programming we also have an objective function, which has to be considered for symmetry computation. We can deal with this by coloring the graph vertices that correspond to the variables $x_1,\ldots,x_n$ by their respective objective coefficient.

Using this graph and transformation techniques detailed in § 3.4, the coordinate symmetries of 353 polytopes from mixed integer optimization were computed in [**30**] (cf. [**33**]). Some of the larger instances, which had more than one million variables or facets, were still computationally tractable. In 208 polytopes a non-trivial symmetry group was found. For the 50 smallest problems, with dimension less than 1500, we also computed the linear symmetry group. We found that in 6 out of these 50 cases the linear symmetry group is larger than the coordinate symmetry group. All these linear symmetries are realized by integral matrices.

3.3. *Centralizer subgroups*

We now give an algorithm for centralizer subgroups that is useful particularly in applications in the geometry of numbers; several examples follow at the end of the section. For a given set $\mathcal{B} \subset M_n(\mathbb{R})$ we want to find the group

$$\mathrm{GL}_v(\mathcal{C},\mathcal{B}) = \{A \in \mathrm{GL}_v(\mathcal{C}) \mid AB = BA \text{ for } B \in \mathcal{B}\},$$

that is, the group of elements in $\mathrm{GL}_v(\mathcal{C})$ that preserve a set $\mathcal{B}$ pointwise by conjugation. Without loss of generality we may assume that the set $\mathcal{B}$ is linearly independent, contains the identity matrix and is written as $\{B_1,\ldots,B_r\}$ with $B_1 = I_n$. We denote by $\mathrm{Lin}_v(\mathcal{C},\mathcal{B})$ the corresponding isomorphic permutation group which is a subgroup of $\mathrm{Lin}_v(\mathcal{C})$.

THEOREM 3. *If $\mathcal{B} = \{B_1,\ldots,B_r\}$ is a set of $n \times n$-matrices with $B_1 = I_n$ then the group $\mathrm{Lin}_v(\mathcal{C},\mathcal{B})$ is the group of permutations σ preserving the directed colored graph with edge and*

vertex colors

$$w_{ij} = (v_i^T Q^{-1} B_1 v_j, \ldots, v_i^T Q^{-1} B_r v_j) \quad \text{with } Q = \sum_{i=1}^{p} v_i v_i^T.$$

Proof. If $A \in \mathrm{GL}_v(\mathcal{C}, \mathcal{B})$ then $AB_i = B_i A$ and $Av_i = v_{\sigma(i)}$. Hence one gets by summation that $AQA^T = Q$ or equivalently $Q^{-1} = A^T Q^{-1} A$ (obtained from $Q^{-1} = (A^T)^{-1} Q^{-1} A^{-1}$ by left and right multiplication). So, if one writes $h_{ij}^k = v_i^T Q^{-1} B_k v_j$ then one gets

$$\begin{aligned} h_{ij}^k &= v_i^T Q^{-1} B_k v_j \\ &= v_i^T A^T Q^{-1} A B_k v_j \\ &= (Av_i)^T Q^{-1} B_k (Av_j) \\ &= h_{\sigma(i)\sigma(j)}^k. \end{aligned}$$

So, any A induces a permutation of the p vertices preserving the vector edge color w_{ij}.

Suppose now that $\sigma \in \mathrm{Sym}(p)$ satisfies $w_{ij} = w_{\sigma(i)\sigma(j)}$. Then, since $B_1 = I_n$ we have $v_i^T Q^{-1} v_j = v_{\sigma(i)}^T Q^{-1} v_{\sigma(j)}$. By an easy linear algebra computation (see [4, 10] for details) we get that there exists $A \in \mathrm{GL}_n(\mathbb{R})$ such that $Av_i = v_{\sigma(i)}$. If one writes $w_i = Q^{-1} A v_i$ then

$$\begin{aligned} w_i^T A B_k v_j &= v_i^T A^T Q^{-1} A B_k v_j \\ &= v_i^T Q^{-1} B_k v_j \\ &= h_{ij}^k \\ &= h_{\sigma(i)\sigma(j)}^k \\ &= v_{\sigma(i)}^T Q^{-1} B_k v_{\sigma(j)} \\ &= v_i^T A^T Q^{-1} B_k A v_j \\ &= w_i^T B_k A v_j. \end{aligned}$$

Since the families $\{v_j\}$ and $\{w_i\}$ span $\mathbb{R}^n$ we get $AB_k = B_k A$. □

There are many contexts where the above theorem is useful.

Example 5. If one wishes to find the group of elements $A \in \mathrm{GL}_n(\mathbb{Q}[\sqrt{5}])$ then one way to do so is to express the elements as elements of $\mathrm{GL}_{2n}(\mathbb{Q})$ that commute with the multiplication by $\sqrt{5}$. This is very useful when working with Humbert forms whose symmetry group in $\mathrm{GL}_n(\mathbb{Z}[\sqrt{5}])$ correspond to a small subgroup of the full group in $\mathrm{GL}_{2n}(\mathbb{Z})$.

Example 6. If one wishes to find the elements belonging to $\mathrm{GL}_n(\mathbb{H})$, with $\mathbb{H}$ the Hamilton's quaternions, then the above method can also be applied; $\mathrm{GL}_n(\mathbb{H})$ acts on $\mathbb{H}^n$ by multiplication on the left and it is characterized in $\mathrm{GL}_{4n}(\mathbb{R})$ by the fact that it commutes with scalar Hamiltonian multiplication on the right.

Example 7. If one wishes to compute the group $\mathrm{GL}_v(\mathcal{C}, W)$ of elements of $\mathrm{GL}_n(\mathbb{R})$ preserving a polytope $\mathcal{C}$ and a vector space W, then the above method can also be applied. Any such element will be an isometry for the scalar product defined by Q^{-1} in the proof of Theorem 3 and so will commute with the orthogonal projection on W.

Let us finally note that one could wish for a similar characterization for elements of $\mathrm{GL}_v(\mathcal{C})$ that preserve a set $\mathcal{B}$ setwise by conjugation and so compute normalizer groups. We do not know of such and actually think that a similar characterization is not possible.

3.4. *Computing with vertex and edge colored graphs*

Many of the strategies presented above depend on the computation of the automorphism group of a graph whose vertices and/or edges are colored. The complexity of the graph isomorphism problem is uncertain. It is one of the rare problems in NP which is neither known to be NP-complete nor in P.

For practical computation there exists graph isomorphism software (see [**40**, **44**, **46**–**48**]) that can compute automorphism groups of graphs. Such programs usually use the partition backtrack algorithm and can compute the automorphism groups of large graphs but their run time is exponential in the worst case. These programs usually cannot handle edge colored graphs and suffer from a performance penalty when using digraphs. Hence, one needs reduction techniques. There are several techniques for reducing an edge colored graph to a vertex colored graph (a complete graph with only two edge colors). We describe two of them in more detail here.

Reduction by intermediate nodes. One transformation described in [**34**] replaces each c-colored edge $\{a, b\}$ with an intermediate c-colored vertex m, which has edges connecting it to a and b. Starting with a complete graph on n vertices, the transformed graph has $n+n(n-1)/2$ vertices which makes this transformation expensive. For the bipartite edge colored graphs that occur in § 3.2 this can be improved by an idea given in [**32**]. Instead of adding intermediate vertices for all edges, we combine some of those with the same color. Define the bipartition as (S, S'). For each $i \in S$ let $X_{i,c} \subseteq S'$ be the set of vertices which are incident to i with an edge of color c. Then it is enough to introduce an intermediate c-colored vertex m with edges to i and to all elements of $X_{i,c}$. For many integer optimization problems these sets $X_{i,c}$ are often large, thus the number of vertices in the graph is usually substantially reduced. If $|S| > |S'|$, it may be advantageous to combine edges the other way around.

Reduction by superposition. For general edge colored graphs we use the following method proposed in the user manual of `nauty` [**46**]. Suppose that we have M edge colors. Then any color can be expressed as a 0/1 word of length $\lceil \log_2(M) \rceil$. Therefore the automorphism group can be obtained from superposition of $\lceil \log_2(M) \rceil$ vertex colored graphs as follows. For $1 \leqslant i \leqslant \lceil \log_2(M) \rceil$ let G_i have the same vertices as the original edge colored graph which we want to transform. Two vertices a, b are connected by an edge in G_i if and only if there is an edge e between a and b in the original graph G and the binary color word of e has a one at its ith position. To get the final superposition graph, take the union of the G_i, coloring all vertices in G_i by some color i. The resulting graph thus has $n\lceil \log_2(M) \rceil$ vertices where n is the number of vertices of the original graph. This solution has good complexity estimates but the preceding method is often the best for the bipartite graphs from integer optimization (see [**30**, **33**]).

Dealing with digraphs. We finally note that colored digraphs can be transformed into colored graphs with twice the number of vertices. Let γ and γ' be two colors that do not occur as an arc or vertex color. Color the uncolored vertices with γ, and construct a colored bipartite graph as follows. Each vertex a corresponds to a pair $\{a, a'\}$, with a retaining the color and a' uncolored. For each vertex and for each arc of the original graph edges are added: edges (a, a') of color γ' for each vertex a and edges (a, b') for each arc (a, b), retaining the color of (a, b).

4. *Computing* $\mathrm{Proj}(\mathcal{C})$

For a given polyhedral cone $\mathcal{C}$, the group $\mathrm{Proj}(\mathcal{C})$ is the group of permutations of extreme rays that are induced by a linear transformation preserving $\mathcal{C}$. We give a method allowing the

computation of the projective group in practice. It is based on a decomposition for polyhedral cones and on some linear algebra tests.

A polyhedral cone $\mathcal{C}$ generated by extreme rays $E = \{R_i \mid 1 \leqslant i \leqslant p\}$ in an n-dimensional vector space V is said to be *decomposable* if there exist two non-empty subspaces V_1, V_2 such that $E = (E \cap V_1) \cup (E \cap V_2)$ and $V = V_1 \oplus V_2$.

THEOREM 4. *For a polyhedral cone $\mathcal{C}$ generated by $E = \{R_i \mid 1 \leqslant i \leqslant p\}$ in an n-dimensional vector space V, there is a unique decomposition*

$$V = \bigoplus_{1 \leqslant k \leqslant h} V_k$$

with

$$E = \bigcup_{1 \leqslant k \leqslant h} E \cap V_k$$

and the cone $\mathcal{C}_k$ generated by $E \cap V_k$ being non-decomposable. This decomposition can be computed in time $O(pn^2)$.

Proof. The existence of the decomposition is obvious since the vector space V is finite dimensional and so the decomposition process has to end at some point. The uniqueness follows immediately from the fact that if there are two decompositions by V_k and V_l' then the family of subspaces $V_k \cap V_l'$ also defines a decomposition.

The method for computing a decomposition is the following. First select some generators v_i of R_i. Then find an n-element set $S \subset \{1, \ldots, p\}$ such that $\{v_i \mid i \in S\}$ is a basis of V, for example, by Gaussian elimination in $O(pn^2)$ arithmetic operations. Every vector v_i for $1 \leqslant i \leqslant p$ is then written as $v_i = \sum_{k \in S} \alpha_{k,i} v_k$. The supports

$$S_i = \{k \in S \mid \alpha_{k,i} \neq 0\}$$

determine the edges of a hypergraph on n vertices. The connected components of this hypergraph correspond to the summands of the decomposition of V. Finding the connected components can be done in time $O(pn)$. □

REMARK 5. The master's thesis of Golynski [14] describes how to use divide and conquer and fast matrix multiplication to reduce the complexity of computing a basis of V to $O(pn^{1+\delta})$ for $0 \leqslant \delta \leqslant 0.3727$.

Besides being useful for giving an algorithm for computing $\mathrm{Proj}(\mathcal{C})$, the notion of decomposability is useful for classifying some polytopes.

THEOREM 6. *Let us take $n \geqslant 3$.*

(i) *The number of classes of non-decomposable polyhedral cones in $\mathbb{R}^n$ with $n+1$ extreme rays under projective equivalence is $\lfloor (n-1)/2 \rfloor$.*

(ii) *The number of classes of polyhedral cones in $\mathbb{R}^n$ with $n+1$ extreme rays under projective equivalence is*

$$\begin{cases} p(p-1) & \text{if } n = 2p, \\ p^2 & \text{if } n = 2p+1. \end{cases}$$

Proof. Let us take $v_1, \ldots, v_{n+1}$ to be the generators of a cone $\mathcal{C}$ in $\mathbb{R}^n$. Up to scalar multiples, there is a unique non-trivial linear dependency relation of the form

$$0 = \sum_{i=1}^{n+1} \alpha_i v_i \quad \text{with } \alpha_i \in \mathbb{R}.$$

Let us assume that $\mathcal{C}$ is non-decomposable. This is equivalent to $\alpha_i \neq 0$ for all i. Since we are considering projective equivalence, we can replace generators v_i by positive multiples. So, only the signs of the α_i matter and so by permutation of the generators only their number n_+ and n_-. The case $n_+ = 0$ is impossible since it implies that all generators are zero. The case $n_+ = 1$ is equally impossible since it implies that one generator is a positive sum of other generators. So, $n_+ \geqslant 2$ and by symmetry $n_- \geqslant 2$. Moreover, $n_+ + n_- = n + 1$. Since one can flip the signs, this gives $f(n) = \lfloor (n/2) \rfloor$ projective types and (i) holds.

For (ii) it suffices to remark that if $\alpha_i = 0$ then we can remove v_i from the discussion and consider a lower-dimensional cone. So, the number of projective types of polyhedral cones with $n + 1$ generators in $\mathbb{R}^n$ is $\sum_{k=3}^{n} f(k)$. □

From the above theorem it follows that any two 4-gons in $\mathbb{R}^2$ are projectively equivalent.

The above decomposition allows one to determine the projective symmetry group. We say that two cones have the same *projective isomorphism type* if there is a bijective linear map between them.

THEOREM 7. *For a polyhedral cone $\mathcal{C}$ generated by rays $E = \{R_i \mid 1 \leqslant i \leqslant p\}$ in a vector space V, uniquely decomposed into $V = \bigoplus_{1 \leqslant k \leqslant h} V_k$ as in Theorem 4, let $\mathcal{C}_k$ be the cone generated by $E \cap V_k$.*

If the projective isomorphism type j for $1 \leqslant j \leqslant q$ occurs n_j times among the $\mathcal{C}_k$ then we have the equality

$$\mathrm{Proj}(\mathcal{C}) = \prod_{j=1}^{q} \mathrm{Proj}(\mathcal{C}_{k_j}) \wr \mathrm{Sym}(n_j)$$

where $\wr$ stands for the wreath product and $\mathcal{C}_{k_j}$ is a representative for the jth type.

Proof. Suppose that we have an element $f \in \mathrm{Proj}(\mathcal{C})$. This element permutes the $\mathcal{C}_k$ but must also preserve the projective isomorphism types. Hence, it belongs to the above-mentioned product. The reverse inclusion is trivial. □

We now discuss a method for computing the projective symmetry group of a non-decomposable polyhedral cone $\mathcal{C}$. Combined with the above theorem, this will allow us to compute the projective symmetry group of arbitrary polyhedra. We first present the following structural result.

THEOREM 8. *Suppose $\mathcal{C}$ is a non-decomposable polyhedral cone generated by rays $\{R_i \mid 1 \leqslant i \leqslant p\}$. Then:*

(i) *we have the isomorphism*

$$\mathrm{GL}(\mathcal{C}) \cong \mathbb{R}_+^* \times \mathrm{Proj}(\mathcal{C});$$

(ii) *there exist vectors v_i such that $R_i = \mathbb{R}_+ v_i$ and*

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

Proof. Let us denote by μ the natural surjective homomorphism (with kernel K) from $\mathrm{GL}(\mathcal{C})$ to $\mathrm{Proj}(\mathcal{C})$. For $A \in K$ we have $AR_i = \alpha_i R_i$ with $\alpha_i > 0$. If the values α_i were not all the same then this would automatically give a decomposition of the space (since each class of rays with the same multiplier will be subdimensional) contradicting the non-decomposability of $\mathcal{C}$. Thus $K = \{\lambda I \mid \lambda \in \mathbb{R}_+^*\}$.

Let us write $\mathcal{G} = \{f \in \mathrm{GL}(\mathcal{C}) \mid \det f = \pm 1\}$. The kernel of μ restricted to $\mathcal{G}$ is trivial. So $\mathcal{G}$ is isomorphic to $\mathrm{Proj}(\mathcal{C})$. Any $f \in \mathrm{GL}(\mathcal{C})$ is uniquely written as $f = \lambda g$ with $g \in \mathcal{G}$ and $\lambda \in \mathbb{R}_+^*$. Hence statement (i) follows.

To show (ii) we can, by (i), identify $\mathrm{Proj}(\mathcal{C})$ with the subgroup $\mathcal{G}$ of $\mathrm{GL}(\mathcal{C})$. This identification is unique, as $\mathcal{G}$ is the subgroup of elements of finite order. For each $\mathrm{Proj}(\mathcal{C})$-orbit O_k of extreme rays we choose a representative ray $R_k \in O_k$ and a vector v_k such that $R_k = \mathbb{R}_+ v_k$. For each element $R \in O_k$ we can find an $f \in \mathcal{G}$ such that $R = f(R_k)$. If there is another $f' \in \mathcal{G}$ such that $R = f'(R_k)$ then $h(v_k) = Cv_k$ with $h = f^{-1}f'$ and $C > 0$. Since $h \in \mathcal{G}$, it has finite order, say, m, and $h^m(v_k) = C^m v_k = v_k$, implying $C = 1$. This means that $f(v_k)$ is uniquely defined. It is then clear that for this choice of generators $\mathrm{Proj}(\mathcal{C}) = \mathrm{Lin}_v(\mathcal{C})$. □

Remark 9. The statements of Theorem 8 cease to hold in a more general setting of a configuration of points in a projective space over $\mathbb{C}$. In the following we construct a counterexample.

Example 8. First we take the group G characterized as $2 \cdot S_4^-$ (namely, number 28 in `GAP` the database of small groups [**42**]) and its faithful 2-dimensional representation over $\mathbb{C}$. The center of G in this representation is $\pm \mathrm{Id}_2$. We then take an element of order 8 in G and compute one of its eigenspaces, corresponding to an eighth primitive root of unity. There are six images of the eigenspace under G. Thus we obtain a transitive action of G on a 6-tuple of lines and so on 6 points in $P^1(\mathbb{C})$. The action on the six lines defines a group S_4. But G is not isomorphic to $2 \times S_4$.

Theorem 10. *Suppose $\mathcal{C}$ is a non-decomposable cone generated by extreme rays $\{R_i \mid 1 \leqslant i \leqslant p\}$ in $\mathbb{R}^n$. Testing if a permutation $\sigma \in \mathrm{Sym}(p)$ belongs to $\mathrm{Proj}(\mathcal{C})$ can be done in $O(p^3 n)$ time.*

Proof. Let us slightly abuse notation and denote by $V_\mu = (v_{\mu(1)}, \ldots, v_{\mu(p)})$ the matrix with columns being a set of generators for the R_i, that is, $R_i = \mathbb{R}_+ v_i$, for $1 \leqslant i \leqslant p$, permuted by a permutation μ. Respectively, let U_μ denote the submatrix of V_μ consisting of its first n columns, and $V = V_{\mathrm{id}}$, $U = U_{\mathrm{id}}$. Without loss of generality, U is invertible.

The sought matrix $A \in \mathrm{GL}(\mathcal{C})$, a preimage of σ, must satisfy $Av_i = \alpha_i v_{\sigma(i)}$, $\alpha_i > 0$, for each $1 \leqslant i \leqslant p$. In the matrix form this can be written as $AV = V_\sigma \,\mathrm{diag}(\alpha_1, \ldots, \alpha_p)$. In particular, $AU = U_\sigma \,\mathrm{diag}(\alpha_1, \ldots, \alpha_n)$, implying

$$A = U_\sigma \,\mathrm{diag}(\alpha_1, \ldots, \alpha_n) U^{-1}. \tag{2}$$

This implies

$$U_\sigma \,\mathrm{diag}(\alpha_1, \ldots, \alpha_n) U^{-1} V = V_\sigma \,\mathrm{diag}(\alpha_1, \ldots, \alpha_p). \tag{3}$$

This is a homogeneous linear system having np equations and unknowns α_i, for $1 \leqslant i \leqslant p$. This system can be solved by, for example, Gaussian elimination in $O(p^3 n)$ arithmetic operations.

Denote by $\mathcal{SP}$ the solution space of (3). A solution α is acceptable if and only if $\alpha > 0$ (the latter implies $\det(A) \neq 0$ by (2)). These conditions are open conditions, so if there is one such solution then there is an open ball of such solutions of dimension $q = \dim \mathcal{SP}$, as well. But we know by Theorem 8 that $q \leqslant 1$ for non-decomposable cones. If $q = 0$ then $\sigma \notin \mathrm{Proj}(\mathcal{C})$. If $q = 1$, we can find a nonzero solution α of (3) and test that $\pm\alpha > 0$. If there is no such α, we conclude that $\sigma \notin \mathrm{Proj}(\mathcal{C})$. Otherwise, picking the right sign of α, we find A using (2) and conclude that $\sigma \in \mathrm{Proj}(\mathcal{C})$. □

Theorem 10 gives a constructive way to compute $\mathrm{Proj}(\mathcal{C})$. Combining with the intermediate subgroup algorithm to compute $\mathrm{Comb}(\mathcal{C})$ given in § 5 gives a more practical method to compute $\mathrm{Proj}(\mathcal{C})$. An easy situation is when $\mathrm{Lin}_v(\mathcal{C}) = \mathrm{Comb}(\mathcal{C})$, which of course implies $\mathrm{Lin}_v(\mathcal{C}) = \mathrm{Proj}(\mathcal{C})$.

Let us take $\alpha_i > 0$. The group $\mathrm{Lin}_{\alpha v}(\mathcal{C})$ (where every generator v_i is scaled by α_i) depends on α and is a subgroup of $\mathrm{Proj}(\mathcal{C})$. By Theorem 8 there exists α such that $\mathrm{Lin}_{\alpha v}(\mathcal{C}) = \mathrm{Proj}(\mathcal{C})$

and it is interesting to know when equality occurs. For any $\alpha > 0$ the group $\mathrm{Lin}_{\alpha v}(\mathcal{C})$ is a permutation group acting on p points which defines an orbit partition $\mathrm{OP}(\alpha)$ of $\{1, \ldots, p\}$.

THEOREM 11. (i) *If* $\{\alpha_i \mid 1 \leqslant i \leqslant p\}$ *and* $\{\alpha'_i \mid 1 \leqslant i \leqslant p\}$ *are two sets of positive multipliers and* $\mathrm{OP}(\alpha) = \mathrm{OP}(\alpha')$ *then* $\mathrm{Lin}_{\alpha v}(\mathcal{C}) = \mathrm{Lin}_{\alpha' v}(\mathcal{C})$.
(ii) *If* $\{\alpha_i \mid 1 \leqslant i \leqslant p\}$ *is a set of positive multipliers and* $\mathrm{Lin}_{\alpha v}(\mathcal{C})$ *is transitive on* $\{1, \ldots p\}$ *then* $\mathrm{Lin}_{\alpha v}(\mathcal{C}) = \mathrm{Proj}(\mathcal{C})$.

Proof. Let us prove (i). We decompose $\mathcal{C}$ into non-decomposable components $\mathcal{C}_k$ for $1 \leqslant k \leqslant h$ and denote by S_k the corresponding subset of $\{1, \ldots, p\}$. Let us take an orbit O under $G_\alpha = \mathrm{Lin}_{\alpha v}(\mathcal{C})$. If $x \in O \cap S_k$ and $H = \mathrm{Stab}_{G_\alpha}(S_k)$ then Theorem 7 giving the expression of the projective symmetry group in terms of a wreath product is also valid for the linear automorphism group. This implies that the orbit of x under H is exactly $O \cap S_k$. Our assumption $\mathrm{OP}(\alpha) = \mathrm{OP}(\alpha')$ implies that $O \cap S_k$ is also an orbit under $H' = \mathrm{Stab}_{G_{\alpha'}}(S_k)$. Furthermore, by the non-decomposability of $\mathcal{C}_k$ we know that α_l is determined up to some constant factor for $l \in O \cap S_k$. Thus there exists a $\beta > 0$ such that $\alpha_l = \beta \alpha'_l$ for $l \in O \cap S_k$. This implies that the linear automorphism groups of $\mathcal{C}_k$ for $\{\alpha_i v_i \mid i \in S_k\}$ and $\{\alpha'_i v_i \mid i \in S_k\}$ are equal. Since $\mathrm{OP}(\alpha) = \mathrm{OP}(\alpha')$ we know that the isomorphism type of the component $\mathcal{C}_k$ under G_α and $G_{\alpha'}$ are the same. Since both groups are actually direct products of wreath products they are necessarily equal.

To prove (ii), note that by Theorem 8 there exists some multiplier vector α' such that $G_{\alpha'} = \mathrm{Proj}(\mathcal{C})$. We see that G_α is a subgroup of $G_{\alpha'}$ so $\mathrm{OP}(\alpha)$ is a partition induced from $\mathrm{OP}(\alpha')$ by splitting some orbit. But by transitivity $\mathrm{OP}(\alpha)$ is reduced to only one component so $\mathrm{OP}(\alpha) = \mathrm{OP}(\alpha')$ and (ii) follows. □

By using item (ii) above one can conclude in some cases that the linear group is actually the projective group.

5. *Computing* $\mathrm{Comb}(\mathcal{C})$

In this section we present a method to compute the combinatorial symmetry group $\mathrm{Comb}(\mathcal{C})$ of a cone $\mathcal{C}$.

Recall that $\mathrm{Comb}(\mathcal{C})$ is the maximal symmetry group of a polyhedral cone that preserves the face lattice. For many polyhedral computations, this is the largest group of symmetries that can be exploited. Although no efficient methods are known for the general case, in this section we describe some techniques that can be useful in certain practical computations. The general idea is to construct a 'sandwich' $G_1 \leqslant \mathrm{Comb}(\mathcal{C}) \leqslant G_2$ between groups G_1 and G_2 that are easier to compute. We also present a possible use of the intermediate subgroup algorithm for computing $\mathrm{Comb}(\mathcal{C})$.

5.1. *Preliminaries*

DEFINITION 4. For an integer $k \geqslant 1$, the group $\mathrm{Skel}_k(\mathcal{C})$ of a polyhedral cone $\mathcal{C}$ is the group of all permutations of extreme rays that preserve $\mathcal{F}_l$ for all $0 \leqslant l \leqslant k$.

Assuming that $\mathcal{C}$ has p extreme rays, we in particular have $\mathrm{Skel}_1(\mathcal{C}) = \mathrm{Sym}(p)$. Moreover, $\mathrm{Skel}_{k+1}(\mathcal{C}) \leqslant \mathrm{Skel}_k(\mathcal{C})$ and $\mathrm{Skel}_{n-1}(\mathcal{C}) = \mathrm{Comb}(\mathcal{C})$. For every integer $k \geqslant 1$ we have the inclusion

$$\mathrm{Lin}_v(\mathcal{C}) \leqslant \mathrm{Proj}(\mathcal{C}) \leqslant \mathrm{Comb}(\mathcal{C}) \leqslant \mathrm{Skel}_k(\mathcal{C}). \tag{4}$$

We note that for 4-dimensional polyhedral cones $\mathcal{C}$, the group $\mathrm{Comb}(\mathcal{C})$ is actually equal to $\mathrm{Skel}_2(\mathcal{C})$, due to Steinitz theorem for 3-dimensional polytopes (see [**38**, Chapter 4]).

Assuming that we know the set $\mathcal{F}_k$ of k-dimensional faces, the group $\mathrm{Skel}_k(\mathcal{C})$ is isomorphic to the automorphism group of a vertex colored graph on $p+|\mathcal{F}_k|$ vertices. The reason is that if an automorphism preserves all the k-dimensional faces, then it preserves all the intersections and so all the faces of dimension at most k. The k-dimensional faces $\mathcal{F}_k$ are thus described by the set S of extreme rays contained in them and so we can build a bipartite graph on $p+|\mathcal{F}_k|$ vertices that encodes this relation.

5.2. *Lucky sandwich*

By the chain of inclusions in (4), the group $\mathrm{Comb}(\mathcal{C})$ is located between two groups which are both automorphism groups of colored graphs. If we can prove that for some k_0 we have $\mathrm{Lin}_v(\mathcal{C}) = \mathrm{Skel}_{k_0}(\mathcal{C})$ then we conclude that $\mathrm{Comb}(\mathcal{C}) = \mathrm{Lin}_v(\mathcal{C})$ and we are finished. This is the most common method (see [**7**, **8**]) for computing combinatorial symmetry groups: determine the set of faces of dimension at most k_0, determine $\mathrm{Skel}_{k_0}(\mathcal{C})$, test if the elements of $\mathrm{Skel}_{k_0}(\mathcal{C})$ are actually in $\mathrm{Lin}_v(\mathcal{C})$ and if yes obtain $\mathrm{Skel}_{k_0}(\mathcal{C}) = \mathrm{Comb}(\mathcal{C})$. But this does not always work since in some cases $\mathrm{Lin}_v(\mathcal{C}) \neq \mathrm{Comb}(\mathcal{C})$. One case where it is guaranteed to work is for simple polytopes, that is, ones for which every vertex is adjacent to exactly n other vertices; for these polytopes $\mathrm{Comb}(\mathcal{C}) = \mathrm{Skel}_2(\mathcal{C})$ (see [**1**, **12**, **18**]).

Typically, the number of k-dimensional faces becomes impractically large for some intermediate values of k. An alternative method for computing $\mathrm{Comb}(\mathcal{C})$ is simply to compute the whole set of facets and then compute $\mathrm{Skel}_{n-1}(\mathcal{C}) = \mathrm{Comb}(\mathcal{C})$ directly. The problem with this method is that the set of facets may be too large for this approach to work and sometimes the facets are precisely what we want to compute in the end.

5.3. *Computing intermediate subgroups*

We now apply the intermediate subgroup algorithm of § 3.1 to computing $\mathrm{Comb}(\mathcal{C})$. The containing group G_2 is taken to be $\mathrm{Skel}_{k_0}(\mathcal{C})$ and the contained group G_1 is taken to be $\mathrm{Lin}_v(\mathcal{C})$.

We need an oracle to decide whether a permutation lies in $\mathrm{Comb}(\mathcal{C})$. Let us assume that we have computed the orbits of facets of $\mathcal{C}$ up to G_1. The possible methods for doing such a computation are reviewed in [**4**]. Denote by $O_1, \ldots, O_r$ the orbits of facets, for which we select some representative $F_1, \ldots, F_r$. They are encoded by their extreme ray incidence as subsets $S_1, \ldots, S_r \subset \{1, \ldots, p\}$. A permutation $\sigma \in \mathrm{Sym}(p)$ belongs to $\mathrm{Comb}(\mathcal{C})$ if and only if any image $\sigma(S_i)$ is in a G_1-orbit of one of our representatives S_j. Such in-orbit tests are done using permutation backtrack algorithms [**23**, **24**] that are implemented, for example, in `GAP` [**42**] and `PermLib` [**48**].

Based on this oracle, the intermediate subgroup algorithm allows us to compute $\mathrm{Comb}(\mathcal{C})$ without having to iterate over all elements of $\mathrm{Skel}_{k_0}(\mathcal{C})$ and test whether they belong to $\mathrm{Comb}(\mathcal{C})$. It is not clear how to do much better since in general one needs the facets in order to get $\mathrm{Comb}(\mathcal{C})$.

Acknowledgements. We thank Frieder Ladisch and Jan-Christoph Schlage-Puchta for their help in simplifying the proof of Theorem 8. We also thank the referee for useful comments that led to an improved manuscript.

References

1. R. Blind and P. Mani-Levitska, 'Puzzles and polytope isomorphism', *Aequationes Math.* 34 (1987) 287–297.

2. R. Bödi, K. Herr and M. Joswig, 'Algorithms for highly symmetric linear and integer programs', *Math. Program.* 137 (2013) 65–90.

3. J. Bokowski, G. Ewald and P. Kleinschmidt, 'On combinatorial and affine automorphisms of polytopes', *Israel J. Math.* 47 (1984) no. 2–3, 123–130.
4. D. Bremner, M. Dutour Sikirić and A. Schürmann, 'Polyhedral representation conversion up to symmetries', *CRM Proc. Lecture Notes* 48 (2009) 45–72.
5. H. S. M. Coxeter, *Projective geometry*, 2nd edn (Springer, New York, 1987).
6. D. A. Cox, J. B. Little and H. K. Schenck, *Toric varieties* (American Mathematical Society, Providence, RI, 2011).
7. A. Deza and M. Deza, 'Skeletons of some relatives of the N-cube', *Operations Research Proceedings 1994* (Springer, 1995) 81–85.
8. A. Deza, B. Goldengorin and D. V. Pasechnik, 'The isometries of the cut, metric and hypermetric cones', *J. Algebraic Combin.* 23 (2006) no. 3, 197–203.
9. M. Dutour Sikirić, K. Hulek and A. Schürmann, 'Smoothness and singularities of the perfect form and the second Voronoi compactification of $\mathcal{A}_g$', Preprint, 2013, arXiv:1303.5846.
10. M. Dutour Sikirić, F. Vallentin and A. Schürmann, 'Classification of eight-dimensional perfect forms', *Electron. Res. Announc. Amer. Math. Soc.* 13 (2007) 21–32.
11. M. Dutour Sikirić, A. Schürmann and F. Vallentin, 'Complexity and algorithms for computing Voronoi cells of lattices', *Math. Comp.* 78 (2009) 1713–1731.
12. E. Friedman, 'Finding a simple polytope from its graph in polynomial time', *Discrete Comput. Geom.* 41 (2009) 249–256.
13. G. Gévais, 'A class of cellulated spheres with non-polytopal symmetries', *Canad. Math. Bull.* 52 (2009) 366–379.
14. A. Golynski, 'A polynomial time algorithm to find the minimum cycle basis of a regular matroid', Master's Thesis, University of New Brunswick, 2002.
15. I. M. Isaacs, *Character theory of finite groups* (Dover, New York, 1994).
16. I. M. Isaacs, *Finite group theory*, Graduate Studies in Mathematics 92 (American Mathematical Society, Providence, RI, 2008).
17. V. Kaibel and A. Schwartz, 'On the complexity of polytope isomorphism problems' *Graphs Comb.* 19-2 (2003) 215–230.
18. G. Kalai, 'A simple way to tell a simple polytope from its graph', *J. Combin. Theory Ser. A* 49 (1988) 381–383.
19. J. Köbler, U. Schöning and J. Torán, *Graph isomorphism problem: the structural complexity* (Birkhäuser, Basel, 1993).
20. M. Kreuzer and H. Skarke, 'On the classification of reflexive polyhedra', *Commun. Math. Phys.* 185 (1997) 495–508.
21. M. Kreuzer and H. Skarke, 'Classification of reflexive polyhedra in three dimensions', *Adv. Theor. Math. Phys.* 2–4 (1998) 853–871.
22. M. Kreuzer and H. Skarke, 'PALP: a package for analysing lattice polytopes with applications to toric geometry', *Comput. Phys. Comm.* 157 (2004) 87–106.
23. J. S. Leon, 'Permutation group algorithms based on partitions. I. Theory and algorithms. Computational group theory, Part 2', *J. Symbolic Comput.* 12 (1991) 533–583.
24. J. S. Leon, 'Partitions, refinements, and permutation group computation', *Groups and computation, II (New Brunswick, NJ, 1995)*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 28 (American Mathematical Society, Providence, RI, 1997) 123–158.
25. C.-K. Li and T. Milligan, 'Linear preservers of finite reflection groups', *Linear Multilinear Algebra* 41 (2003) 49–81.
26. L. Liberti, 'Reformulations in mathematical programming: automatic symmetry detection and exploitation', *Math. Program.* 131 (2012) 273–304.
27. F. Margot, 'Symmetry in integer linear programming', *50 years of integer programming* (Springer, Berlin, 2010) 647–686.
28. M. Oswald, 'Weighted consecutive ones problems', PhD Thesis, Universität Heidelberg, 2001.
29. A. Paffenholz and G. M. Ziegler, 'The E_t-construction for lattices, spheres and polytopes', *Discrete Comput. Geom.* 32 (2004) no. 4, 601–621.
30. M. E. Pfetsch and T. Rehn, 'Symmetry handling in integer programs revisited' (in preparation).
31. W. Plesken and B. Souvignier, 'Computing isometries of lattices', *J. Symbolic Comput.* 24 (1997) 327–334.
32. J.-F. Puget, *Automatic detection of variable and value symmetries*, Lecture Notes in Computer Science 3709 (Springer, Berlin, 2005).
33. T. Rehn, 'Exploring core points for fun and profit a study of lattice-free orbit polytopes', PhD Thesis, University of Rostock, 2014.
34. D. Salvagnin, 'A dominance procedure for integer programming', Master's Thesis, University of Padova, 2005.
35. A. Schrijver, *Combinatorial optimization. Polyhedra and efficiency, vol. A, B, C* (Springer, Berlin, 2003).
36. A. Schürmann, *Computational geometry of positive definite quadratic forms* (American Mathematical Society, Providence, RI, 2009).

37. A. Schürmann, 'Exploiting symmetries in polyhedral computations', *Discrete Geometry and Optimization*, Fields Institute Communications 69 (Springer, New York, 2013) 265–278.
38. G. Ziegler, *Lectures on polytopes*, Graduate Texts in Mathematics vol. 152 (Springer, New York, 1995).

Software

39. CPLEX, 'Mathematical programming technology', http://www.ilog.com/products/cplex/.
40. P. T. Darga, H. Katebi, M. Liffiton, I. Markov and K. Sakallah, saucy, http://vlsicad.eecs.umich.edu/BK/SAUCY/.
41. M. Dutour Sikirić, polyhedral, http://drobilica.irb.hr/∼mathieu/Polyhedral/.
42. The GAP group, 'GAP—groups, algorithms, and permutations, version 4.4.6', 2005, http://www.gap-system.org.
43. Gurobi, 'High-end libraries for math programming', http://www.gurobi.com/.
44. T. Junttila and P. Kaski, bliss, http://www.tcs.hut.fi/Software/bliss/.
45. M. Kreuzer, H. Skarke and others, 'PALP: a package for analysing lattice polytopes', http://hep.itp.tuwien.ac.at/∼kreuzer/CY/CYpalp.html.
46. B. D. McKay, The nauty program, http://cs.anu.edu.au/people/bdm/nauty/.
47. A. Piperno, The trace program, http://cs.anu.edu.au/people/bdm/nauty/.
48. T. Rehn, Permlib, http://www.geometrie.uni-rostock.de/software/.
49. T. Rehn and A. Schürmann, SymPol, http://www.geometrie.uni-rostock.de/software/.

Web page

50. A. Paffenholz, http://www.mathematik.tu-darmstadt.de/∼paffenholz/data.html.

David Bremner
Faculty of Computer Science
University of New Brunswick
Box 4400, Fredericton NB
E3B 5A3 Canada
bremner@unb.ca

Dmitrii V. Pasechnik
Department of Computer Science
University of Oxford
Wolfson Building
Parks Road
Oxford OX1 3QD
United Kingdom
dima.pasechnik@cs.ox.ac.uk

Achill Schürmann
Institute of Mathematics
University of Rostock
18051 Rostock
Germany
achill.schuermann@uni-rostock.de

Mathieu Dutour Sikirić
Rudjer Bosković Institute
Bijenicka 54
10000 Zagreb
Croatia
mdsikir@irb.hr

Thomas Rehn
initOS GmbH & Co. KG
Hegelstraße 28
39104 Magdeburg
Germany
thomas.rehn@research.initos.com