Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

TIMO BERTHOLD AND MARC E. PFETSCH

Detecting Orbitopal Symmetries

Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, {berthold,pfetsch}@zib.de

ZIB-Report 08-33 (August 2008)

Detecting Orbitopal Symmetries

Timo Berthold and Marc E. Pfetsch

August 14, 2008

Abstract

Orbitopes can be used to handle symmetries which arise in integer programming formulations with an inherent assignment structure. We investigate the detection of symmetries appearing in this approach. We show that detecting so-called orbitopal symmetries is graph-isomorphism hard in general, but can be performed in linear time if the assignment structure is known.

Symmetries are usually not desirable in integer programming (IP) models, because they derogate the performance of state-of-the-art IP-solvers like SCIP [1, 2]. The reason for this is twofold: Solutions that are equivalent to ones already discovered are found again and again, which makes the search space "unnecessarily large". Furthermore, the bounds obtained from the linear programming (LP) relaxations are very poor, and the LP-solution is almost meaningless for the decision steps of the IP-solving algorithm. Overall, IP-models mostly suffer much more from inherent symmetries than they benefit from the additional structure.

Margot [5, 6] and Ostrowski et al. [8, 9] handle symmetries in general IPs, without knowing the model giving rise to the instance.

Kaibel et al. [3, 4] took a polyhedral approach to deal with special IP-symmetries. They introduced orbitopes [4], which are the convex hull of 0/1-matrices of size $p \times q$, lexicographically sorted with respect to the columns. For the cases with at most or exactly one 1-entry per row, they give a complete and irredundant linear description of the corresponding orbitopes. These orbitopes can be used to handle symmetries in IP-formulations in which assignment structures appear, such as graph coloring problems; see the next section for an example.

All of the above approaches assume that the symmetry has been detected in advance or is known. Therefore, automatic detection of symmetries in a given IP-formulation is an important task.

In this paper, we deal with the detection of orbitopal symmetries that arise in the orbitopal approach. While this problem is polynomially equivalent to the graph automorphism problem, whose complexity is an open problem, orbitopal symmetries can be found in linear time, if at least the assignment structure is given. Otherwise we show that the problem is as hard as the graph automorphism problem.

1 Symmetries in Binary Programs

In this section, we introduce symmetries in binary programs and their detection by color-preserving graph automorphisms.

For any $k \in \mathbb{N}$, let $[k]$ denote the set $\{1, \dots, k\}$. Let $m, n \in \mathbb{N}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$. We deal with *binary programs (BPs)* in the following form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & x_j \in \{0,1\} \qquad \text{for all } j \in [n]. \end{aligned} \tag{1}$$

Without loss of generality, we assume that there is no zero row in A and that no two rows in A are positive multiples of each other. Let N denote the number of nonzero entries of A. For $k \in \mathbb{N}$, let $\mathfrak{S}(k)$ denote the full symmetric group of order k.

For $x \in \mathbb{R}^n, \sigma \in \mathfrak{S}(n)$, we write $\sigma(x)$ for the vector which is obtained by permuting the components of x according to σ, i.e., $\sigma(x)_i = x_{\sigma(i)}$. For $\sigma \in \mathfrak{S}(m)$ and $\pi \in \mathfrak{S}(n)$, we write $A(\sigma, \pi)$ for the matrix which is obtained by simultaneously permuting the rows of A according to σ and the columns of A according to π.

Let $\mathcal{F} \subseteq \{0,1\}^n$ be the set of feasible solutions of BP (1). If there is a permutation $\sigma \in \mathfrak{S}(n)$ such that $x \in \mathcal{F}$ if and only if $\sigma(x) \in \mathcal{F}$, then σ is called a *symmetry* of $\mathcal{F}$. Obviously, the set of all symmetries of $\mathcal{F}$ is a subgroup of $\mathfrak{S}(n)$. Clearly, it is $\mathcal{NP}$-complete to determine whether a binary program has a non-trivial symmetry group.

To avoid this complexity, one concentrates on finding symmetries of the BP-formulation. Focusing on the BP, however, implies that the symmetry depends on the problem formulation. We give a formal definition of symmetry groups, which is similar to the one of Margot [5, 6].

Definition 1.1. *A subgroup $\mathfrak{G}$ of the full symmetric group $\mathfrak{S}(n)$ is a symmetry group of BP (1), if and only if there is a subgroup $\mathfrak{H}$ of $\mathfrak{S}(m)$, s.t. the following conditions hold for all elements $\pi \in \mathfrak{G}$:*

(i) $\pi(c) = c$,

(ii) there exists $\sigma \in \mathfrak{H}$ *s.t.* $\sigma(b) = b$ *and* $A(\sigma, \pi) = A$.

We reduce the problem of finding symmetries in an BP to a graph automorphism problem. Let $V_{\text{row}} \coloneqq \{u_1, \dots, u_m\}$, $V_{\text{col}} \coloneqq \{v_1, \dots, v_n\}$, $V \coloneqq V_{\text{col}} \cup V_{\text{row}}$, $E \coloneqq \{\{u_i, v_k\} \in V_{\text{row}} \times V_{\text{col}} \mid a_{ik} \neq 0\}$, and $G = (V, E)$. Note that G is bipartite and that the size of G is in $O(N)$.

The coefficients of the BP are treated as follows. We introduce a color $\gamma(r) \in \mathbb{N}$ for each value in the set $\{b_1, \dots, b_m\}$. We assign the color $\gamma(r)$ to all vertices $u_i \in V_{\text{row}}$ with $b_i = r$. The same is done for the objective c and the "variable vertices" V_{col}.

We proceed analogously for the matrix coefficients. If at least one $(i,j) \in [m] \times [n]$ with $a_{ij} = r$ exists, color $\hat{\gamma}(r)$ is assigned to all edges $\{u_i, v_j\}$ with $a_{ij} = r$. We call the edge- and vertex-colored graph G the *coefficient graph* of BP (1).

Definition 1.2. *Given a graph $G = (V,E)$, a mapping $\zeta\colon V \mapsto V$ is called an automorphism of G, if it preserves adjacency: $\{u,v\} \in E \Leftrightarrow \{\zeta(u),\zeta(v)\} \in E$. If for a given vertex and edge coloring, the colors of all vertices and edges stay the same under ζ, we call the automorphism* color-preserving.

For two different graphs G and $\hat{G}$, a *color-preserving isomorphism* is defined analogously.

Note that it is neither known whether the decision problem "Is there a non-trivial automorphism of G?" can be solved in polynomial time nor whether it is $\mathcal{NP}$-complete. Nevertheless, there are codes, such as nauty [7], which solve practically relevant graph automorphism and isomorphism instances within reasonable time. Note that computing color-preserving graph automorphisms is polynomially equivalent to computing graph automorphisms. Symmetry detection can be reduced to finding color-preserving graph automorphisms:

Proposition 1.3. *Every symmetry of a binary program induces a color-preserving automorphism of its coefficient graph and vice versa.*

In the following, we want to concentrate on symmetries, which arise from permuting blocks of variables as in the orbitope approach. As an example consider the *maximal k-colorable subgraph problem*. Given a graph G and a number $k \in \mathbb{N}$, the task is to find a subset of vertices $\hat{V} \subseteq V$ such that the subgraph induced by $\hat{V}$ is k-colorable. The standard BP-model for the maximal k-colorable subgraph problem uses binary variables x_{vc} that determine whether color c is assigned to vertex v:

$$\begin{array}{lrll} \max & \sum_{v\in V}\sum_{c\in[k]} x_{vc} & & \\ \text{s.t.} & \sum_{c\in[k]} x_{vc} \le 1 & & \text{for all } v \in V \\ & x_{uc} + x_{vc} \le 1 & & \text{for all } \{u,v\} \in E \text{ and } c \in [k] \\ & x_{vc} \in \{0,1\} & & \text{for all } v \in V \text{ and } c \in [k]. \end{array}$$

For a given k-colorable subgraph, permuting the colors in $[k]$ yields an orbit of $k!$ structurally identical solutions. If we consider x as a 0/1-matrix of size $|V| \times k$, permuting color classes corresponds to permuting columns—"blocks of variables"—of the matrix x. Each column/block consists of $|V|$ variables x_{vc} belonging to a particular color c.

We generalize this structure as follows: Let $q \in \mathbb{N}$ divide n and let $\mathcal{C} := \{C_1, \ldots, C_q\}$ be a partition of the column index set of A into q distinct subsets of the same cardinality. We call C_j a *variable block*.

Groups acting on variable blocks, like $\mathfrak{S}(k)$ in the k-colorable subgraph example, are called orbitopal symmetries. More precisely, the

group $\mathfrak{S}(q)$ is called an *orbitopal symmetry* of BP (1), if for all $j, \hat{\jmath} \in [q]$ there exists a bijection $\pi_{j\hat{\jmath}} \colon C_j \to C_{\hat{\jmath}}$ such that $c_k = c_{\pi_{j\hat{\jmath}}(k)}$ for all $k \in C_j$, and for every row $i \in [m]$

$$\sum_{k \in C_j} a_{ik} x_k \qquad + \sum_{k \in C_{\hat{\jmath}}} a_{ik} x_k \qquad + \sum_{\ell \neq j, \hat{\jmath}} \sum_{k \in C_\ell} a_{ik} x_k \quad \leq b_i \qquad (2)$$

there exists a row $\hat{\imath} \in [m]$ that has the form

$$\sum_{k \in C_j} a_{i\pi_{j\hat{\jmath}}(k)} x_k \quad + \sum_{k \in C_{\hat{\jmath}}} a_{i\pi^{-1}_{j\hat{\jmath}}(k)} x_k \quad + \sum_{\ell \neq j, \hat{\jmath}} \sum_{k \in C_\ell} a_{ik} x_k \quad \leq b_i. \qquad (3)$$

Let $j, \hat{\jmath} \in [q]$ and $\sigma_{j\hat{\jmath}} \colon [m] \to [m]$, $i \mapsto \hat{\imath}$ be the mapping which links the rows of A. Note that, since there are no identical rows, $\sigma_{j\hat{\jmath}} = \sigma^{-1}_{\hat{\jmath}j}$, in particular $\sigma_{jj} = \mathrm{id}$. For the k-colorable subgraph problem, there is an orbitopal symmetry acting on the blocks of variables associated to a common color.

The set of maps $\pi : [n] \to [n]$ defined by $\pi_{j\hat{\jmath}}$ on C_j, $\pi^{-1}_{j\hat{\jmath}}$ on $C_{\hat{\jmath}}$, and the identity on the remaining elements forms a symmetry of BP (1). Indeed, condition (i) of Definition 1.1 is fulfilled and the requirements (2) and (3) show that condition (ii) holds as well.

2 Complexity of Detecting Orbitopal Symmetries

In the following, we want to describe a polynomial time algorithm, which is able to verify whether a partition $\mathcal{C}$ induces an orbitopal symmetry of an BP without having knowledge of the mappings π, σ.

Definition 2.1. *Let $S \subseteq [n]$, $\ell \in \mathbb{N}$, $\bowtie \in \{\leq, =, \geq\}$.*

(i) A linear constraint of the form $\sum_{k \in S} x_k \ \bowtie \ \ell$ is called a leading constraint of $\mathcal{C}$*, if it stays invariant under the mappings $\pi_{j\hat{\jmath}}$ and contains exactly one variable from each variable block C_j.*

(ii) We call a set $\mathcal{S} \coloneqq \{S_1, \ldots, S_p\}$ of leading constraints a leading system of $\mathcal{C}$ *if every variable is contained in exactly one S_i.*

In the above definition, we identify a leading constraint with its set of variable indices. Note that for a leading system $n = pq$ holds.

For the maximal k-colorable subgraph problem, the set packing constraints $\sum_{c \in [k]} x_{vc} \leq 1$ form a leading system. The leading constraints determine the orbit of a variable under the symmetry $\mathfrak{S}(q)$.

Proposition 2.2. *Given a set of variable blocks $\mathcal{C}$ and a leading system $\mathcal{S}$ of $\mathcal{C}$, verifying that they describe an orbitopal symmetry of the binary program (1) is possible in $O(qmN)$ time.*

Proof. Let $j, \hat{\jmath} \in [q]$ and $i \in [p]$. Let $\{k\} = S_i \cap C_j$ and $\{\hat{k}\} = S_i \cap C_{\hat{\jmath}}$. Recall that S_i is invariant under $\pi_{j\hat{\jmath}}$, which means that $\pi_{j\hat{\jmath}}(k) = \hat{k}$. Hence, constructing the maps $\pi_{j\hat{\jmath}}$ elementwise is possible in $O(N)$ time.

For every constraint (2) of BP (1), row (3) describes its image under $\pi_{j\hat{\jmath}}$. If this image constraint does not exist in BP (1), $\mathcal{C}$ and $\mathcal{S}$ do not yield a symmetry of the BP.

Searching the image row is possible in $O(N)$ time. This search has to be performed for all rows. Hence, checking a variable block pair can be achieved in $O(mN)$ time. It suffices to only check block pairs $\{1, j\}$, since $\pi_{j\hat{\jmath}} = \pi_{1\hat{\jmath}} \circ \pi_{1j}^{-1}$. We get an overall running time of $O(qmN)$. □

The next lemma is tailored towards the typical case, in which removing the leading constraints decomposes the BP into blocks. Let $G(\mathcal{S})$ be the coefficient graph of BP (1) without the leading constraints $\mathcal{S}$.

Lemma 2.3. *Let be $\mathfrak{S}(q)$ be an orbitopal symmetry of BP (1). If for all rows, which are not leading constraints, all non-zeros are within one variable block, then $G(\mathcal{S})$ is partitioned into q components which are pairwise color-preserving isomorphic.*

Note that these components do not have to be connected.

Theorem 2.4. *For a given leading system $\mathcal{S}$, detecting the corresponding orbitopal symmetry is possible in $O(N)$ time.*

If there are no leading constraints, detecting orbitopal symmetries is as hard as the graph isomorphism problem.

Proof. Following Lemma 2.3, we detect the orbitopal symmetry by determining the connected components of the graph $G(\mathcal{S})$. Using a breadth-first-search, this takes $O(|V| + |E|) = O(N)$ time. For checking that all components are pairwise isomorphic, it is sufficient to show that all components are isomorphic to the first component. Let $j \in [q] \setminus \{1\}$. The mappings π_{1j} can be constructed in $O(n)$ time as in the proof of Proposition 2.2 by evaluating $\mathcal{S}$.

For testing whether the π_{1j} describe color-preserving isomorphisms, each edge of the graph has to be regarded. This takes $O(N)$ time.

The reduction to graph isomorphism is achieved as follows. Let two graphs G and $\hat{G}$ with the same number of vertices and edges be given. For each vertex v in one of the graphs, we introduce a distinct binary variable x_v and for each edge $\{u, v\}$, we introduce a set packing constraint $x_u + x_v \leq 1$. The variables are partitioned into two disjoint blocks C_U, C_V. The BP has an orbitopal symmetry arising from the blocks C_U and C_V, if and only if G and $\hat{G}$ are isomorphic. □

One can show that the detecting leading constraints within a BP is also polynomially equivalent to a graph isomorphism problem.

Finally, we want to investigate the case in which the leading constraints do not yield a complete system. As an example, think of a graph

coloring model, which uses variables x_{vc} to assign color c to vertex v, connected by leading constraints $\sum_c x_{vc} = 1$, which ensure that each vertex is colored exactly once. It uses one additional variable y_c per block, which indicates whether color c is used or not. This case can be handled by the following result, which we state without proof:

Corollary 2.5. *If there is a constant number of variables per block, which are not contained in any leading constraint, detecting orbitopal symmetries is possible in $O(N)$ time.*

References

[1] T. Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007.

[2] T. Achterberg, T. Berthold, S. Heinz, M. Pfetsch, and K. Wolter. SCIP – Solving Constraint Integer Programs, documentation. `http://scip.zib.de`.

[3] V. Kaibel, M. Peinhardt, and M. E. Pfetsch. Orbitopal fixing. In M. Fischetti and D. Williamson, editors, *Proc. of the 12th IPCO*, volume 4513 of *LNCS*, pages 74–88. Springer-Verlag, 2007.

[4] V. Kaibel and M. E. Pfetsch. Packing and partitioning orbitopes. *Mathematical Programming*, 114(1):1–36, 2008.

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

[6] F. Margot. Symmetric ILP: Coloring and small integers. *Discrete Optimization*, 4:40–62, 2007.

[7] B. McKay. nauty User's Guide (version 2.4), 2007.

[8] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smirglio. Orbital branching. In M. Fischetti and D. Williamson, editors, *Proc. of the 12th IPCO*, volume 4513 of *LNCS*, pages 104–118. Springer-Verlag, 2007.

[9] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smirglio. Constraint orbital branching. In A. Lodi, A. Panconesi, and G. Rinaldi, editors, *Proc. of the 13th IPCO*, volume 5035 of *LNCS*, pages 225–239. Springer-Verlag, 2008.