Mathematisches Forschungsinstitut Oberwolfach

Report No. 38/2010

DOI: 10.4171/OWR/2010/38

# Mini-Workshop: Exploiting Symmetry in Optimization

Organised by
Volker Kaibel, Magdeburg
Leo Liberti, Palaiseau
Achill Schürmann, Delft
Renata Sotirov, Tilburg

August 22nd – August 28th, 2010

Abstract. The feasible regions of mathematical optimization models quite often exhibit a high degree of symmetry. In recent years, several groups of researchers have independently worked on algorithmic approaches to exploit such symmetries in a variety of contexts. Many of the techniques that have been developed are related or rely on similar computational tools. The workshop brought together researchers working on symmetry aspects in different areas of optimization. The exchange of state-of-the-art knowledge between these areas lead to identification of important directions for future research activities.

*Mathematics Subject Classification (2000):* 90Cxx, 52B15, 52B55, 05E18, 20B40.

## Introduction by the Organisers

If a mathematical optimization problem has many symmetries, then algorithms need to exploit this in order to be efficient. Here, *symmetry* usually means that a (large) group acts on the set of feasible solutions with the property that the objective function of the optimization problem is constant on every orbit of the action. Such symmetries can be inherent to the problem (e.g., automorphisms of some graph on which the problem is defined), but they can also be introduced by the way the problem is modeled.

Different strategies to cope with symmetric optimization problems have been developed over the last few years. One important approach is to break the symmetry of the solution space during a branch-and-bound algorithm by trying to avoid dynamically to enter equivalent copies of parts of the search space (e.g. by *isomorphism pruning* or by *orbital branching*). Another strategy is to enhance the model by additional constraints, preferably leaving only one representative out of every orbit of equivalent solutions. In case of the very important class of linear problems, the underlying geometric objects are convex polyhedra, whose algorithmic treatment (e.g., *polytope conversion*) in many cases needs sophisticated exploitation of symmetry as well. Quite often, the approaches heavily rely on efficient algorithms in computational group theory.

The aim of the workshop was to bring together researchers who are working on aspects of symmetry in areas such as mixed integer linear optimization, mixed integer and continuous nonlinear optimization (in particular semidefinite programming), algorithmic polytope theory, constraint programming, and computational group theory. The 17 participants, including four organizers, very well covered these topics. From the first day on the atmosphere of the workshop was very lively, the audience was actively participating in the presentations, and a lot of discussions were going on in smaller groups whenever there were no talks scheduled. In particular, there was a lot of exchange between researchers from the different communities. The doctoral students attending the workshop were very well integrated into these activities.

During the workshop, all participants and organizers gave presentations. The schedule was as follows. On Monday, there were two tutorials on computational group theory and the software package GAP, as well as three talks on research projects that aim at providing other tools for computational algebra or strongly rely on the use of GAP. The entire Tuesday was devoted to symmetries in integer linear optimization. The topics on Wednesday were polyhedral symmetries and exploitation of symmetries in computing Hilbert bases. On Thursday, nonlinear optimization as well as constraint programming were at the focus of the talks. The program on Friday started with two talks scheduled additionally upon special request of the participants. One of these talks was an online demonstration on how to use GAP to answer some questions that came up during an earlier talk. The program ended with an extensive discussion of open problems and future perspectives. Next to a lot of rather concrete requests with respect to integrating additional features into existing algebraic software tools, further research topics emerged from the discussion. Among those, the question for a detailed study to identify the symmetries arising in the most common libraries of optimization problems, new methods and paradigms for both automatically detecting symmetries in models as well as providing interfaces to the model creating user for enhancing models by information on symmetry, the continued development of alternative approaches, e.g., based on invariant subspaces or fundamental domains, and, just to name one more topic, investigations of symmetric triangulations and algorithms to construct them.

In view of the very stimulating interaction between the researchers during the workshop and of the presumably ongoing strong interest in symmetries in optimization problems (as demonstrated by the many open directions of future research), the workshop participants strongly agreed that a similar meeting in one or two years would be most desirable.

## Mini-Workshop: Exploiting Symmetry in Optimization

## Table of Contents

## Abstracts

### An Introduction to Computational Group Theory

Alexander Hulpke

The computation of orbits and stabilizers is a fundamental problem in computational group theory. While the ordinary orbit/stabilizer algorithm ( in priciple solves this task, it suffers from two issues that quickly grow worse as the orbit length/stabilizer index grows: Storing the orbit will become problematic, also the number of Schreier generators becomes large and requires either random selection (at the potential cost of unverified results) or rejection of redundant elements (requiring an element test in the prospective stabilizer subgroup). Software engineering methods, such as the use of subgroup orbits or so-called "tadpoles" — to every orbit element $\gamma$ we assign (by defining an "image" function based on hash values) a eventually periodic sequence of orbit elements, depending on $\gamma$. We only store sequence representatives — can push the feasibility limit somewhat further, but for better results more group theoretic information is necessary.

The first fundamental improvement is given by the use of backtrack algorithms for the search of stabilizers, representing the group via a stabilizer chain (which can be obtained from any sequence of actions). Still, the cost is related to the stabilizer index and can only be reduced if one can replace the group with an appropriate subgroup, for example by first stabilizing a (weaker) invariant of the object to be stabilized.

Two further approaches become possible by first computing the radical and associated normal subgroups of the radical factor. (Standard permutation group algorithms can do so, for matrix groups, the matrix group recognition project essentially provides such a structure, see [2].)

First, this data structure provides often a multitude of normal subgroups, which can be used in turn to use a normal-subgroup version of the orbit algorithm: Compute the orbit under $N$, then compute the action of $G$ on $N$-orbits, correcting the stabilizer of the $N$-orbit to stabilize the original element.

Secondly, such a data structure enables algorithms for computing (maximal) subgroups, in which one can compute the stabilizer first. Furthermore one can test whether a given maximal subgroup $U < G$ stabilizes an element $\gamma$, which can be mapped by $g \in G$ to $\omega$. (Such a mapping element can be found comparatively easily, thanks to the birthday paradox.) Then $U^g$ contains this stabilizer, and the stabilizer calculation can be done in $U$. The whole orbit is obtained from the $U$-orbit by mapping with coset representatives for $U$ in $G$.

Thirdly, in principle (in practice it requires knowledge of all subgroups of the simple groups) one can use a modification of the maximal subgroup algorithm to determine for a known subgroup $S$ of the stabilizer (e.g. from a partial run) all minimal supergroups $S < T < G$. Then testing whether any of these stabilize, one can verify whether $S$ is the full stabilizer, even if not the whole orbit is known.

A detailled description and discussion of many of the involved algorithms can be found in [1, 4] and – in more basic form – in the author's notes [3].

## References

[1] Derek F. Holt, Bettina Eick, and Eamonn A. OBrien, *Handbook of Computational Group Theory*, Chapman & Hall/CRC, 2005.

[2] Derek F. Holt, Mark J. Stather, *Computing a chief series and the soluble radical of a matrix group over a finite field*, LMS J. Comput. Math. 11 (2008), 223–251.

[3] Alexander Hulpke, *Notes on Computational Group Theory*, Summer 2010, available at `www.hulpke.com`.

[4] Ákos Seress, *Permutation group algorithms*, Cambridge University Press, 2003.

## PermLib – A C++ Library for Computations in Permutation Groups

Thomas Rehn

Many algorithms using permutation groups to solve problems in various areas of mathematics rely on methods from computational group theory. The usual way to work with permutation groups on computers is to represent them with bases and strong generating sets (BSGS). In this structure a backtrack search can be performed to search for certain subgroups of interest like setwise stabilizers or to decide whether two sets are in the same group orbit (cf. [1, 3]). These problems can usually be solved with standard computer algebra software like GAP [4] or Magma [5]. Because these software packages lack interfaces for external software the author's PermLib library offers a lightweight implementation of fundamental algorithms and data structures to work with permutation groups. PermLib is written in C++ for maximal interoperability and is released under an open source license. It currently provides computations of orbits and stabilizers and in-orbit checks. The performance of PermLib and the implemented algorithms are analyzed in the author's thesis [2].

For the end user the most important directory of the PermLib package is `include`. No installation is required to use PermLib, only its `include` directory has to be added to the user's project includes. However, PermLib uses parts of the Boost C++ library, which has to be installed. The file `permlib_api.h` in `include/permlib` offers an abstraction of the core functionality. It provides the data types `Permutation`, `PermutationGroup` and `OrbitAsSet` and the functions `construct`, `setStabilizer`, `setImage` and `orbits`. Examples covering all functionality can be found in the `example` directory of PermLib, especially in `api-example.cpp`. PermLib is available from [6].

## References

[1] D. F. Holt, B. Eick, and E. A. O'Brien, *Handbook of Computational Group Theory*, Chapman & Hall/CRC, 2005.

[2] T. Rehn, *Fundamental Permutation Group Algorithms for Symmetry Computation*, Diploma Thesis (Computer Science), Otto von Guericke University Magdeburg, 2010.

[3] Á. Seress, *Permutation Group Algorithms*, Cambridge University Press, 2003.

[4] GAP – Groups, Algorithms, Programming, `http://www.gap-system.org`.

[5] MAGMA Computational Algebra System, `http://magma.maths.usyd.edu.au`.
[6] PermLib, `http://fma2.math.uni-magdeburg.de/~latgeo/permlib/permlib.html`.

## A GAP/Sage package for computation with coherent configurations

Dmitrii V. Pasechnik
(joint work with Keshav Kini)

Coherent configurations (CCs, for short—terminology due to D. Higman) generalize the centralizer algebras of a permutation representations of finite groups and association schemes.

CC of *degree* $n$ and *dimension* $d$ is a collection $\mathcal{A} = \{A_1, \dots, A_d\}$ of $d$ 0-1 $n \times n$-matrices s.t.

(1) $\sum_{i=1}^d A_i = J_n$, the all-1 matrix.
(2) $\exists S \subset \mathcal{A}$: $I_n = \sum_{A \in S} A$.
(3) $A \in \mathcal{A}$, then $A^* := A^\top \in \mathcal{A}$.
(4) $\exists\ p_{i,j}^k$, $0 < i, j, k \leq d$, s.t. $A_i A_j = \sum_{k=1}^d p_{i,j}^k A_k$, $\forall i, j$.

$\mathcal{A}$ is a basis of a $*$-matrix algebra with identity. If $I \in \mathcal{A}$ then we have a (non-commutative) association scheme.

CCs find applications in finite permutation groups ($S$-rings, representations, sporadic groups, Moonshine, "synchronization", etc), coding theory (e.g. bounds on codes, sphere packings, etc), algebraic graph theory (e.g. distance-regular graphs), optimization ("Data compression" for various problems, in particular semidefinite and linear programming)

**Schurean CCs and their representations.** I. Schur thought that all CCs are coming from *orbitals* (aka 2-orbits) of permutation groups:

(1) given permutation group $G := (G, \Omega)$, with $|\Omega| = n$,
(2) for $x, y \in \Omega$; define $O_{xy} := \{(x^g, y^g) \mid g \in G\}$ (orbit on 2-tuples))
(3) define matrix $A := A(O_{xy})$ — the adjacency matrix of the graph with vertex set $\Omega$ and edge set $O_{xy}$;
(4) take $\mathcal{A} = \mathcal{A}(G) := \{A(O_{xy}) \mid x, y \in \Omega\}$

True for $n < 15$ – every CC is like this, but gets wrong by a large margin as $n \to \infty$: e.g. for $n = 36$ there are $\gg 10^3$ of CCs (even if $d = 3$), and only a handful of such $G$'s.

**Remark.** Given a (vertex and edge) weighted graph $\Gamma$ with adjacency matrix $A(\Gamma)$, there is unique minimal (w.r.t. $d$) CC s.t. $A(\Gamma) = \sum_{A \in \mathcal{A}} c_A A$, computable by *Weisfeiler-Leman stabilization* in polynomial time. If Schur were right we would have had a polynomial-time algorithm for graph isomorphism (Weisfeiler-Leman stabilization would be recognizing isomorphic graphs).

Let $\mathbb{C}[\mathcal{A}] = \{\sum_{A \in \mathcal{A}} c_A A \mid c_A \in \mathbb{C}\} \subseteq M_n(\mathbb{C})$ be the $\mathbb{C}$-algebra generated by $\mathcal{A}$. $\mathbb{C}[\mathcal{A}]$ is semi-simple, i.e.

$$\mathbb{C}[\mathcal{A}] \cong \oplus_{j=1}^{\overline{d}} M_{d_j}(\mathbb{C}), \qquad \text{(as a $*$-algebra).}$$

In the group case, when $\mathcal{A} = \mathcal{A}(G)$, and $\pi : G \to M_n(\mathbb{C})$ the permutation representation of $G$, we have

$$\mathbb{C}[\mathcal{A}] \cong \oplus_{j=1}^{\overline{d}} M_{m_\chi}(\mathbb{C}), \quad \text{where } \pi = \sum_{\chi \in \mathrm{Irr}(G)} m_\chi \chi.$$

One of goals of the package is to compute $*$-representations $\phi_\chi : \mathbb{C}[\mathcal{A}] \to M_{m_\chi}(\mathbb{C})$, i.e. $\phi_\chi(A)^* = \phi_\chi(A^*)$, $\forall A \in \mathcal{A}$.

It is easy to construct the regular ($*$-)representation of $\mathcal{A}$. The map

$$\mathrm{reg} : \mathbb{C}[\mathcal{A}] \to M_d(\mathbb{C}), \quad A_i \mapsto B_i,\ (B_i)_{kj} = p_{ij}^k$$

is an algebra isomorphism, and $d = \sum_{\chi \in \mathrm{Irr}(G)} m_\chi^2$. (This is true for any subalgebra of $M_n(\mathbb{C})$).

Sometimes (e.g. in optimization applications) we need a $*$-isomorphism. Scaling $A \in \mathcal{A}$

$$\overline{A} := \mathrm{Tr}(AA^*)^{-1/2} A \quad \text{ and setting } \overline{A}_i \overline{A}_j = \sum_{k=1}^{d} \overline{p}_{ij}^k \overline{A}_k$$

we obtain

$$\mathrm{reg} : \mathbb{C}[\mathcal{A}] \to M_d(\mathbb{C}), \quad \overline{A}_i \mapsto \overline{B}_i,\ (\overline{B}_i)_{kj} = \overline{p}_{ij}^k.$$

Note that $\overline{B}_i$'s can contain quadratic irrationals.

We can factor reg into $\mathrm{reg}_\chi$:

$$\mathbb{C}[\mathcal{A}] \xrightarrow{\mathrm{reg}} M_d(\mathbb{C}) \xrightarrow{\mathrm{reg}_\chi} M_{m_\chi^2}(\mathbb{C}),$$

where $\mathrm{reg}_\chi$, for $\chi \in \mathrm{Irr}(G)$, is given by the projector, certain weighted sum of conjugacy classes of $\pi(G)$:

$$\pi_\chi := \sum_{g \in \mathcal{G}} \chi(g^{-1}) \sum_{h \in g^G} \pi(h) \in C(\mathbb{C}[\mathcal{A}]),$$

so that $\pi_\chi \mathbb{C}[\mathcal{A}] \cong M_{m_\chi}(\mathbb{C})$. This is easy when the character table of $G$ and a transversal $\mathcal{G}$ of representatives of conjugacy classes $g^G$ are available.

It is easy to compute generators for the centre of $\mathbb{C}[\mathcal{A}]$, namely, $\sum_{h \in g^G} \pi(h)$, for $g \in \mathcal{G}$ (they generate $C(\mathbb{C}[\mathcal{A}])$). The naive summing over $g^G$ is way too slow. However, we can find

$$\sum_{h \in g^G} \pi(h) = \sum_{A \in \mathcal{A}} \alpha_A A$$

directly, by viewing $g \in \mathcal{G}$ as $h \in \mathrm{Aut}(\Gamma(A))$, where $\Gamma(A)$ means the digraph defined by $A$, and noting that $\alpha_A = |g^G| \,\mathrm{Tr}(\pi(g)A)$. This is implemented in our GAP [1] package [3] and appears to work very well.

We already explained how to compute $\pi_\chi$, providing

$$\pi_\chi \mathbb{C}[\mathcal{A}] \cong M_{m_\chi}(\mathbb{C}) \cong M_{m_\chi}(\mathbb{C}) \otimes I_{m_\chi} \subset M_{m_\chi^2}(\mathbb{C}).$$

Further, we employ the following heuristic.

- Assume we know $X = X^* \in \pi_\chi \mathbb{C}[\mathcal{A}]$ such that the $f(t) = \mathrm{charpol}(X)$ factors over the splitting field of $G$ into linear factors $(t-\rho_1), \ldots, (t-\rho_{m_\chi})$, so that $\rho_i \neq \rho_j$ for $i \neq j$.
- Take any $X$-eigenvectors $v_1, \ldots, v_{m_\chi}$ of the eigenvalues $\rho_1, \ldots, \rho_{m_\chi}$.
- Then $v_1, \ldots, v_{m_\chi}$ span a $m_\chi$-dimensional submodule of $\pi_\chi \mathbb{C}[\mathcal{A}]$.
- Compute this action and return it.

Works for many examples. It is currently an open question whether one can do better, without that factoring assumption.

**An example application: Lovasz-Schrijver bound $\theta'(\Gamma)$.** Let $\Gamma = (V, E)$ be a graph, $n = |V|$, $G \leq \mathrm{Aut}(\Gamma)$. Then for $\mathcal{A} = \mathcal{A}(G) = \{A_1, \ldots, A_d\}$ there is $S \subset \{1, \ldots, d\}$ s.t. $A(\Gamma) = \sum_{s \in S} A_s$. Introduce a variable $y_k$ for each $k \notin S$ and require that

$$\sum_{k \in D} y_k \langle A_k, A_k \rangle = 1, \quad \text{where } \sum_{k \in D} A_k = I_n. \tag{1}$$

$$\sum_{k \notin S} y_k \phi(A_k) \text{ is positive semidefinite,} \tag{2}$$

where $\phi$ is a faithful $*$-representation of $\mathcal{A}$, e.g. reg. Then

$$\theta'(\Gamma) := 1 + \max_{y} \sum_{k \notin S \cup D} y_k \langle A_k, A_k \rangle, \quad y \geq 0 : (1) \text{ and } (2)$$

gives an upper bound, due to Lovasz and Schrijver, on $\alpha(\Gamma)$, the size of a maximum coclique in $\Gamma$. The lower the dimension of $\phi$, the more tractable the computation, solving this *semidefinite programming problem*, becomes. In order to provide an easy interface between our GAP package [2] and an SDP solver, we use Sage [4].

## References

[1] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4.12*, 2008. (http://www.gap-system.org).

[2] D. Pasechnik and K. Kini. Cohcfg, a GAP package for coherent configurations. software download, 2010. http://www1.spms.ntu.edu.sg/~dima/software/cohcfg-a1.tgz.

[3] D. Pasechnik and K. Kini. A GAP package for computation with coherent configurations. In K.Fukuda, J.v/d.Hoeven, M.Joswig, and N.Takayama, editors, *Mathematical Software ICMS 2010, Proceedings*, volume 6327 of *Lecture Notes in Comput. Sci.*, pages 69–72. Springer, Berlin, 2011.

[4] W. Stein et al. *Sage Mathematics Software (Version 4.4)*. The Sage Development Team, 2010. (http://www.sagemath.org).

## A GAP package for polytope and lattice computations using symmetries

Mathieu Dutour Sikirić

A polytope can be described alternatively as the convex hull of its finite set of vertices or as the intersection of its finite set of facet defining inequalities. The process of obtaining one description from the other is named *dual description problem* and is an essential step in many algorithms. We propose here some methods for computing such dual description in the cases where the polytope is highly symmetric.

The key such method is the computation of the set of orbits by the adjacency decomposition method [2, 8]. It allows an optimal use of the existing symmetries and allowed the solution of several famous problems [4, 7, 3]. It is based on existing software for polyhedral computations [14, 13], on `nauty` [16] for graph computation and `GAP` [15] for group theoretic computations. Running the adjacency decomposition method implies computing the set of facets adjacent to a given facet. This computation is precisely a dual description computation and so we may apply the adjacency decomposition method recursively when this becomes too complicated. The problem is that the number of cases may grow too fast. We used three methods for dealing with this problem:

(1) One is to connectivity results like Balinski theorem to prove that we do not need to go any further in the computation.
(2) Another is to use a banking system to store known dual descriptions and check in advance before computing one.
(3) Another is to use the specific symmetry group of a face, which might be larger than the stabilizer of the face.

So, one needs efficient methods for computing symmetry groups. For a given polyhedral cone $C \subset \mathbb{R}^n$ given by generating rays $(v_i)_{1 \le i \le N}$ one can define three possible symmetry groups:

(1) Combinatorial symmetry group $Comb(C)$: this is the group of transformations $\sigma \in \mathrm{Sym}(N)$ preserving the set of faces of $P$ globally.
(2) Projective symmetry group $Proj(C)$: this is the group of transformations $\sigma \in \mathrm{Sym}(N)$ such that there exist $\alpha_\sigma > 0$, $A \in \mathrm{GL}_n(\mathbb{R})$ with $Av_i = \alpha_\sigma(i) v_{\sigma(i)}$.
(3) Linear symmetry group $Lin(C)$: this is the group of transformations $\sigma \in \mathrm{Sym}(N)$ such that there exist $A \in \mathrm{GL}_n(\mathbb{R})$ with $Av_i = v_{\sigma(i)}$.

Essentially the only general way to obtain $Comb(C)$ is to compute the set of facets of $C$ which is precisely the problem we want to solve. The linear symmetry group can be computed efficiently by defining an edge colored graph for which we can apply `nauty` [7, 8]. Finding a general method for computing the projective symmetry group is unfortunately an open problem.

One of the standard application of the recursive adjacency decomposition method is the computation of the Delaunay tesselations of lattices [3]. Using this we can solve in a standard way many computations of rigidity degree, free vector, quantization constant, etc.

Another context where ideas reminiscent of adjacency decomposition can be applied is the determination of polyhedral tesselations. Currently we can deal with perfect forms [7] and L-type [11], with or without symmetry being specified. Recently some extension of the software have been done towards working with polytopes with coordinates in an algebraic number field [9].

The program polyhedral is also useful for computing with face lattice and flag systems [5, 6]. It can also compute the volume of polytope [1], which proved useful in deriving classical mathematical proofs [10].

Besides its polytopal features, the polyhedral software [12] also has some combinatorial and lattice enumeration functions.

## References

[1] B. Büeler, A. Enge, K. Fukuda, *Exact volume computation for polytopes: a practical study*, pages 131–154 in *Polytopes—combinatorics and computation (Oberwolfach, 1997)*, Birkhäuser, 2000.

[2] T. Christof and G. Reinelt, *Combinatorial optimization and small polytopes*, Top (Spanish Statistical and Operations Research Society), **4** (1996) 1–64.

[3] M. Dutour Sikiric, A. Schuermann, F. Vallentin, *Complexity and algorithms for computing Voronoi cells of lattices*, Mathematics of computation 78 (2009), 1713–1731.

[4] M. Dutour Sikiric, A. Schuermann, F. Vallentin, *The contact polytope of the Leech lattice*, accepted in Discrete and Computational Geometry.

[5] M. Deza, M. Dutour Sikiric, S. Shpectorov, *Hypercube Embedding of Wythoffians*, Ars Math. Contemp. 1 (2008), 99–111.

[6] M. Dutour, G. Ellis, *Wythoff polytopes and low-dimensional homology of Mathieu groups*, Journal of Algebra 322 (2009) 4143–4150.

[7] M. Dutour Sikiric, A. Schuermann, F. Vallentin, *Classification of eight dimensional perfect forms*, Electron. Res. Announc. Amer. Math. Soc. 13 (2007), 21–32.

[8] D. Bremner, M. Dutour Sikiric, A. Schuermann, *Polyhedral representation conversion up to symmetries*, CRM proceedings, volume 48 (2009) 45–72.

[9] M. Dutour Sikirić, *The dual description of $W(H_4)$*, in preparation.

[10] M. Dutour Sikiric, A. Schuermann, F. Vallentin, *Inhomogeneous extreme forms*, preprint.

[11] A. Schuermann, M. Dutour Sikiric, F. Vallentin, *A generalization of Voronoi's reduction theory and its application*, Duke Math. J. 142 (2008), 127–164.

Software

[12] M. Dutour Sikirić, *Polyhedral package*, `http://www.liga.ens.fr/~dutour`

[13] D. Avis, *A C-implementation of the reverse search vertex enumeration algorithm*, School of Computer Science, McGill University, Montreal, Canada 1993, `http://www-cgrl.cs.mcgill.ca/~avis/C/lrs.html`.

[14] K. Fukuda, *cdd+ reference manual*, Institute for operations research, Swiss Federal Institute of Technology, Zurich, Switzerland, 1995, `http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html`.

[15] The GAP Group, GAP — Groups, Algorithms, and Programming, Version 4.4.6; 2005. `http://www.gap-system.org`.

[16] B.D. McKay, *The nauty program*, `http://cs.anu.edu.au/people/bdm/nauty/`.

## Solving Symmetric Integer Programs

Jeff Linderoth
(joint work with François Margot, Jim Ostrowski, Fabrizio Rossi, Stefano Smriglio, Greg Thain)

We will discuss mechanisms for dealing with integer programs that contain a great deal of symmetry. The methods use information encoded in the symmetry group of the integer program to guide the branching decision and prune nodes of the search tree. We will discuss orbital branching [5], isomorphism pruning [2, 3], and new flexible variants of isomorphism pruning. Some of these methods have been recently incoporated into commercial IP software. We will conclude with a brief discussion of powerful computing platforms known as computational grids and instances of using these platforms for solving symmetric integer programs [4, 1].

### References

[1] J. Linderoth, F. Margot, and G. Thain. Improving bounds on the football pool problem via symmetry reduction and high-throughput computing. *INFORMS Journal on Computing*, 21:445–457, 2009.

[2] F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94:71–90, 2002.

[3] F. Margot. Exploiting orbits in symmetric ILP. *Mathematical Programming, Series B*, 98:3–21, 2003.

[4] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Constraint orbital branching. In *IPCO 2008: The Thirteenth Conference on Integer Programming and Combinatorial Optimization*, volume 5035 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2008.

[5] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. *Mathematical Programming*, 2010. to appear.

## Orbitopes and Orbitopal Branching

Volker Kaibel
(joint work with Yuri Faenza, Andreas Loos, Matthias Peinhardt, Marc Pfetsch)

An *orbitope* is the convex hull of all 0/1-matrices that are lexicographically maximal within their orbits under some group action that works via permuting columns. We distinguish between *full orbitopes* (no restrictions) and *packing-*, *partitioning-*, as well as *covering-orbitopes*, where for the latter ones only matrices with at most, exactly, or at least one, respectively, 1-entry per row are considered. Investigations of these orbitopes have been started a few years ago in order to understand those linear inequalities that can be used for lexicographical symmetry breaking in certain integer programming models, like, e.g., common models of graph partitioning problems.

In the first part of this talk, we survey the most important results that have been obtained on these polytopes, mainly focussing on the case of the full symmetric group acting on the columns of the matrices. In this case, a 0/1-matrix (with a feasible number of ones in their rows) is a vertex of the respective orbitope if and only if its columns come in lexicographically non-decreasing order. The main results are the following: For full orbitopes we can provide polynomial size extended formulations [2], but no linear descritpion in the original space seems to be in sight, except for the case of matrices with two columns only, for which a complete description is known. For the packing- and for the partitioning cases, we have both complete descriptions in the original space [4] (by means of *shifted-column-inequalities*) as well as very small (linear size) extended formulations [1]. For the covering case none of this can be expected as the liner optimization problem over these polytopes turns out to be NP-hard.

In the second part of the talk we describe the method of *orbitopal fixing* [3] that allows to exploit knowledge on partitioning orbitopes within the context of branch-and-cut algorithms without explicitly adding any constraints to the formulation. Instead, at every node of the branch-and-bound-tree a linear time procedure is applied that performs simultaneous fixing with respect to the orbitope. We briefly discuss some computational results for graph partitioning problems indicating that the method is quite effective.

## References

[1] Y. Faenza and V. Kaibel, *Extended Formulations for Packing and Partitioning Orbitopes*, Math. Oper. Res. **34**, 2009, pp. 686–697

[2] V. Kaibel and A. Loos, *Branched polyhedral systems*, Proceedings of IPCO XIV (F. Eisenbrand and B. Shepherd, eds., LNCS, vol. 6080, Springer-Verlag, 2010, pp. 177–190.

[3] V. Kaibel, M. Peinhardt, and M. E. Pfetsch, *Orbitopal fixing*, Proceedings of IPCO XII (M. Fischetti and D. Williamson, eds.), LNCS, vol. 4513, Springer-Verlag, 2007, pp. 74–88.

[4] V. Kaibel, M. Peinhardt, and M. E. Pfetsch, *Packing and partitioning orbitopes*, Math. Programming, Ser. A **114** (2008), no. 1, 1–36.

## Symmetries in linear and integer linear programming

Katrin Herr
(joint work with Richard Bödi, Michael Joswig)

We investigate the potential of symmetry in the context of linear and integer linear programming. A *symmetry* of a linear program LP$(A, b, c)$ of the form

$$\begin{array}{ll} \max & c^t x \\ \text{s.t.} & Ax \leq b\,,\ x \in \mathbb{R}^n \end{array}$$

is a linear transformation which leaves the feasible region $P(A, b) := \{x \in \mathbb{R}^n \mid Ax \leq b\}$ invariant, only inducing a permutation of the rows of the matrix $A$, and which does not change the utility value $c^t x$ of any feasible point. Symmetries of integer linear programs additionally need to preserve the standard lattice $\mathbb{Z}^n$, which confines the set of potential symmetries of bounded and full-dimensional integer linear programs to $\mathrm{O}_n\mathbb{Z}$, the group of signed permutation matrices. Notice that this definition of symmetry is based on the description of the problem by linear

inequalities, although this description might disguise some of the symmetries of the set of feasible integer points.
In the following we consider symmetries of linear and integer linear programs that are elements of $\mathrm{O}_n\mathbb{Z}$, which act on the signed standard basis $\{\pm e_1, \pm e_2, \ldots, \pm e_n\}$ of $\mathbb{R}^n$ as signed permutations. Throughout let $\Gamma$ be a subgroup of $\mathrm{O}_n\mathbb{Z}$. Then $\Gamma$ splits the signed standard basis into disjoint orbits. There are two kinds of orbits to distinguish: The *bipolar* orbits contain at least one pair $\pm e_i$, while the *unipolar* orbits do not. Since $\Gamma$ is a linear group, a signed permutation $\sigma \in \Gamma$ with $\sigma e_i = \epsilon e_j$ and $\epsilon \in \{\pm\}$ maps $-e_i$ to $-\epsilon e_j$. Hence, a bipolar orbit $O$ only consists of pairs, that is, $-O = O$. On the other hand, for each unipolar orbit $O$ the set $-O = \{-e_i \,|\, e_i \in O\}$ forms another orbit, and $\Gamma$ acts equivalently on $O$ and $-O$. An action is called *semi-transitive* if there exists an orbit representing a basis of $\mathbb{R}^n$. The main idea for the exploitation of symmetry in linear programming is to efficiently restrict the linear program to a certain subspace which is known to contain an optimal solution. The convexity of the feasible region $P(A, b)$ guarantees this property for the *fixed space* of $\mathbb{R}^n$ given by

$$\mathrm{Fix}_\Gamma(\mathbb{R}^n) := \{x \in \mathbb{R}^n \;|\, \gamma x = x \text{ for all } \gamma \in \Gamma\} \ .$$

To use this knowledge in an efficient way, it is necessary to provide an appropriate characterization of the fixed space. To this end, we define a vector $\beta_O := \sum_{v \in O} v$ for each orbit $O$ of the orbit decomposition. Then $\beta_O$ spans the fixed space $\mathrm{Fix}_\Gamma(\mathrm{lin}_\mathbb{R}(O))$. Notice that $\mathrm{Fix}_\Gamma(\mathrm{lin}_\mathbb{R}(O))$ is zero-dimensional for bipolar orbits and one-dimensional for unipolar orbits. A characterization of the fixed space of $\mathbb{R}^n$ is now given by

$$\mathrm{Fix}_\Gamma(\mathbb{R}^n) \;=\; \mathrm{lin}_\mathbb{R}\{\beta_O \;|\, O \text{ orbit of } \Gamma\} \;=\; \mathrm{lin}_\mathbb{R}\{\beta_O \;|\, O \text{ unipolar orbit of } \Gamma\} \ .$$

In fact, it suffices to choose one orbit per pair of unipolar orbits since $\beta_O = -\beta_{-O}$. The characterization allows for projection onto the fixed space, and the dimension of the resulting linear program is equal to half of the number of unipolar orbits. In the semi-transitive case, we already receive a one-dimensional problem.

We now turn to integer linear programming. Notice that by restricting the feasible region of a linear program to the standard lattice we lose convexity, and with it the guarantee for an optimal solution in the fixed space. For the exploitation of symmetry in integer linear programming we focus on affine hyperplanes orthogonal to the utility vector $c$ that contain at least one integer point; we will call them *$c$-layers*. The $c$-layers between the origin and the point $c$ are characterized by

$$H_{c,\frac{k}{||c||^2}} = \ker\,(x \mapsto c^t x) + \frac{k}{||c||^2} c\,, \;\; k = 0\,, \ldots, ||c||^2\,.$$

In particular, the number of $c$-layers between the origin and the point $c$ is finite, and, in the case of a semi-transitive action, it equals the dimension of the integer linear program. A straightforward procedure is to check IP-feasibility of each $c$-layer in descending order with respect to the utility value, starting with the $c$-layer next to the solution of the LP-relaxation. For general infeasible integer linear programs this procedure does not stop. However, for integer linear programs with a symmetry group acting semi-transitively on the signed standard basis the following criterion reveals infeasibility. A *center* of a $c$-layer is defined as the intersection point of the $c$-layer with the line spanned by the utility vector $c$. For semi-transitive actions this line is equal to the fixed space. Therefore, the center is the only element of the restriction of the linear program to the $c$-layer and the fixed space. Thus, the $c$-layer is LP-feasible if and only if its center is LP-feasible. Due to the convexity of the linear program, the LP-infeasibility of one center implies the LP-infeasibility of the $c$-layers with smaller utility values. Hence, in the semi-transitive case, we can stop the procedure described above after having tested the first integral center, and state IP-infeasibility of the problem if no feasible integer point has been found during the procedure up to this point in time. For integer linear programs with highly transitive symmetry groups we can check IP-feasibility of a $c$-layer efficiently: A *core point* is an integer point with minimal Euclidean distance to the center of a $c$-layer. Given an integer linear program of dimension $n$ with a $(\lfloor\frac{n}{2}\rfloor + 1)$-transitive symmetry group (i.e., isomorphic to $\mathcal{A}_n$ or $\mathcal{S}_n$ for $n \geq 5$) a $c$-layer is feasible if and only if all core points are feasible. Therefore, we can restrict the set of integer points that need to be tested to the core points. Moreover, a $(\lfloor\frac{n}{2}\rfloor)$-transitive symmetry group acts transitively on the set of core points. Hence, it suffices to test the IP-feasibility of one core point per $c$-layer. Conclusively, for symmetry groups isomorphic to $\mathcal{A}_n$ or $\mathcal{S}_n$, where $n$ is the dimension of the integer linear program, the algorithm tests the feasibility of one core point in each of the at most $n$ $c$-layers, i.e., the algorithm is linear in the number of dimensions.

## References

[1] S. P. Fekete and J. Schepers. A combinatorial characterization of higher-dimensional orthogonal packing. *Mathematics of Operations Research*, 29:353–368, 2004.

[2] E. J. Friedman. Fundamental domains for integer programs with symmetries. In *Combinatorial Optimization and Applications, First International Conference, COCOA 2007, Proceedings*, pages 146–153.

[3] J. E. Humphreys. *Reflection groups and Coxeter groups.* Cambridge Studies in Advanced Mathematics 29. Cambridge: Cambridge University Press. XII, 204 p. , 1992.

[4] V. Kaibel, M. Peinhardt, and M. E. Pfetsch. Orbitopal fixing. In *Integer Programming and Combinatorial Optimization, 12th International Conference, IPCO 2007, Proceedings*, pages 74–88.

[5] V. Kaibel and M. Pfetsch. Packing and partitioning orbitopes. *Math. Program.*, 114(1):1–36, 2008.

[6] F. Margot. Pruning by isomorphism in branch-and-cut. *Math. Program.*, 94(1):71–90, 2002.

[7] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. In *Integer Programming and Combinatorial Optimization, 12th International Conference, IPCO 2007, Proceedings*, pages 104–118.

## Symmetry in Scheduling Problems

James Ostrowski
(joint work with M.F. Anjos, A.Vannelli)

Symmetry has been an obstacle in mixed integer linear programming (MILP) for more than 40 years. In [4], Jeroslow presented a class of problems with only one equality constraint on $n$ variables where branch-and-bound trees contain an exponential number nodes if symmetry is not removed from the problem. Around the same time period [3] suggested that a class of highly symmetric covering problems called Steiner Triple Systems (STS) be included in test libraries because they were notoriously difficult, especially considering the small number of variables in the problem. In the past decade, effective symmetry breaking techniques have been developed for MILP problems. Symmetry breaking methods such as isomorphism pruning [6] are able to remove all symmetries from the branch-and-bound tree. While we know how to break symmetry, we do not yet have a clear understanding of the most effective way to break symmetry, and more importantly, how symmetry-breaking methods interact with other MILP features such as branching strategies and cutting plane methods.

To better study how symmetry breaking affects integer programming techniques, we examine symmetry breaking in scheduling problems. Scheduling problems cover a very broad class of problems with important real world applications. The structure of symmetry present in these problems allow for efficient symmetry breaking techniques. It is this structure, also found in bin-packing and graph coloring problems, that makes them ideal candidates for our study.

We discuss the relationship between symmetry-breaking techniques and branching strategies in reducing computation time. If good branching strategies are known a priori, symmetry breaking constraints that augment the branching strategy can be added to the problem with great result. However, if branching strategies are not known, adding symmetry-breaking constraints to the problem formulation may not be the best option. In this case, orbital branching- a symmetry breaking method that removes symmetry *during* the branch-and-bound process [8], is shown to be most effective.

The addition of symmetry breaking constraints like in [1] and orbitopal fixing [5] restrict the set of feasible solutions to be the matrices $x$ with lexicographically decreasing columns. The difference between the methods is that [1] restricts the feasible region by explicitly adding inequalities, while orbitopal fixing uses these inequalities implicitly to fix variables. Explicitly adding these inequalities may remove fractional solutions that would have been optimal in the LP relaxation, resulting in better LP relaxations and thus, smaller branch-and-bound trees. Another advantage of the inequality method over orbital branching is that commercial solvers may be able to use these inequalities to generate stronger cutting planes. The cost of the improved relaxations is that the additional inequalities make the LP relaxations more computationally intensive. Also, as many optimal solutions may be made infeasible by the constraints, finding an optimal solution may be more difficult.

Because the minimal fundamental domain is defined a priori, both the constraint version of the formulation and orbitopal fixing can perform more fixing than orbital branching.

While there are many applications, we provide computational results for the deterministic operating room (OR) scheduling problem. In this application there is a set of $m$ blocks of surgeries planned for the day and a set of $r$ available operating rooms. It is assumed that the operating rooms are identical and that the time required for each surgery block is known. Using an operating room incurs a one-time fixed cost. If the time required to perform the blocks assigned to a particular operating room is over a predefined limit an overtime cost must be paid for every additional hour. The OR scheduling problem is to find the minimum cost allocation of patients to operating room.

There can be more than one minimal fundamental domain. Does it matter which fundamental domain is used? Reducing the feasible region to a fundamental domain can allow for more opportunities to fix variables. It is these fixings that make symmetry-exploiting tools like orbital branching and orbitopal fixing powerful. Ideally, it is preferable to fix variables with respect to symmetry as early in the tree as possible. When the fundamental domain is generated by choosing only lexicographical minimal solutions, i.e. the fundamental domain described by adding the lex-ordering constraints, this additional fixing is done when branching on variables with small row indices. For example, suppose there were 100 surgery blocks and 25 operating rooms. Fixing a variable $x_{100,j}$, for any $j$, as a result of a branching disjunction does not strengthen any lexicographic inequality. Branching on the variable $x_{i,j}$ for any small $i$ and any $j$, does strengthen lex inequalities and leads to additional fixings. For example, fixing $x_{2,2}$ to zero as a result of branching (either by branching on $x_{2,2}$ directly, or by fixing $x_{2,1}$ to one) also allows for the fixing of $x_{3,3}$, $x_{4,4}$, ... $x_{k,k}$ to zero.

Because restriction of the feasible region to the lexicographic fundamental domain strengthens the branching disjunctions associated with variables of small row indices, it is important (for symmetry considerations) to branch on variables with a small row index early in the branch-and-bound tree. However, from an integer programming perspective, the choice of branching variables is very important, and can have a significant effect on the size of the branch-and-bound tree. Good branching candidates from an IP perspective are those that improve the LP relaxation the most. What if variables that are good branching candidates from a symmetry point of view are bad candidates from an IP point of view? This can be remedied by a better choice of fundamental domain.

An intuitive and effective branching strategy for bin packing type problems like the OR scheduling problem is to first branch on items with the largest weight, in this case the largest surgery time. Given this branching strategy, a fundamental domain that increases the importance of surgery blocks with larger completion time can be created. This is done by reindexing the variables such that the $d_i$

terms, the completion time for each surgery, form a decreasing sequence. Now, instead of representing a random surgery block, variable $x_{1,j}$ indicates whether the block with the largest completion time is performed in room $j$.

As orbital branching creates the minimal fundamental domain throughout the branching process, it is not necessary to determine how to best reindex variables. In theory, reindexing should not affect the overall performance of orbital branching, as it uses symmetry to strengthen CPLEX's branching, something that should not be dependent on variable indices. However, in the case of tie-breaks, indices may play a role in determining what variable is chosen for branching.

Computational results for the reindexed problems show reindexing variables leads to significant improvements in computation time for both the constraint method as well as orbitopal fixing. With orbital branching, reindexing the variables does affect the results, but not in a predictable way.

As is shown by the results, choosing an appropriate fundamental domain is important if one wishes to break symmetry by adding constraint or by orbitopal fixing. Doing so seems to lead to the best possible solution times. However, in order to construct an ideal fundamental domain, good branching strategies must be known a priori. If the branching strategy is not know a priori, orbital branching is the more effective choice in solving these problems. One method for determining a branching strategy for general MILP problems is discussed in [7]. The idea is to perform a dive in the branch-and-bound tree using strong branching to choose branching variables. The variables are then reindexed to reflect the order in which they were branched upon during the initial dive. The minimal fundamental domain used is the one generated by the lexicographically minimal solutions in the reindexed problem. Ideally strong branching will branch on variables that influence the LP bound the most first, so these variables should be given smaller indices.

## References

[1] B. Denton, A. Miller, H. Balasubramanian, and T. Huschka. Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations Research*, accepted, 2009.

[2] S. Gul, B. Denton, J. Fowler, and T. Huschka. Bi-criteria scheduling of surgical services for an outpatient procedure center. *Working Paper*, 2010.

[3] D. R. Fulkerson, G. L. Nemhauser, and L. E. Trotter. Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triples. *Mathematical Programming Study*, 2:72–81, 1973.

[4] R. Jeroslow. Trtivial integer programs unsolvable by branch-and-bound. *Mathematical Programming*, 6:105–109, 1974.

[5] V. Kaibel, M. Peinhardt, and M.E. Pfetsch. Orbitopal fixing. In *IPCO 2007: The Twelfth Conference on Integer Programming and Combinatorial Optimization*, pages 74–88. Springer, 2007.

[6] F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94:71–90, 2002.

[7] F. Margot. Exploiting orbits in symmetric ILP.

[8] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Orbital branching. *Mathematical Programming*, 2010. To appear. *Mathematical Programming, Series B*, 98:3–21, 2003.

## The Maximum $k$-Colorable Subgraph Problem and Symmetry

Marc E. Pfetsch
(joint work with Tim Januschowski, Cork University)

Given an undirected graph and some positive integer $k$, the Maximum $k$-Colorable Subgraph Problem is to select an induced subgraph of largest cardinality, which is k-colorable, i.e., one can assign one of $k$ colors to each node such that adjacent nodes receive different colors. This problem has a straight-forward assignment integer programming formulation. It is symmetric with respect to the columns, i.e., color classes. Thus, one can use so-called orbitopes to remove these symmetries. Orbitopes were introduced in [3] in order to handle symmetry in problems that have an assignment structure as in the maximum $k$-colorable subgraph problem.

This talk will mainly deal with the intersection of the polytope corresponding to the Maximum $k$-Colorable Subgraph Problem with (packing) orbitopes. The study of this intersection has been started by Januschowski [1].

The maximum $k$-colorable subgraph problem is strongly connected to the coloring problem and the stable set problem and has rarely been studied in the literature. We use it as a prototype of a problem where we can combine problem-specific with symmetry handling structure. One of the messages of this talk is that the interaction of these structures can have complicated effects on the facet structure.

In this talk, we show that the LP-relaxation remains weak, even when adding the complete description of the corresponding orbitope. However, several facets of one of the two polytopes can be changed to yield facets of the intersection. This includes so-called shifted-column inequalities, which in some cases yield facets and in some cases can be strengthened to facet-defining so-called clique shifted column inequalities. We give a characterization of these different cases. It turns out, that the conditions for these inequalities to be facet-defining depend on the labeling of the graph and are usually quite complicated.

We will complement these results with preliminary computational experiments. These computations show that handling symmetries is important, where the variant that uses orbitopal fixing is slightly superior to variants that additionally or only use shifted column inequalities. Orbitopal fixing was introduced in [2] and allows to use the orbitope structure in order to additionally fix variables at the nodes of the branch-and-bound tree. Additional cutting planes can help to reduce the number of nodes, but not the computation time. We also study the effects of turning off cutting planes, graph symmetry constraints, and of changing the order of the nodes. In all cases the performance deteriorates with turning of cuttings planes having the largest negative impact.

### References

[1] T. Januschowski, *Symmetry breaking for the maximal k-colourable subgraph problem*, Diploma Thesis, TU Berlin, 2007.

[2] V. Kaibel, M. Peinhardt, and Marc E. Pfetsch, *Orbitopal Fixing*, Proc. of the 12th Integer Programming and Combinatorial Optimization conference (IPCO), M. Fischetti and D. Williamson (eds.), Springer-Verlag, LNCS 4513, pp. 74–88.

[3] V. Kaibel and M. E. Pfetsch, *Packing and Partitioning Orbitopes*, Math. Program. 114, 1–36 (2008)

## Orbitwise polyhedral representation conversion

David Bremner

(joint work with Thea Gegenberg, Achill Schürmann, Mathieu Dutour Sikirić, Gordon Williams)

Converting between the inequality and finite-generator representations of a convex polyhedron is both a fundamental problem in algorithmic geometry and a useful subroutine in various optimization techniques. Unfortunately many problems of interest remain out of reach for current conversion methods. In certain applications the output is both large and symmetric. This has motivated study of the problem of *orbitwise* representation conversion: instead of producing all of the output, one looks for at least one element in each orbit (under some natural symmetry group).

Before considering the orbitwise problem, it is worth having a high level understanding of the strengths and weaknesses of existing methods for the symmetry-free version of the problem (cf. [1]). There are three main techniques (and some others closely related via e.g. geometric duality). In methods based on the pivot operation of the simplex method, the graph of feasible bases is explored; this is efficient precisely when the system has not many more feasible bases than output elements (i.e. is *non-degenerate*). In incremental methods one inductively finds the second representation for the polyhedron defined by $n-1$ of $n$ input elements and then updates this description for the last piece of input. These methods work well in many degenerate cases, but sometimes construct an exponentially large intermediate structure which is discarded in the final output. A third class of methods recursively constructs the entire face lattice (or a substantial part of it). These methods are also largely immune to degeneracy, but typically construct a large number of faces of intermediate dimensions not of interest in the application. For this reason, recursive methods have not been as widely applied as incremental and pivoting based methods.

In contrast to the symmetry-free case, methods based on recursive decomposition into orbits of subproblems have been the most successful in applications related to combinatorial optimization and the geometry of numbers. For the sake of exposition, suppose we are converting from the extreme ray representation of a convex cone to its inequality representation. Orbits of subproblems can be constructed either by dividing the input rays into orbits, and finding all facet orbits adjacent to a given representative ray (so-called *incidence decomposition*), or by finding the facets adjacent to a given representative facet (with the initial facet found e.g. by Linear Programming); in this *adjacency decomposition* method one then carries out a graph search in the dual graph (facet/ridge graph) of the polytope. In both cases one is left with one or more representation conversion problems in one lower dimension. Depending on various heuristics, one can either solve the subproblems directly (perhaps using one of the methods below), or further apply decomposition. Existing software using decomposition methods includes the `Polyhedral` package of Dutour Sikirić [5] and the `Sympol` package of Rehn [10].

Of the direct methods of orbitwise representation conversion, the incremental *Cascade Method* of Jaquet [7] was as far as I know the first to be elaborated. In this method, most easily understood in terms of projection, the $n$ input vectors in $\mathbb{R}^d$ are lifted to a simplicial cone in $\mathbb{R}^n$. Representatives of orbits of facets and ridges are then successively projected down, with stabilizers being recomputed, and orbits split or fused at each step. In addition to the well known dependence on input ordering (i.e. intermediate size) common to other incremental methods, the success of the Cascade Method also depends on the computation of many stabilizer groups, which also depends on the ordering.

In [4] the authors describe a method of pivoting under symmetry. Because this method finds orbits of the complete basis graph, it is particularly sensitive to finding a good perturbation scheme that does not destroy too much symmetry. Although it remains open to what extent this is possible in general, there seem to be a few specialized applications where pivoting is a good approach, particularly in certain algebraic methods where one apparently wants all of the orbits of bases. A prototype system in GAP [3] allowing exploration of pivoting and perturbation under symmetry is available.

In addition to systems based on recursive decomposition, there is also the possibility of using the *extend and canonicalize* technique used to good effect in the combinatorial enumeration community (see e.g. [6], [9], or [8]). In an *extend* step, one extends representatives of $k$-face orbits to $(k + 1)$-faces. A *canonical representative* is then chosen for each orbit of $(k + 1)$-faces. Given an efficient *parent* function which determines for a given $(k + 1)$-face, what canonical $k$-face it is an extension of, memoryless enumeration can be performed using the *reverse search* technique of Avis and Fukuda [2].

One common theme in efficient implementations of all of methods described here is the use of invariants to avoid expensive isomorphism tests. Another approach to avoiding isometry tests, and also permitting a more straightforward use of perturbation, is to construct a linear approximation of a fundamental domain for the given symmetry group. In this talk I also describe some ongoing work with Gegenberg, Schürmann and Williams using this fundamental domain approach in incremental and pivoting based approaches.

## References

[1] David Avis, David Bremner, and Raimund Seidel. How good are convex hull algorithms? *Comput. Geom.*, 7(5-6):265–301, 1997. 11th ACM Symposium on Computational Geometry (Vancouver, BC, 1995).

[2] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Appl. Math.*, 65(1-3):21–46, 1996. First International Colloquium on Graphs and Optimization (GOI), 1992 (Grimentz).

[3] David Bremner, *symbal home page*, `http://www.cs.unb.ca/~bremner/software/symbal`

[4] David Bremner, Mathieu Dutour Sikirić, and Achill Schürmann. Polyhedral representation conversion up to symmetries. *Polyhedral Computation*, CRM Proceedings and Lecture Notes, 48:45–71, 2009.

[5] Mathieu Dutour Sikirić, *Polyhedral home page*, `http://www.liga.ens.fr/~dutour/Polyhedral`.

[6] I. A. Faradžev. Constructive enumeration of combinatorial objects. In *Problèmes combinatoires et théorie des graphes (Colloq. Internat. CNRS, Univ. Orsay, Orsay, 1976)*, volume 260 of *Colloq. Internat. CNRS*, pages 131–135. CNRS, Paris, 1978.

[7] D.O. Jaquet, *Énumération complète des formes parfaites en dimension* 7, Ann. Inst. Fourier **43-1** (1993), 21–55.

[8] Brendan D. McKay. Practical graph isomorphism. In *Proceedings of the Tenth Manitoba Conference on Numerical Mathematics and Computing, Vol. I (Winnipeg, Man., 1980)*, volume 30, pages 45–87, 1981.

[9] Ronald C. Read. Every one a winner or how to avoid isomorphism search when cataloguing combinatorial configurations. *Ann. Discrete Math.*, 2:107–120, 1978. Algorithmic aspects of combinatorics (Conf., Vancouver Island, B.C., 1976).

[10] Thomas Rehn, *SymPol home page*, `http://fma2.math.uni-magdeburg.de/~latgeo/sympol/sympol.html`

## Generating sets and generating functions under symmetry

Matthias Köppe

**Generating sets.** Let $C \subset \mathbb{R}^n$ be a pointed rational polyhedral cone. The *Hilbert basis* $H(C)$ of $C$ is the set of nonzero vectors $\mathbf{x} \in C \cap \mathbb{Z}^n$ such that $\mathbf{x} \neq \mathbf{y}+\mathbf{z}$ for any $\mathbf{y}, \mathbf{z} \in C \cap \mathbb{Z}^n$. It is a theorem that $H(C)$ is a finite set that generates $C \cap \mathbb{Z}^n$ over the non-negative integers. Hilbert bases play various important roles in Integer Optimization. For instance, via the notion of Graver bases, which are unions of Hilbert bases, efficient primal (augmentation) algorithms for linear and separable convex objective functions are obtained, whenever the bases are available.

In the language of commutative algebra, the computation of Hilbert bases provides the normalization of affine semigroups. In particular, if the Hilbert bases only consists of the primitive representatives of the rays of the cone, this proves the normality of the semigroup.

A problem in algebraic statistics concerns the normality of semigroups associated with contingency tables. Let $r_1, \dots, r_N$ be natural numbers. An $r_1 \times r_2 \times \cdots \times r_N$ *contingency table* is a function $T\colon \{1, \dots, r_1\} \times \cdots \times \{1, \dots, r_N\} \to \mathbb{Z}_+$. *Marginals* are formed by summing over entries; for example, by fixing all but one index and summing the entries when that one index varies, one obtains *line sums.* Let $A\mathbf{x} = \mathbf{b}$ denote the corresponding constraints for the entries of the table (collected in a vector $\mathbf{x} \in \mathbb{Z}_+^{r_1 \cdots r_N}$), where $\mathbf{b}$ is a vector of prescribed line sums. Let $S_{r_1,\dots,r_N}$ be the semigroup generated by the column vectors of $A$. The question is now which of these semigroups are normal.

The normality of monoids derived from $r_1 \times r_2 \times \cdots \times r_N$ contingency tables (with line sums) was settled almost completely in [4]. The remaining open cases ($5 \times 5 \times 3$, $5 \times 4 \times 3$, and $4 \times 4 \times 3$) could not be solved computationally by the existing state-of-the art software for Hilbert basis computations (`Normaliz`, `4ti2`).

With Raymond Hemmecke, we used a symmetric Hilbert basis computation to resolve these cases computationally, showing that all of them are normal. The construction recursively computes polyhedral subdivisions of the cone and uses the large symmetry group of the tables ($S_{r_1} \times S_{r_2} \times \cdots \times S_{r_N}$) to cover "small" cones by symmetric copies of "large" cones; then the small cones can be discarded. This is implemented in the software package `LattE-for-tea-too`. Independently, Bruns, Ichim, and Söger gave another computational proof, exploiting the fact that the cones are nearly compressed; hence many cones in any pulling triangulation are unimodular.

The joint paper [2] details both computational approaches and highlights other challenging examples from algebraic statistics that could be solved with them.

**Generating functions.** The Khovanski–Pukhlikov–Lawrence theorem asserts the existence of a linear map (valuation) $F$ from the vector space generated by indicator functions of rational polyhedra in $\mathbb{R}^n$ to the rational functions in $\mathbb{Q}(z_1, \ldots, z_n)$ that agrees for pointed polyhedra $P$ with the rational function defined by the series (generating function) $\sum_{\mathbf{a} \in P \cap \mathbb{Z}^n} \mathbf{z}^{\mathbf{a}}$, for all $\mathbf{z}$ for which this series converges absolutely.

By Barvinok's theorem [1], for every fixed dimension $n$, it can be efficiently computed in the form

$$F([P])(\mathbf{z}) = \sum_{i \in I} \epsilon_i \frac{\mathbf{z}^{\mathbf{a}_i}}{\prod_{j=1}^{n} (1 - \mathbf{z}^{\mathbf{b}_{i,j}})},$$

where all data are integers and $I$ is a polynomial-sized index set, a so-called "short" rational generating function. Variants of Barvinok's algorithm are implemented in the software packages `LattE`, `LattE macchiato`, and `barvinok`.

In optimization, short rational generating functions appear as follows. By evaluating the generating function at $\mathbf{z} = (1, \ldots, 1)$ (nontrivial, because $\mathbf{1}$ is a pole of all basic rational terms in the sum), one obtains the number of integer points in a polytope. Thus the integer feasibility problem and, by binary search, the integer optimization problem can be solved. This gives a proof of the polynomial-time solvability of integer linear programs in fixed dimension, which is independent of Lenstra's branching on hyperplanes algorithm.

For the the problem of maximizing a polynomial objective function (without convexity) over the integer points in a polytope, one can use the relation

$$\max\{f(\mathbf{x}) : \mathbf{x} \in P \cap \mathbb{Z}^n\} = \lim_{k \to \infty} \left( f^k\big(z_1 \tfrac{\partial}{\partial z_1}, \ldots, z_n \tfrac{\partial}{\partial z_n}\big) F([P])(\mathbf{z}) \bigg|_{\mathbf{z}=\mathbf{1}} \right)^{1/k}$$

to obtain a fully polynomial time approximation scheme (FPTAS) [3].

If the polyhedron $P$ is symmetric, polyhedral computations using symmetry (orbitwise computation of vertices and their tangent cones; computation of triangulations that preserve a large part of the existing symmetry) can be used to obtain an orbitwise rational generating function.

An open question is how to then solve the evaluation problem in an orbitwise way, to extract information from the rational generating function. The known evaluation procedures choose a generic vector $\mathbf{t} \in \mathbb{Q}^d$, i.e., a vector not orthogonal to any of the vectors $\mathbf{b}_{ij}$, which is used to define a curve $\mathbf{z}(\tau)$ (with $\mathbf{z}(0) = \mathbf{1}$) which meets the poles of the rational summands only transversally. Then residue calculus in one variable can be used to compute the limit for $\tau \to 0$. The necessary generic choice conflicts with the goal of exploiting symmetry. A solution for a first interesting special case, where orbits are generated by a cyclic group, is presented.

## References

[1] A. I. Barvinok, *Polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed*, Mathematics of Operations Research **19** (1994), 769–779.

[2] W. Bruns, R. Hemmecke, B. Ichim, M. Köppe, and C. Söger, *Challenging computations of Hilbert bases of cones associated with algebraic statistics*, e-print arXiv:1001.4145v1 [math.CO], to appear in *Experimental Mathematics*, 2010.

[3] J. A. De Loera, R. Hemmecke, M. Köppe, and R. Weismantel, *Integer polynomial optimization in fixed dimension*, Mathematics of Operations Research **31** (2006), no. 1, 147–153.

[4] H. Ohsugi and T. Hibi, *Toric ideals arising from contingency tables*, Commutative Algebra and Combinatorics, Ramanujan Mathematical Society Lecture Note Series, vol. 4, 2006, pp. 87–111.

## Polyhedral symmetries

Achill Schürmann

Polyhedra with symmetries occur naturally in many contexts of pure and applied mathematics, and in particular in symmetric (mixed) linear (integer) programming problems. Often not all of the symmetries of a problem are known and one may ask what the possible symmetries are and how they could be found? What do known symmetries tell us about a given polyhedron? Here, we address these and related basic questions about symmetric polyhedra, some of which appear to be widely open.

**Symmetry groups.** First of all, let us ask what kind of symmetries are there? And how can they be detected? In practice, the answer to this question largely depends on the given description of a polyhedron. For simplicity of this exposition we assume that we have a description as the convex hull

$$P = \operatorname{conv}\{x_1, \dots, x_k\}$$

of finitely many points $x_i \in \mathbb{R}^n$. So we in particular assume that the polyhedron is bounded.

*Combinatorial.* The combinatorial (or full) symmetry group of a polyhedron is the group of all permutations of the vertices that preserve the combinatorial structure of the polyhedron, that is, its face-lattice. So it naturally embeds as a permutation group into the full symmetric group $S_k$ on $k$-element sets. For example, the combinatorial symmetry group of *any* 4-gon is the dihedral group $D_4$ of order 8. To compute the combinatorial symmetry group of a polyhedron $P$, one needs not only to know the vertices of $P$, but also a dual description by inequalities (cf. [KS03]), which is usually infeasible in practice.

*Geometric.* Geometric symmetries are distance and angle preserving linear maps. For example, the only 4-gon having a geometric symmetry group of order 8 is the square. If we assume (without loss of generality) that the *vertex barycenter* $\frac{1}{k}\sum_{i=1}^{k} x_i$ of $P$ is equal to the origin, then the geometric symmetries of $P$ form a subgroup $O_P$ of the orthogonal group. Typical geometric symmetries occurring in optimization problems are coordinate permutations.

*Linear.* Often polyhedra may have a slightly larger linear symmetry group $G_P$. For example, a non-square parallelogram has a geometric symmetry group of order 4, but a linear symmetry group of order 8, namely $D_4$. Although the linear group is usually smaller than the full combinatorial symmetry group, it has the advantage that it can be computed without any additional knowledge about the face lattice. This is due to the following theorem, whose proof can be found in [Sch09, Appendix A]:

**THM:** Let $P = \operatorname{conv}\{x_1, \ldots, x_k\}$ be a full dimensional polyhedron in $\mathbb{R}^n$. Then its linear automorphism group $G_P$ is equal to the automorphism group of the complete graph $K_k$ with edge labels $x_i^t Q^{-1} x_j$, where $Q = \sum_{i=1}^{k} x_i x_i^t$.

The requirement that $P$ is full dimensional is not essential, but simplifies the exposition, as we may "work" with $Q^{-1}$. As this matrix is positive definite, it has a Cholesky decomposition $Q^{-1} = A^t A$ with a regular matrix $A$ that allows us to realize the linear symmetry group of $P$ as the geometric symmetry group of a transformed polyhedron $AP = \{Ax : x \in P\}$. That is, $G_P = O_{AP}$.

For the computation of graph isomorphisms several software packages are available. A simple alternative is our software package [`SymPol`] that takes care of the technicalities (also for lower dimensional input). It uses the above theorem to compute the automorphism group of a polyhedron that is given by a standard description with inequalities or generators (vertices and rays). The package also allows to convert a given polyhedral description up to symmetry, using the *Incidence and Adjacency Decomposition Methods* (see [BDS09]).

The detection method above is problematic for large $k$. To the best of our knowledge, other practical detection methods are not known. A new method that works well for larger $k$, when $n$ is comparatively small, will be described in [Reh10]. It is based on computing automorphisms of the Gram matrix $Q$ (see [PS97]).

**Symmetry types.** Given a polyhedron $P$ with $k$ vertices, what are the possible geometric and linear symmetry groups $O_P$ and $G_P$? For a 4-gon for example, the geometric symmetry group can not be isomorphic to the cyclic group $C_4$ of order 4, as the presence of such rotational symmetries would imply an additional reflection symmetry, forcing $O_P$ to be isomorphic to $D_4$. This on the other hand would imply that $P$ is a square. It is easily checked that the linear symmetry group $G_P$ of a 4-gon $P$ can not even be of order 4. Such implications provide additional insight that could potentially be used in polyhedral algorithms and which could be valuable not only for the detection of symmetries. To the best of our knowledge not much is known in this direction.

The only systematic study of such implications for geometric symmetry groups, with results mainly in 2 and 3-dimensions, can be found in [Rob84]. For this, the "space of polytopes" is considered as a manifold which decomposes into strata of *symmetry types.* The points of the manifold represent similarity classes of polytopes, and are grouped into "cells" with the same geometric symmetry group, respectively the same symmetry type. For example, parallelograms form a two-dimensional family of 4-gons (up to similarity). On the boundary of the associated 2-cell, one finds four "extreme cases", two 1-cells (rectangles and rhombuses) and two 0-dimensional cells (*perfect symmetry types*), one consisting of a square and one consisting of a segment (the similarity class of one-dimensional polyhedra).

Many basic questions about possible symmetry types appear to be open: The symmetry types in dimension $n \geq 3$ have not yet been classified. The perfect symmetry types for $n \geq 4$ are unknown, and for odd dimensions $n \geq 5$ it is not even known if there exist infinitely many of them. To the best of our knowledge, a corresponding study of *linear symmetry types* (according to the groups $G_P$) has not at all been looked at.

**Decompositions.** Not only in (mixed) integer linear programming problems, the considered symmetry groups $G$ are usually – if not always – linear (respectively affine). This implies that there exists a linear (or affine) *invariant subspace* $\mathrm{fix}_G(\mathbb{R}^n)$. In linear programming one may simply restrict the optimization over a polyhedron $P$ (of feasible solutions) to the $G$-fixed part in

$$\mathrm{fix}_G(P) \;=\; P \cap \mathrm{fix}_G(\mathbb{R}^n)\,,$$

as there is always an optimal solution in the invariant subspace (see also the exposition of Katrin Herr). In integer programming this is usually not possible, as it can not be guaranteed that an *integral* solution is contained in the invariant subspace. Nevertheless, we think that "working" with or within $\mathrm{fix}_G(P)$ could be beneficial (generalizing the approach of [BH09]). For this one needs to consider the polyhedron $P$ as a collection of *$G$-symmetric fibers* that project (orthogonally with respect to a suitable inner product) onto the fixed points $\mathrm{fix}_G(P)$. In particular the vertices of $P$ themselves lie in such fibers and they allow a nice description of $\mathrm{fix}_G(P)$ in terms of barycenters (a proof is given in [SS10]):

**THM:** Let $P = \mathrm{conv}\{x_1, \ldots, x_k\}$ be a polyhedron in $\mathbb{R}^n$ and suppose $G$ is a subgroup of the linear symmetry group $G_P$. Suppose the set of $x_i$ is split into $l$

disjoint $G$-orbits $O_1, \dots, O_l$. Then

$$\operatorname{fix}_G P = \operatorname{conv}\{b_1, \dots, b_l\},$$

where $b_i = \frac{1}{|O_i|} \sum_{x \in O_i} x$ denotes the barycenter of orbit $O_i$.

## References

[BDS09] D. Bremner, M. Dutour Sikirić, and A. Schürmann, *Polyhedral representation conversion up to symmetries*, Proceedings of the 2006 CRM workshop on polyhedral computation (D. Bremner D. Avis and A. Deza, eds.), CRM Proceedings & Lecture Notes, vol. 48, AMS, Providence, 2009, pp. 45–71.

[BH09] R. Bödi and K. Herr *Symmetries in Linear and Integer Programs*, preprint at `arXiv:0908.3329`.

[KS03] V. Kaibel and A. Schwartz, *On the complexity of polytope isomorphism problems*, Graphs Combin. **19** (2003), 215–230.

[PS97] W. Plesken and B. Souvignier, *Computing isometries of lattices*, J. Symbolic Comput. **24** (1997), 327–334.

[Reh10] T. Rehn, *Polyhedral Description Conversion up to Symmetries*, Diploma thesis (Mathematics), Otto von Guericke University Magdeburg, 2010, in preparation.

[Rob84] S.A. Robertson, *Polytopes and symmetry*, Cambridge University Press, Cambridge, 1984.

[Sch09] A. Schürmann, *Computational geometry of positive definite quadratic forms*, University Lecture Series, vol. 48, American Mathematical Society, Providence, RI, 2009.

[SS10] R. Scharlau and A. Schürmann, *The barycenter polytope*, preprint in preparation, 2010.

## Software

[`SymPol`] *a C++ tool for the work with symmetric polyhedra*, by T. Rehn and A. Schürmann, preliminary version 0.1.1, `http://fma2.math.uni-magdeburg.de/~latgeo/sympol/sympol.html`.

## Symmetry Breaking Constraints in Constraint Programming

Barbara Smith

A common way to exploit symmetry in constraint satisfaction problems is to transform the symmetric CSP instance by adding constraints in such a way the new CSP has at least one solution from each symmetry equivalence class of solutions in the original CSP, and ideally only one. Crawford, Ginsberg, Luks and Roy, in a 1996 paper [1], gave a standard procedure for deriving so-called lex-leader constraints in SAT problems that has subsequently been adapted for the general CSP, principally for variable symmetries. The lex-leader constraint for a given element of the symmetry group excludes any solution that is lexicographically larger than its symmetric equivalent, given an ordering of the variables of the CSP instance.

Ensuring that there is only one solution in the transformed CSP instance for every symmetry equivalence class requires in principle a lex-leader constraint for every element of the symmetry group. Where it is impracticable to generate so many constraints, we can resort to partial symmetry breaking, and generate constraints for only a subset of the symmetry group. I discuss which symmetries might be a good choice in that case.

In constructing lex-leader constraints, we also have to choose a variable ordering and I discuss the effect of changing the ordering and why it might be efficient to use the same variable ordering for search.

## References

[1] J. M. Crawford, M. L. Ginsberg, E. Luks, A. Roy, *Symmetry breaking predicates for search problems*, in: Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR 96), 1996, 148–159.

## Algebraic Symmetry in Semidefinite Programs

Renata Sotirov
(joint work with Etienne de Klerk)

Semidefinite programming (SDP) is a generalization of linear programming where the nonnegativity constraints are replaced by positive semidefiniteness on the matrix variables. SDP has recently become a very powerful tool for providing tight relaxations for hard combinatorial optimization problems. Derived SDP relaxations are often large scale and therefore hard to solve with the currently available algorithms. In order to avoid computational difficulties, there are several techniques for reducing complexity of semidefinite programs.

In this talk, we show how to exploit algebraic symmetry of the data matrices, when present, in order to greatly reduce the size of the SDP relaxations. Here, we assume that data matrices of an SDP relaxation belong to a matrix *-algebra of low dimension. (Recall that a matrix *-algebra over $\mathbb{C}$ is a subspace of $\mathbb{C}^{n\times n}$, where $n$ is the order of the data matrices, that is closed under matrix multiplication and taking conjugate.) Under this algebraic symmetry assumption, we may restrict the optimization to the corresponding matrix *-algebra. Every matrix *-algebra has a basis that is known as a coherent configuration and a canonical block-diagonal structure after a suitable unitary transformation. This can be exploited in order to reduce the size of the corresponding SDP problem. More details may be found e.g., in the survey by Parrilo and Gatermann [1].

The described approach has several areas of applications: error correcting binary codes, kissing numbers, truss topology design, the traveling salesman problem, the quadratic assignment problem, etc. For an overview on problems in which algebraic symmetry has been successfully exploited see [4]. To illustrate the approach, we consider here the quadratic assignment problem and report the best known bounds for large instances whose one of the data matrix is a Hamming distance matrix (see [2, 3]).

## References

[1] K. Gatermann and P.A. Parrilo, *Symmetry groups, semidefinite programs, and sum of squares*, Journal of Pure and Applied Algebra, **192** (2004), 95–128.

[2] E. de Klerk and R. Sotirov, *Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem*, Math. Program. A, **122/2** (2010), 225–246.

[3] E. de Klerk and R. Sotirov. *Improved semidefinite programming bounds for quadratic assignment problems with suitable symmetry*, Math. Program. A (accepted).

[4] E. de Klerk and R. Sotirov. *A new library of structured SDP instances*, Optimization Methods and Software, **24/6** (2009), 959–971.

## Infinite-dimensional semidefinite programs

Frank Vallentin

(joint work with Christine Bachoc, Hans D. Mittelmann)

Starting point of my talk are approximation algorithms for NP-hard problems in combinatorial optimization which are based on semidefinite programming (SDP), a recent and powerful method in convex optimization. One example is the theta number of Lovász [5] which provides an upper bound for the largest size of an independent set of finite graphs based on a solution of a semidefinite program.

Many problems in extremal discrete geometry can be formulated as maximum independent set problems, but then for infinite geometric graph, see for instance [8] for packing and coloring problems.

A famous example is the kissing number problem which goes back to a discussion betweem Newton and Gregory in 1692: What is the maximum number of non-overlapping unit balls that can simultaneously touch a central unit ball?

Here the vertex set of the underlying infinite graph $\Gamma(S^{n-1}, (0, \pi/3))$ is the unit sphere $S^{n-1} = \{x \in \mathbb{R}^n : x \cdot x = 1\}$ and we have edges between two different vertices if the angular distance between them is strictly less than $\pi/3$, or if the inner product between them is strictly bigger than $1/2$. Independent sets of this graph correspond to possible touching points in a kissing configuration.

To tackle this problem we generalize the theta number (and strengthenings based on SDP hierarchies of Lovász, Schrijver [6], and of Lasserre [4]) to infinite graphs which yields an infinite-dimensional semidefinite program. For instance, the following infinite-dimensional semidefinite program gives an upper bound to the kissing number in dimension $n$:

$$\vartheta'(\Gamma(S^{n-1}, (0, \pi/3))) = \inf \Big\{ \lambda : \begin{array}{l} K \in \mathcal{C}(S^{n-1} \times S^{n-1})_{\succeq 0}, \\ K(x, x) = \lambda - 1, \text{ for all } x \in S^{n-1}, \\ K(x, y) \leq -1, \text{ for all } x, y \in S^{n-1} \\ \quad \text{with } x \cdot y \leq 1/2 \Big\}, \end{array}$$

where $\mathcal{C}(S^{n-1} \times S^{n-1})_{\succeq 0}$ denotes the cone of positive definite Hilbert-Schmidt kernels. (In fact it turns out that this semidefinite program is equivalent to Delsarte's linear programming bound, see [2]).

By using symmetries and tools from harmonic analysis one can solve these semidefinite programs by computer giving the best known upper bounds in dimensions up to 24.

For further details see [3] where the method is explained and [7] where computational results are given.

## References

[1] C. Bachoc, D.C. Gijswijt, A. Schrijver, F. Vallentin, *Invariant semidefinite programs*, preprint, 49 pages, 2010.

[2] C. Bachoc, G. Nebe, F.M. de Oliveira Filho, F. Vallentin, *Lower bounds for measurable chromatic numbers*, Geom. Funct. Anal. **19** (2009), 645–661.

[3] C. Bachoc, F. Vallentin, *New upper bounds for kissing numbers from semidefinite programming*, J. Amer. Math. Soc. **21** (2008), 909–924.

[4] J.B. Lasserre, *An explicit equivalent positive semidefinite program for nonlinear* $0-1$ *programs*, SIAM J. Optim. **12** (2002), 756–769.

[5] L. Lovász, *On the Shannon capacity of a graph*, IEEE Trans. Inf. Th. **25** (1979), 1–7.

[6] L. Lovász, A. Schrijver, *Cones of matrices and set-functions and* $0-1$ *optimization*, SIAM J. Optim. **1** (1991), 166–190.

[7] H.D. Mittelmann, F. Vallentin, *High accuracy semidefinite programming bounds for kissing numbers*, Experiment. Math. **19** (2010), 174-178.

[8] F.M. de Oliveira Filho, *New Bounds for Geometric Packing and Coloring via Harmonic Analysis and Optimization*, PhD thesis, University of Amsterdam, 2009.

# Symmetry in Mathematical Programming

Leo Liberti

## 1. Encoding of a mathematical program

Mathematical Programming (MP) is a formal language for describing optimization problems of the form:

$$\min\{f(x) \mid g(x) \le 0 \wedge x \in X\}, \tag{1}$$

where $x \in \mathbb{R}^n$ is a vector of decision variables, $X \subseteq \mathbb{R}^n$ might include bounds and integrality constraints on subsequences of $x$, and $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are functions that can be written as strings of a formal language $\mathscr{E}$ on the alphabet $\mathscr{A} = \mathscr{O} \cup \mathbb{Q} \cup \mathscr{V}$, where $\mathscr{V} = \{\mathsf{x}_i \mid i \in \mathbb{N}\}$ and $\mathscr{O} = \{+, -, \times, \div, (\cdot)^{(\cdot)}, \log, \exp, (, )\}$. The strings of $\mathscr{E}$ are only and all those that can be obtained as follows: (a) $\forall \mathsf{s} \in \mathbb{Q} \cup \mathscr{V}$ $(\mathsf{s} \in \mathscr{E})$; (b) $\forall \otimes \in \mathscr{O}$ representing a $k$-ary operator and $\mathsf{e}_1, \ldots, \mathsf{e}_k \in \mathscr{E}$, $\otimes(\mathsf{e}_j \mid j \le k)$ is in $\mathscr{E}$ [2]. If $P \in$ MP, let $\mathcal{F}(P) \subseteq X$ be the set of its feasible solutions, i.e. those $x \in X$ satisfying $g(x) \le 0$, and $\mathcal{G}(P) \subseteq \mathcal{F}(P)$ the set of its globally optimal solutions.

The recognition process (or parsing) of a valid string $h \in \mathscr{E}$ naturally yields a directed graph $D(h)$ whose leaf nodes are elements of $\mathbb{Q} \cup \mathscr{V}$ and whose non-leaf nodes are elements of $\mathscr{O}$. Every $P \in$ MP has a Directed Acyclic Graph (DAG) representation $D(P)$ obtained as a minor of $D(f) \cup \bigcup_{i \le m} D(g_i)$ by contracting all equal leaf nodes [2].

## 2. Solution and formulation groups

For a group $G$ acting on $X$, we let $Gx = \{gx \mid g \in G\}$ be the orbit of $x$ in $G$ for all $x \in X$; we let $\mathsf{stab}(Y, G) = \langle g \in G \mid \forall y \in Y\ (gy \in Y)\rangle$ be the setwise stabilizer and $G^Y = \langle g \in G \mid \forall y \in Y\ (gy = y)\rangle$ be the pointwise stabilizer of $Y$ w.r.t. $G$ for all $Y \subseteq X$. Let $[n] = \{1, \ldots, n\}$. For a permutation $\pi \in S_n$ let $\Gamma(\pi)$ be the set of all the cycles in its (unique) disjoint cycle representation. For $N \subseteq [n]$ we define $\pi[N] = \prod_{\sigma\in\Gamma(\pi)\cap\mathsf{stab}(N,S_n)} \sigma$ to be the restriction of $\pi$ to $N$. If $g_1, \ldots, g_k \in G \leq S_n$ are generators for $G$ (i.e. $G = \langle g_j \mid j \leq k\rangle$) we define $G[N] = \langle g_j[N] \mid j \leq k\rangle$ to be the restriction of $G$ to $N$. We consider the action of $G \leq S_n$ on $X$ given by $\pi x = (x_{\pi(i)} \mid i \leq n)$. This action induces a right action $P \mapsto P\pi$ (where $\pi \in S_n$) on MP by replacing $x$ with $\pi x$ everywhere in (1). $S_m$ also induces a left action $P \mapsto \sigma P$ (where $\sigma \in S_m$) given by replacing $g = (g_1, \ldots, g_m)$ by $\sigma g$. Because optimization problems (1) are independent of the order of the constraints, $\sigma P = P$ for all $\sigma \in S_m$, which implies $\forall \sigma \in S_m, \pi \in S_n\ (\sigma P)\pi = \sigma(P\pi)$.

We define the *solution group* $G^*(P) = \mathsf{stab}(\mathcal{G}(P), S_n)$ and the *formulation group* $G_P = \langle \pi \in S_n \mid \exists \sigma \in S_m\ (\sigma P\pi = P)\rangle$ of a MP formulation $P$ given by (1); it is easy to show that $G_P \leq G^*(P)$. Computing the solution group in general requires aprioristic knowledge of $\mathcal{G}(P)$, which is usually the ultimate aim when considering and solving MPs, and is therefore impractical. Since deciding whether two function encodings $h_1, h_2 \in \mathscr{E}$ are equal has linear complexity in $|D(P)|$, computing generators for $G_P$ is a decidable problem [6] which can be solved once the formulation of $P$ is known. By choosing an appropriate colouring $\gamma : D(P) \to \mathbb{N}$ of the vertices of $D(P)$ (in order to avoid permutations of nodes of different types, e.g., operator nodes with variable nodes), we show that $G_P = \mathsf{Aut}(D(P), \gamma)[N]$, where $\mathsf{Aut}(\mathcal{G}, \delta)$ is the group of automorphisms of the graph $\mathcal{G}$ which stabilizes each equivalence class given by the vertex colouring $\delta$ setwise [6].

## 3. Impact of symmetry on solution algorithms

Although there appears to be no clearly defined relation between how large $G_P$ is and how hard it is to solve $P$ in practice, there are two common sense arguments motivating the study of symmetries in MP, having to do with two different algorithmic classes. When $P$ is solved exactly (or approximately) using Branch-and-Bound (BB) type algorithms, such as [3] when $P$ is a Mixed-Integer Linear Program or [1] when $P$ is a Mixed-Integer Nonlinear Program, and only one global optimum is required, then multiple symmetric global optima generally yield larger BB trees, and therefore longer solution processes. When heuristic or meta-heuristic algorithms (e.g. [7]) are used in order to find good solutions of $P$, the picture is sometimes reversed: if the algorithm stochastically explores the neighbourhood of the most recent (or best) found local optimum, having several optima generally prevents the algorithm from getting stuck early on in the search [5].

Our research, being motivated by BB type algorithms, aims to determine $G_P$ in view to somehow exclude symmetric global optima in order to reduce the size of the search tree. The techniques used to break symmetries can be split into two categories: *static* and *dynamic* symmetry breaking [9]. Static symmetry breaking consists in reformulating $P$ in such a way that at least one global optimum is preserved, but (hopefully) many are excluded: this is known as a *narrowing* reformulation [4]. The reformulated problem is then solved by means of an appropriate solution algorithm. Dynamic symmetry breaking techniques are embedded in the BB algorithm and attempt to determine whether the current node is the symmetric map of another, previously treated, node [8]. The rest of this abstract focuses on static symmetry breaking.

The occurrence of symmetry in MP is far from rare. The computational experiments in [6] show that 18% of the instances in three well-known public libraries (MIPLib, GlobalLib, MINLPLib) have a nontrivial formulation group.

## 4. Symmetry breaking constraints

In [6], new methods were presented in order to break symmetries with two types of general-purpose Symmetry Breaking Constraints (SBC) derived from the set $\Omega$ of orbits of the action of $G_P$ on the variable index set $[n]$. Specifically, for a nontrivial orbit $\omega \in \Omega$, if the transitive constituent $G_P[\omega]$ can be ascertained to be isomorphic to the full symmetric group $\mathsf{Sym}(\omega)$ on $\omega$, then a unique order can be imposed on the variables indexed by $\omega$ by adjoining the following linear inequalities to $P$:

$$\forall j \in \omega \smallsetminus \{\max \omega\} \; x_j \leq x_{j^+}, \tag{2}$$

where $j^+$ is the successor of $j$ in $\omega$. Otherwise, for any structure $G_P[\omega]$ might have, one can always choose a variable (for example $x_{\min \omega}$) that should have minimum values among all those indexed by $\omega$:

$$\forall j \in \omega \smallsetminus \{\min \omega\} \; x_{\min \omega} \leq x_j. \tag{3}$$

In general, if $\omega, \theta \in \Omega$ with $\omega \neq \theta$ and $g^\omega(x) \leq 0$ and $g^\theta(x) \leq 0$ are SBCs w.r.t., resp., $\omega$ and $\theta$, adjoining both $g^\omega(x) \leq 0$ and $g^\theta(x) \leq 0$ to $P$ does not yield a valid narrowing of $P$. We showed in [6] some sufficient conditions by which SBCs originating from different orbits could be combined into a valid narrowing of $P$. Specifically, this holds if: (a) $G_P[\omega \cup \theta]$ contains a subgroup $H$ such that $H[\omega] \cong C_{|\omega|}$ and $H[\theta] \cong C_{|\theta|}$ (where $C_p$ is the cyclic group of order $p$ for all $p \in \mathbb{N}$) and (b) $\mathsf{gcd}(|\omega|, |\theta|) = 1$. We remark that these conditions are quite restrictive but not believed necessary.

An idea to improve this state of affairs was suggested to me during the workshop by J. Ostrowski. Consider the SBCs generated by the following procedure.

(1) Let $G = G_P$
(2) Let $\Omega$ be the set of orbits of the action of $G$ on $[n]$
(3) Choose an orbit $\omega \in \Omega$
(4) Adjoin some SBCs $g(x_j \mid j \in \omega) \leq 0$ to $P$
(5) If $G^\omega$ is nontrivial, set $G = G^\omega$ and go to Step 2.

The above algorithm generates a sequence $\omega_1, \ldots, \omega_k$ of disjoint subsets of $[n]$ (not necessarily a partition thereof) with corresponding SBCs $g^1(x) \leq 0$, $\ldots$,

$g^k(x) \leq 0$ which can all be simultaneously adjoined to $P$, for each $\omega_h$ is an orbit of the pointwise stabilizer of $\omega_{h-1}$.

## References

[1] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24(4):597–634, 2009.

[2] A. Costa, P. Hansen, and L. Liberti. Formulation symmetries in circle packing. In R. Mahjoub, editor, *Proceedings of the International Symposium on Combinatorial Optimization*, volume 36 of *Electronic Notes in Discrete Mathematics*, pages 1303–1310, Amsterdam, 2010. Elsevier.

[3] ILOG. *ILOG CPLEX 11.0 User's Manual.* ILOG S.A., Gentilly, France, 2008.

[4] L. Liberti. Reformulations in mathematical programming: Definitions and systematics. *RAIRO-RO*, 43(1):55–86, 2009.

[5] L. Liberti. Symmetry in mathematical programming. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume IMA. Springer, New York, accepted.

[6] L. Liberti. Reformulations in mathematical programming: Automatic symmetry detection and exploitation. *Mathematical Programming*, DOI 10.1007/s10107-010-0351-0.

[7] L. Liberti, N. Mladenović, and G. Nannicini. A good recipe for solving MINLPs. In V. Maniezzo, T. Stützle, and S. Voß, editors, *Hybridizing metaheuristics and mathematical programming*, volume 10 of *Annals of Information Systems*, pages 231–244, New York, 2009. Springer.

[8] F. Margot. Pruning by isomorphism in branch-and-cut. *Mathematical Programming*, 94:71–90, 2002.

[9] F. Margot. Symmetry in integer linear programming. In M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, and L. Wolsey, editors, *50 Years of Integer Programming*, pages 647–681. Springer, Berlin, 2010.

## Detecting and breaking symmetries in circle packing

Alberto Costa

### 1. Introduction

The performance of Branch-and-Bound algorithms is severely impaired by the presence of symmetric optima in a given problem; in fact, this situation causes longer branches, and a higher number of nodes to explore. In order to break these symmetries, some *Symmetry Breaking Constraints* (SBCs) can be adjoined to the original formulation [1, 2], obtainig a *narrowing* reformulation [3]. After finding these symmetries for the *circle packing in a square* problem [4], three different classes of SBCs are proposed, and some computational results obtained with these SBCs are provided.

### 2. Static Symmetry Breaking Constraints for circle packing in a square problem

Consider the following problem.

Circle Packing in a Square (CPS). Given $N \in \mathbb{N}$ and $S \in \mathbb{Q}_+$, can $N$ non-overlapping circles of unit radius be arranged in a square of side $2S$?

| Type of SBCS | Set of inequalities |
|---|---|
| *weak* | $x_1 \leq x_2$, $x_1 \leq x_3, \dots, x_1 \leq x_9$ $x_1 \leq y_1$, $x_1 \leq y_2 \dots, x_1 \leq y_9$ |
| *strong* | $x_1 \leq x_2$, $x_2 \leq x_3$, $x_3 \leq x_4$, $x_4 \leq x_5$, $x_5 \leq x_6$, $x_6 \leq x_7$, $x_7 \leq x_8$, $x_8 \leq x_9$ |
| *mixed* | $x_1 \leq x_2$, $x_2 \leq x_3$, $y_1 \leq y_4$, $x_4 \leq x_5$, $x_5 \leq x_6$, $y_4 \leq y_7$, $x_7 \leq x_8$, $x_8 \leq x_9$ |

TABLE 1. SBCS for the instance with $N = 9$ and $S = 3$.

We formulate the CPS as the following nonconvex NLP:

$$(1)\quad \max\{\alpha \mid (x_i - x_j)^2 + (y_i - y_j)^2 \geq 4\alpha \, \forall i < j \leq N \wedge x_i,\, y_i \in [1-S, S-1] \, \forall i \in N\}$$

For any given $N, S > 1$, if a global optimum $(x^*, y^*, \alpha^*)$ of (1) has $\alpha^* \geq 1$ then the CPS instance is a YES one.

The following theorem is proved in [5]:

**Theorem 2.1.** *The formulation group of the CPS is isomorphic to $C_2 \times S_N$.*

In this case $C_2$ (the cyclic group of order 2) represents the permutation between $x$ and $y$ axes, while $S_N$ (the symmetric group of order $N$) represents the permutations of the circles.

In order to break these symmetries, three classes of constraints are proposed:

- *weak* constraints [5]: $x_1 \leq x_i$, $\forall i \leq N$, $x_1 \leq y_i$, $\forall i \leq N$;
- *strong* constraints [5]: $x_i \leq x_{i+1}$, $\forall i < N$;
- *mixed* constraints [6]: let $L = \lfloor S \rfloor$; starting from the *strong* constraints, replace $x_{iL} \leq x_{iL+1}$ with $y_{1+(i-1)L} \leq y_{1+iL}$, $\forall i \in \{1, 2, \dots, \lceil \frac{N}{L} \rceil - 1\}$.

2.1. **Example.** Consider the instance with $N = 9$ circles and $S = 3$ (hence, the side of the square is 6). Table 1 shows the different SBCs in this case.

## 3. Computational results

The experiments performed in [5] show that the results obtained with the *strong* and the *weak* SBCs are better than the ones obtained with the original formulation; however, the formers are more efficient than the latters, as suggested by their names. Nevertheless, the best results are obtained with the *mixed* constraints, as reported in [6]. Our comparative results, shown in table 2, have been obtained on a 2.4GHz Intel Xeon CPU with 24 GB RAM running Linux and the solver COUENNE [7] for some "limit" instances of CPS (i.e. $N$ circles fit in the square but $N + 1$ do not); the table displays the following statistics at termination (10h of CPU time): objective function value $f^*$ of the incumbent, number of BB nodes closed, number of BB nodes still on the tree. The best upper bound at termination was fixed at 2 (and hence the gap was always > 100%) for all reformulations and instances.

| | *strong* | | | *mixed* | | |
|---|---|---|---|---|---|---|
| *Instance* ($N\_S$) | $f^*$ | *nodes* | *tree* | $f^*$ | *nodes* | *tree* |
| 16_4 | 0.660 | 2381772 | 642285 | **1** | 2795501 | 839240 |
| 25_5 | 1 | 461224 | 188835 | 1 | 521487 | 222846 |
| 36_6 | 0 | 49962 | 23784 | **1** | 76409 | 34825 |
| 49_7 | 0 | 12577 | 6090 | **1** | 21366 | 10136 |
| 68_8 | 0 | 4 | 1 | **0.943** | 1057 | 497 |
| 86_9 | 0 | 4 | 1 | **0.640** | 5 | 1 |

Table 2. Results obtained with strong and mixed constraints.

## 4. Conclusion

As expected, adjoining SBCs improve the results. As a matter of fact, with these new formulations it is possible to solve more instances of CPS, although this approach does not scale so well to big ones. Moreover, unfortunately the upper bound on does not decrease by moving from the original formulation to one of the narrowings, but we notice a strange behaviour of the *mixed* SBCs: it seems that with this constraints the optimal solution is found earlier on the search. The future work has two main directions: first, finding other classes of SBCs for the CPS. Second, try to adjoin dynamically the constraints at each Branch-and-Bound node.

## References

[1] L. Liberti, *Automatic generation of symmetry-breaking constraints*, in: B. Yang, D.-Z. Du and C. Wang, editors, *COCOA Proceedings*, LNCS **5165**:328–338, 2008.

[2] L. Liberti, *Reformulations in mathematical programming: Automatic symmetry detection and exploitation*, Mathematical Programming, to appear.

[3] L. Liberti, *Reformulations in mathematical programming: Definitions and systematics*, RAIRO-RO **43**:55–86, 2009.

[4] M. Locatelli and U. Raber, *Packing equal circles in a square: a deterministic global optimization approach*, Discrete Applied Mathematics **122**:139–166, 2002.

[5] P. Hansen, A. Costa and L. Liberti, *Formulation symmetries in circle packing*, ISCO 2010 Proceedings, in M. Haouari, A.R. Mahjoub (Eds.), Electronic Notes in Discrete Mathematics **36**:1303–1310. Elsevier, 2010.

[6] A. Costa, P. Hansen and L. Liberti, *Static symmetry breaking in circle packing*, in U. Faigle, R. Schrader, D. Herrmann (Eds.), CTW 2010 Proceedings, 47–50, Kln, Germany, 2010.

[7] P. Belotti, J. Lee, L. Liberti, F. Margot and A. Wächter, *Branching and bounds tightening techniques for non-convex MINLP*, Optimization Methods and Software **24**:597–634, 2009.

*Reporter: Volker Kaibel*

## Participants

**Prof. Dr. David Bremner**
Department of Computer Science
University of New Brunswick
PO Box 4400
Fredericton , NB E3B 5A3
CANADA

**Prof. Dr. Alberto Costa**
Laboratoire d'Informatique (LIX)
Ecole Polytechnique
F-91128 Palaiseau Cedex

**Prof. Dr. Mathieu Dutour Sikiric**
Rudjer Boskovic Institute
Laboratory of Satellite Oceanography
Dept. of Environmental & Marine Research
Bijenicka 54
10000 Zagreb
CROATIA

**Katrin Herr**
Technische Universität Darmstadt
FB Mathematik, AG Optimierung
Algorithmic Discrete Mathematics
Dolivostr. 15
64293 Darmstadt

**Prof. Dr. Alexander Hulpke**
Department of Mathematics
Colorado State University
Weber Building
Fort Collins , CO 80523-1874
USA

**Prof. Dr. Volker Kaibel**
Fakultät für Mathematik
Otto-von-Guericke-Universität Magdeburg
Universitätsplatz 2
39106 Magdeburg

**Dr. Matthias Koeppe**
Department of Mathematics
University of California, Davis
1, Shields Avenue
Davis , CA 95616-8633
USA

**Prof. Dr. Leo Liberti**
Laboratoire d'Informatique (LIX)
Ecole Polytechnique
F-91128 Palaiseau Cedex

**Prof. Dr. Jeffrey T. Linderoth**
Dept. of Industrial Engineering
The University of Wisconsin-Madison
1513, University Avenue
Madison , WI 53706-1572
USA

**Prof. Dr. James Ostrowski**
Department of Management Sciences
University of Waterloo
200 University Avenue West
Waterloo , ON N2L 3G1
CANADA

**Prof. Dr. Dmitrii V. Pasechnik**
School of Physical & Mathematical Sc.
Nanyang Technological University
21 Nanyang Link
Singapore 637 371
SINGAPORE

**Prof. Dr. Marc Pfetsch**
Institut f. Mathematische Optimierung
T.U. Braunschweig
Pockelsstr. 14
38106 Braunschweig

**Thomas Rehn**
Immermannstr. 16
39108 Magdeburg

**Prof. Dr. Achill Schürmann**
Institut für Mathematik
Universität Rostock
Ulmenstr. 69
18057 Rostock

**Dr. Barbara M. Smith**
The School of Computing
University of Leeds
GB-Leeds LS2 9JT

**Prof. Dr. Renata Sotirov**
Department of Econometrics
Tilburg University
P. O. Box 90153
NL-5000 LE Tilburg

**Prof. Dr. Frank Vallentin**
Faculty of Electrical Engineering,
Applied Mathematics & Computer Sc.
Delft University of Technology
Mekelweg 4
NL-2628 CD Delft