Discrete Optimization 8 (2011) 595–610

Contents lists available at SciVerse ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

Orbitopal fixing[☆]

Volker Kaibel[a], Matthias Peinhardt[a], Marc E. Pfetsch[b,*]

[a] Otto-von-Guericke Universität Magdeburg, Fakultät für Mathematik, Universitätsplatz 2, 39106 Magdeburg, Germany

[b] TU Braunschweig, Institute for Mathematical Optimization, Pockelsstr. 14, 38106 Braunschweig, Germany

ARTICLE INFO

Article history:
Received 20 July 2009
Received in revised form 6 October 2010
Accepted 19 July 2011
Available online 27 August 2011

Keywords:
Integer programming
Symmetry
Orbitopes
Graph partitioning
Variable fixing

ABSTRACT

The topic of this paper are integer programming models in which a subset of 0/1-variables encode a partitioning of a set of objects into disjoint subsets. Such models can be surprisingly hard to solve by branch-and-cut algorithms if the order of the subsets of the partition is irrelevant, since this kind of symmetry unnecessarily blows up the search tree.

We present a general tool, called orbitopal fixing, for enhancing the capabilities of branch-and-cut algorithms in solving such symmetric integer programming models. We devise a linear time algorithm that, applied at each node of the search tree, removes redundant parts of the tree produced by the above mentioned symmetry. The method relies on certain polyhedra, called orbitopes, which have been introduced in [14]. It does, however, not explicitly add inequalities to the model. Instead, it uses certain fixing rules for variables. We demonstrate the computational power of orbitopal fixing at the example of a graph partitioning problem.

1. Introduction

Being welcome in most other contexts, symmetry causes severe trouble in the solution of many integer programming (IP) models. This paper describes a method to enhance the capabilities of branch-and-cut algorithms with respect to handling symmetric models of a certain kind that frequently occurs in practice.

We illustrate this kind of symmetry by the example of a graph partitioning problem (another notorious example is the vertex coloring problem). Here, one is given an undirected graph $G = (V, E)$ with non-negative edge weights $w \in \mathbb{Q}^E_{\geq 0}$ and an integer $q \geq 2$. The task is to partition V into q disjoint subsets such that the sum of all weights of edges connecting nodes in the same subset is minimized; thus, this problem is equivalent to maximizing the weights of the edges in a q-cut.

A straight-forward IP model for this graph partitioning problem arises by introducing 0/1-variables x_{ij} for all $i \in [p] := \{1, \ldots, p\}$ and $j \in [q]$ that indicate whether node i is contained in subset j (where we assume $V = [p]$). In order to model the objective function, we furthermore need 0/1-variables y_{ik}, for all edges $\{i, k\} \in E$, indicating whether nodes i and k are contained in the same subset. This yields the following IP-model (see, e.g., [1]):

$$\begin{array}{lll} \min & \sum_{\{i,k\}\in E} w_{ik} y_{ik} & \\ \text{s.t.} & \sum_{j=1}^{q} x_{ij} = 1 & \text{for all } i \in [p] \\ & x_{ij} + x_{kj} - y_{ik} \leq 1 & \text{for all } \{i,k\} \in E,\ j \in [q] \\ & x_{ij} \in \{0,1\} & \text{for all } i \in [p],\ j \in [q] \\ & y_{ik} \in \{0,1\} & \text{for all } \{i,k\} \in E. \end{array} \tag{1}$$

[☆] This work has been partially supported by the DFG Research Center MATHEON in Berlin.

[*] Corresponding author.
E-mail addresses: kaibel@ovgu.de (V. Kaibel), peinhard@ovgu.de (M. Peinhardt), m.pfetsch@tu-bs.de (M.E. Pfetsch).

1572-5286/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.disopt.2011.07.001

The x-variables describe a 0/1-matrix of size $p \times q$ with exactly one 1-entry per row. They encode the assignment of the nodes to the subsets of the partition. The methods that we discuss in this paper do only rely on this structure and thus can be applied to many other models as well. We use the example of the graph partitioning problem as a prototype application and report on computational experiments for this application in Section 7.

Graph partitioning problems are discussed, for example, in [2,3,1]. They arise, for instance, as relaxations of frequency assignment problems in mobile telecommunication networks, see [1]. The maximization version (min k-cut) of the graph partitioning problem is relevant as well [4,5]. Also capacity bounds on the subsets of the partition (which can easily be incorporated into the model) are of interest, in particular for the graph equipartition problem [6–9]. For the closely related clique partitioning problem see [10,11]. Semidefinite relaxations and solution approaches are discussed in [12,13].

As it is given above, the model is unnecessarily difficult for state-of-the-art IP solvers. Even solving small instances requires enormous efforts (see Section 7). One reason is that every feasible solution (x, y) to this model can be turned into $q!$ different ones by permuting the columns of x (viewed as a 0/1-matrix) in an arbitrary way, thereby not changing the structure of the solution (in particular: its objective function value). Phrased differently, the symmetric group of all permutations of the set $[q]$ operates on the solutions by permuting the columns of the x-variables in such a way that the objective function remains constant along each orbit. Therefore, when solving the model by a branch-and-cut algorithm, basically the same work will be done in the tree at many places. Thus, there should be potential for reducing the running times significantly by exploiting this symmetry. A more subtle second point is that interior points of the convex hulls of the individual orbits are responsible for quite weak linear programming (LP) bounds. We will, however, not address this second point in this paper.

In order to remove symmetry, the above model for the graph partitioning problem is often replaced by models containing only edge variables, see, e.g. [6]. For this, however, the underlying graph has to be complete, which might introduce many unnecessary variables. Moreover, formulation (1) is sometimes favorable, e.g., if node-weighted capacity constraints should be incorporated.

One way to deal with symmetry is to restrict the feasible region in each of the orbits to a single representative, e.g., to the lexicographically maximal (with respect to the row-by-row ordering of the x-components) element in the orbit. In fact, this can be done by adding inequalities to the model that enforce the columns of x to be sorted in a lexicographically decreasing way. This can be achieved by $O(pq)$ many *column inequalities*. In [14] even a complete (and irredundant) linear description of the convex hull of all 0/1-matrices of size $p \times q$ with exactly one 1-entry per row and lexicographically decreasing columns is derived; a shorter proof of this completeness result appears in [15]. The corresponding polytopes are called *partitioning orbitopes*. A similar result can be proved for the case of *packing orbitopes*, in which there is at most one 1-entry per row. The descriptions basically consist of an exponentially large super class of column inequalities, called *shifted column inequalities*, for which there is a linear time separation algorithm available. We recall some of these results in Section 2.

Incorporating the inequalities from the orbitope description into the IP model removes symmetry. At each node of the branch-and-cut tree this ensures that the corresponding IP is infeasible as soon as there is no representative in the subtree rooted at that node. In fact, already the column inequalities are sufficient for this purpose.

In this paper, we investigate a way to utilize these inequalities (or the orbitope that they describe) without explicitly adding any of the inequalities to the models. The reason for doing this is the unpleasant effect that adding (shifted) column inequalities to the models might result in more difficult LP relaxations. One way of avoiding the addition of these inequalities to the LPs is to derive logical implications instead: If we are working in a branch-and-cut node at which the x-variables corresponding to index subsets I_0 and I_1 are fixed to zero and one, respectively, then there might be a (shifted) column inequality yielding implications for all representatives in the subtree rooted at the current node. For instance, it might be (and this is easy to check for a given inequality) that for some $(i^\star, j^\star) \notin I_0 \cup I_1$ we have $x_{i^\star j^\star} = 0$ for all 0/1-points x with $x_{ij} = 0$ $((i, j) \in I_0)$ and $x_{ij} = 1$ $((i, j) \in I_1)$ that satisfy the inequality. In this case, $x_{i^\star j^\star}$ can be fixed to zero for the whole subtree rooted at the current node, enlarging I_0. Similarly, also fixings of variables to 1 might be possible. We call the iterated process of searching for such additional fixings *sequential fixing* with (shifted) column inequalities.

Let us mention at this point that deviating from parts of the literature, we do not distinguish between "fixing" and "setting" of variables in this paper.

Sequential fixing with (shifted) column inequalities is a special case of constraint propagation, which is well known from constraint logic programming, see [16–18] for an overview. Modern IP solvers like SCIP [19] use such strategies also in the node preprocessing during the branch-and-cut algorithm. With orbitopes, however, we can aim at something better: Consider a branch-and-cut node identified by fixing the variables corresponding to sets I_0 and I_1 to zero and one, respectively. Denote by $W(I_0, I_1)$ the set of all vertices x of the orbitope with $x_{ij} = 0$ for all $(i, j) \in I_0$ and $x_{ij} = 1$ for all $(i, j) \in I_1$. We define the sets $I_0^\star$ and $I_1^\star$ of all indices $(i^\star, j^\star)$ of variables, for which all x in $W(I_0, I_1)$ satisfy $x_{i^\star j^\star} = 0$ and $x_{i^\star j^\star} = 1$, respectively. We call the respective fixing of the variables corresponding to $I_0^\star$ and $I_1^\star$ *simultaneous fixing*. Simultaneous fixing is always at least as strong as sequential fixing.

Investigations of sequential and simultaneous fixing for orbitopes are the central topic of the paper. The main contributions and results are the following:

- We present a linear time algorithm for *orbitopal fixing*, i.e., for solving the problem to compute simultaneous fixings for partitioning orbitopes (Theorem 4) and packing orbitopes (Corollary 1).

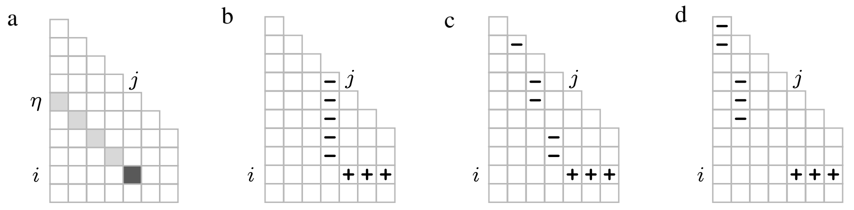


Fig. 1. (a) Example for coordinates $(9, 5) = \langle 5, 5\rangle$. (b), (c), (d) Three shifted column inequalities, (b) being a column inequality.

- In contrast to this, we prove that orbitopal fixing for *covering orbitopes* (the convex hulls of all lexicographically maximal 0/1-matrices with *at least* one 1-entry in every row) is NP-hard (Theorem 5).
- We show that, for general 0/1-polytopes, sequential fixing, even with complete and irredundant linear descriptions, is weaker than simultaneous fixing (Theorem 2). For the case of partitioning orbitopes, we clarify the relationships between different versions of sequential fixing with (shifted) column inequalities, where (despite the situation for general 0/1-polytopes) the strongest one is as strong as orbitopal fixing (Theorem 3).
- We report on computer experiments (Section 7) with the graph partitioning problem described above, showing that orbitopal fixing leads to significant performance improvements for branch-and-cut algorithms.

This paper extends the one that appeared in the proceedings of IPCO XII [20]. It contains the following additional material: a proof for the second part of Theorem 3, the above mentioned results for packing and covering orbitopes (Section 5), a comparison to the related approaches of Margot [21–23] and Linderoth et al. [24,25] for the orbitope case (Section 6), and, finally, computational results for a significantly improved version of our graph partitioning code (Section 7).

While our methods are based on lexicographically maximal choices of representatives from the orbits, a more general approach admitting orderings defined by arbitrary linear functions was introduced by Friedman, see [26]. There are also a number of approaches for symmetry handling available from the constraint logic programming literature, see, e.g., [27–29]. Their general idea is similar to the above mentioned approaches by Margot and Linderoth et al. During the traversal of the tree, different techniques are used to avoid the processing of (some) symmetric parts of the tree. For an excellent survey of methods for symmetry breaking in integer programming we refer to [30].

2. Orbitopes

Throughout the paper, let p and q be integers with $p \geq q \geq 2$. The *partitioning/packing/covering orbitope* $O^{=}_{p,q}$ / $O^{\leq}_{p,q}$ / $O^{\geq}_{p,q}$ is the convex hull of all 0/1-matrices $x \in \{0,1\}^{[p]\times[q]}$ with exactly/at most/at least one 1-entry per row, whose columns are in non-increasing lexicographical order, i.e., they satisfy

$$\sum_{i=1}^{p} 2^{p-i} x_{ij} \geq \sum_{i=1}^{p} 2^{p-i} x_{i,j+1} \tag{2}$$

for all $j \in [q-1]$.

We will mainly be concerned with partitioning orbitopes $O^{=}_{p,q}$. An exception is Section 5, in which we will show that the linear time method for orbitopal fixing of Section 4 can easily be carried over to packing orbitopes $O^{\leq}_{p,q}$, while there is no polynomial time method for orbitopal fixing for covering orbitopes $O^{\geq}_{p,q}$, unless P = NP.

Let the symmetric group of size q act on $\{0,1\}^{[p]\times[q]}$ via permutation of the columns. Then the vertices of $O^{=}_{p,q}$ are exactly the lexicographically maximal matrices with exactly one 1-entry per row in the orbits under the symmetric group action; the lexicographic order is defined as in (2).

As these vertices have $x_{ij} = 0$ for all (i,j) with $i < j$, we drop these components and consider $O^{=}_{p,q}$ as a subset of the space $\mathbb{R}^{\mathcal{I}_{p,q}}$ with $\mathcal{I}_{p,q} := \{(i,j) \in \{0,1\}^{[p]\times[q]} \ : \ i \geq j\}$. Thus, we consider matrices, in which the i-th row has $q(i) := \min\{i, q\}$ components.

The main result in [14] is a complete linear description of $O^{=}_{p,q}$. In order to describe the result, it will be convenient to address the elements in $\mathcal{I}_{p,q}$ via a different "system of coordinates": For $j \in [q]$ and $1 \leq \eta \leq p - j + 1$, define $\langle \eta, j\rangle := (j + \eta - 1, j)$. Thus (as before) i and j denote the row and the column, respectively, while η is the index of the diagonal (counted from above) containing the respective element; see Fig. 1(a) for an example.

A set $S = \{\langle 1, c_1\rangle, \langle 2, c_2\rangle, \ldots, \langle \eta, c_\eta\rangle\} \subset \mathcal{I}_{p,q}$ with $c_1 \leq c_2 \leq \cdots \leq c_\eta$ and $\eta \geq 1$ is called a *shifted column*. For $(i,j) = \langle \eta, j\rangle \in \mathcal{I}_{p,q}$, a shifted column S as above with $c_\eta < j$, and the set $B = \{(i,j), (i,j+1), \ldots, (i,q(i))\}$, we call $x(B) - x(S) \leq 0$ a *shifted column inequality*. The set B is called its *bar*. In case of $c_1 = \cdots = c_\eta = j - 1$ the shifted column inequality is called a *column inequality*. See Fig. 1 for examples.

Finally, a bit more notation is needed. For each $i \in [p]$, we define $\text{row}_i := \{(i,j) \ : \ j \in [q(i)]\}$. For $A \subset \mathcal{I}_{p,q}$ and $x \in \mathbb{R}^{\mathcal{I}_{p,q}}$, we denote by $x(A)$ the sum $\sum_{(i,j)\in A} x_{ij}$.

Theorem 1 (*See [14]*)**.** *The orbitope $\mathrm{O}^{=}_{p,q}$ is completely described by the non-negativity constraints $x_{ij} \geq 0$, the row-sum equations $x(\mathrm{row}_i) = 1$, and the shifted column inequalities.*

In fact, in [14] it is also shown that, up to a few exceptions, the inequalities in this description define facets of $\mathrm{O}^{=}_{p,q}$. Furthermore, a linear time separation algorithm for the exponentially large class of shifted column inequalities is given. For a compact extended formulation of $\mathrm{O}^{=}_{p,q}$ that also leads to a simplified proof of Theorem 1, see [15].

3. The geometry of fixing variables

In this section, we deal with general 0/1-integer programs and, in particular, their associated polytopes. We will define some basic terminology used later in the special treatment of orbitopes, and we are going to shed some light on the geometric situation of fixing variables.

For some positive integer d, we denote by

$$\mathrm{C}^d = \{x \in \mathbb{R}^d \;:\; 0 \leq x_i \leq 1 \text{ for all } i \in [d]\}$$

the 0/1-cube, where $[d]$ is the corresponding set of indices of variables. For two disjoint subsets $I_0, I_1 \subseteq [d]$ (hence, $I_0 \cap I_1 = \varnothing$) we call

$$\{x \in \mathrm{C}^d \;:\; x_i = 0 \text{ for all } i \in I_0,\ x_i = 1 \text{ for all } i \in I_1\}$$

the *face of* C^d *defined by* (I_0, I_1). All nonempty faces of C^d are of this type.

For a polytope $P \subseteq \mathrm{C}^d$ and for a face F of C^d defined by (I_0, I_1), we denote by $\mathrm{Fix}_F(P)$ the smallest face of C^d that contains $P \cap F \cap \{0,1\}^d$ (i.e., $\mathrm{Fix}_F(P)$ is the intersection of all faces of C^d that contain $P \cap F \cap \{0,1\}^d$). If $\mathrm{Fix}_F(P)$ is the nonempty cube face defined by $(I_0^\star, I_1^\star)$, then $I_0^\star$ and $I_1^\star$ consist of all $i \in [d]$ for which $x_i = 0$ and $x_i = 1$, respectively, holds for all $x \in P \cap F \cap \{0,1\}^d$. In particular, we have $I_0 \subseteq I_0^\star$ and $I_1 \subseteq I_1^\star$, or $\mathrm{Fix}_F(P) = \varnothing$. Thus, if I_0 and I_1 are the indices of the variables fixed to zero and one, respectively, in the current branch-and-cut node (with respect to an IP with feasible points $P \cap \{0,1\}^d$), the node can either be pruned, or the sets $I_0^\star$ and $I_1^\star$ yield the maximal sets of variables that can be fixed to zero and one, respectively, for the whole subtree rooted at this node. Unless $\mathrm{Fix}_F(P) = \varnothing$, we call $(I_0^\star, I_1^\star)$ the *fixing of* P *at* (I_0, I_1). Similarly, we call $\mathrm{Fix}_F(P)$ the *fixing* of P at F.

Remark 1. If $P, P' \subseteq \mathrm{C}^d$ are two polytopes with $P \subseteq P'$ and F and F' are two faces of C^d with $F \subseteq F'$, then $\mathrm{Fix}_F(P) \subseteq \mathrm{Fix}_{F'}(P')$ holds.

In general, it is not clear how to compute fixings efficiently. Indeed, computing the fixing of P at $(\varnothing, \varnothing)$ includes deciding whether $P \cap \{0,1\}^d = \varnothing$, which, of course, is NP-hard in general. On the other hand, the following holds.

Lemma 1. *If one can optimize a linear function over $P \cap \{0,1\}^d$ in polynomial time, the fixing $(I_0^\star, I_1^\star)$ at (I_0, I_1) can be computed in polynomial time.*

Proof. Let $c \in \mathbb{R}^d$ be the objective function vector defined by

$$c_i = \begin{cases} 1 & \text{if } i \in I_1 \\ -1 & \text{if } i \in I_0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i \in [d].$$

For each $i^\star \in [d] \setminus (I_0 \cup I_1)$ we have

$$\max\{(c + \mathbb{e}_{i^\star})^T x \;:\; x \in P \cap \{0,1\}^d\} < |I_1| + 1$$

(where $\mathbb{e}_i$ is the ith unit vector) if and only if $i^\star \in I_0^\star$, and

$$\max\{(c - \mathbb{e}_{i^\star})^T x \;:\; x \in P \cap \{0,1\}^d\} < |I_1|$$

if and only if $i^\star \in I_1^\star$. Thus, we can compute $I_0^\star$ and $I_1^\star$ by solving $2(d - |I_0| - |I_1|)$ many linear optimization problems over $P \cap \{0,1\}^d$. $\square$

Note that the reverse to the implication stated in Lemma 1 does not hold, in general. This can, e.g., be seen at the example of 0/1-knapsack problems with $P = \{x \in \mathbb{R}^d \;:\; \sum_{i=1}^d a_i x_i \leq b\}$ (with $a_1, \ldots, a_d \geq 0$). For every (I_0, I_1) the fixing $(I_0^\star, I_1^\star)$ can be computed in linear time:

$$I_0^\star = I_0 \dot\cup \left\{ i^\star \notin I_0 \cup I_1 a_{i^\star} + \sum_{j \in I_1} a_j > b \right\}, \quad I_1^\star = I_1.$$

In contrast, the optimization problem over $P \cap \{0,1\}^d$ is NP-hard.

If the linear optimization problem over $P \cap \{0,1\}^d$ cannot be solved efficiently, one can still try to compute (hopefully large) subsets of $I_0^\star$ and $I_1^\star$ by considering relaxations of P. In case of an IP that is based on an intersection with an orbitope, one might use the orbitope as such a relaxation. We will deal with the fixing problem for partitioning orbitopes in Section 4 (and for packing and covering orbitopes in Section 5). Since the optimization problem for partitioning and packing orbitopes can be solved in polynomial time (see [14]), by Lemma 1, the corresponding fixing problems can be solved in polynomial time as well. However, we will even describe linear time algorithms for these cases.

If P is given via an inequality description, one possibility is to use the knapsack relaxations obtained from single inequalities among the description. For each of these relaxations, the fixing can easily be computed. If the inequality system describing P is exponentially large, and the inequalities are only accessible via a separation routine, it might in some cases nevertheless be possible to decide efficiently whether any of the exponentially many knapsack relaxations allows to fix some variable (see Section 4.2).

Suppose, $P = \{x \in \mathrm{C}^d : Ax \leq b\}$ and $P_r = \{x \in \mathrm{C}^d : a_r^T x \leq b_r\}$ is the knapsack relaxation of P for the rth-row $a_r^T x \leq b_r$ of $Ax \leq b$, where $r = 1, \ldots, m$. Let F be some face of C^d. The face G of C^d obtained by setting $G := F$ and then iteratively replacing G by $\mathrm{Fix}_G(P_r)$ as long as there is some $r \in [m]$ with $\mathrm{Fix}_G(P_r) \subsetneq G$, is denoted by $\mathrm{Fix}_F(Ax \leq b)$. Note that the outcome of this procedure is independent of the choices made for r, due to Remark 1. We call the pair $(\tilde{I}_0, \tilde{I}_1)$ defining the cube face $\mathrm{Fix}_F(Ax \leq b)$ (unless this face is empty) the *sequential fixing of* $Ax \leq b$ *at* (I_0, I_1). In the context of sequential fixing we often refer to (the computation of) $\mathrm{Fix}_F(P)$ as *simultaneous fixing*.

Due to Remark 1, it is clear that $\mathrm{Fix}_F(P) \subseteq \mathrm{Fix}_F(Ax \leq b)$ holds.

Theorem 2. *In general, even for a system of facet-defining inequalities describing a full-dimensional* 0/1*-polytope, sequential fixing is weaker than simultaneous fixing.*

Proof. The following example shows this. Let $P \subset \mathrm{C}^4$ be the four-dimensional polytope defined by the trivial inequalities $x_i \geq 0$ for $i \in \{1, 2, 3\}$, $x_i \leq 1$ for $i \in \{1, 2, 4\}$, the inequality $-x_1 + x_2 + x_3 - x_4 \leq 0$ and $x_1 - x_2 + x_3 - x_4 \leq 0$. Let F be the cube face defined by $(\{4\}, \varnothing)$. Then, sequential fixing does not fix any further variable, although simultaneous fixing yields $I_0^\star = \{3, 4\}$ (and $I_1^\star = \varnothing$). Note that P has only 0/1-vertices, and all inequalities are facet defining ($x_4 \geq 0$ and $x_3 \leq 1$ are implied). □

4. Fixing variables for partitioning orbitopes

For this section, suppose that $I_0, I_1 \subseteq \mathcal{I}_{p,q}$ are subsets of indices of partitioning orbitope variables with the following properties:

(P1) $|I_0 \cap \mathrm{row}_i| \leq q(i) - 1$ for all $i \in [p]$.
(P2) For all $(i, j) \in I_1$, we have $(i, \ell) \in I_0$ for all $\ell \in [q(i)] \setminus \{j\}$.

In particular, P1 and P2 imply that $I_0 \cap I_1 = \varnothing$. Let F be the face of the 0/1-cube $\mathrm{C}^{\mathcal{I}_{p,q}}$ defined by (I_0, I_1). Note that if P1 is not fulfilled, then $\mathrm{O}_{p,q}^{=} \cap F = \varnothing$. The following statement follows immediately from Property P2.

Remark 2. If a vertex x of $\mathrm{O}_{p,q}^{=}$ satisfies $x_{ij} = 0$ for all $(i, j) \in I_0$, then $x \in F$.

We assume that the face $\mathrm{Fix}_F(\mathrm{O}_{p,q}^{=})$ is defined by $(I_0^\star, I_1^\star)$, if $\mathrm{Fix}_F(\mathrm{O}_{p,q}^{=})$ is not empty. *Orbitopal fixing* (for partitioning orbitopes) is the problem to compute the simultaneous fixing $(I_0^\star, I_1^\star)$ from (I_0, I_1), or determine that $\mathrm{Fix}_F(\mathrm{O}_{p,q}^{=}) = \varnothing$.

Remark 3. If $\mathrm{Fix}_F(\mathrm{O}_{p,q}^{=}) \neq \varnothing$, it is enough to determine $I_0^\star$, as we have $(i, j) \in I_1^\star$ if and only if $(i, \ell) \in I_0^\star$ holds for all $\ell \in [q(i)] \setminus \{j\}$.

4.1. Intersection of partitioning orbitopes with cube faces

We start by deriving some structural results on partitioning orbitopes that are crucial in our context. Since $\mathrm{O}_{p,q}^{=} \subset \mathrm{C}^{\mathcal{I}_{p,q}}$ is a 0/1-polytope (i.e., it is integral), we have $\mathrm{conv}(\mathrm{O}_{p,q}^{=} \cap F \cap \{0, 1\}^{\mathcal{I}_{p,q}}) = \mathrm{O}_{p,q}^{=} \cap F$. Thus, $\mathrm{Fix}_F(\mathrm{O}_{p,q}^{=})$ is the smallest cube face that contains the face $\mathrm{O}_{p,q}^{=} \cap F$ of the orbitope $\mathrm{O}_{p,q}^{=}$.

Let us, for $i \in [p]$, define values $\alpha_i := \alpha_i(I_0) \in [q(i)]$ recursively by setting $\alpha_1 := 1$ and, for all $i \in [p]$ with $i \geq 2$,

$$\alpha_i := \begin{cases} \alpha_{i-1} & \text{if } \alpha_{i-1} = q(i) \text{ or } (i, \alpha_{i-1} + 1) \in I_0 \\ \alpha_{i-1} + 1 & \text{otherwise.} \end{cases}$$

The set of all indices of rows, in which the α-value increases, is denoted by

$$\Gamma(I_0) := \{i \in [p] : i \geq 2, \alpha_i = \alpha_{i-1} + 1\} \cup \{1\}$$

(where, for technical reasons, 1 is included).

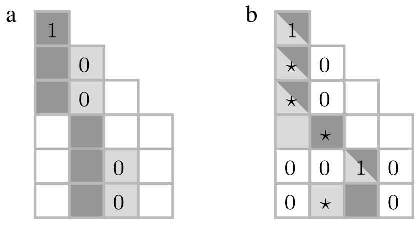


Fig. 2. (a) Example for Remark 4. Dark-gray entries indicate entries $(i, \alpha_i(I_0))$ and light-gray entries indicate the entries in $S_i(I_0)$ for $i = 6$. (b) Example for Lemma 3. As before, dark-gray entries indicate entries (i, α_i). Light-gray entries indicate entries $(i, \mu_i(I_0))$. The $\star$'s indicate 1s set in the point $x^\star$ as constructed in Lemma 3.

The following observation follows readily from the definitions.

Remark 4. For each $i \in [p]$ with $i \geq 2$ and $\alpha_i(I_0) < q(i)$, the set $S_i(I_0) := \{(k, \alpha_k(I_0) + 1) : k \in [i] \setminus \Gamma(I_0)\}$ is a shifted column with $S_i(I_0) \subseteq I_0$.

Fig. 2(a) shows an example.

Lemma 2. *For each $i \in [p]$, no vertex of $\mathrm{O}^{=}_{p,q} \cap F$ has its 1-entry in row i in a column $j \in [q(i)]$ with $j > \alpha_i(I_0)$.*

Proof. Let $i \in [p]$. We may assume $\alpha_i(I_0) < q(i)$, because otherwise the statement is trivially true. Thus, $B := \{(i, j) \in \mathrm{row}_i : j > \alpha_i(I_0)\} \neq \varnothing$.

Let us first consider the case $i \in \Gamma(I_0)$. As we have $\alpha_i(I_0) < q(i) \leq i$ and $\alpha_1(I_0) = 1$, there must be some $k < i$ such that $k \notin \Gamma(I_0)$. Let k be maximal with this property. Thus, we have $k' \in \Gamma(I_0)$ for all $1 < k < k' \leq i$. According to Remark 4, $x(B) - x(S_k(I_0)) \leq 0$ is a shifted column inequality with $x(S_k(I_0)) = 0$, showing $x(B) = 0$ as claimed in the lemma.

Thus, let us suppose $i \in [p] \setminus \Gamma(I_0)$. If $\alpha_i(I_0) \geq q(i) - 1$, the claim holds trivially. Otherwise, $B' := B \setminus \{(i, \alpha_i(I_0) + 1)\} \neq \varnothing$. Similarly to the first case, now the shifted column inequality $x(B') - x(S_{i-1}(I_0)) \leq 0$ proves the claim. □

For each $i \in [p]$, we define $\mu_i(I_0) := \min\{j \in [q(i)] : (i, j) \notin I_0\}$. Because of Property P1, the sets over which we take minima here are non-empty.

Lemma 3. *If we have $\mu_i(I_0) \leq \alpha_i(I_0)$ for all $i \in [p]$, then the point $x^\star = x^\star(I_0) \in \{0, 1\}^{\mathcal{I}_{p,q}}$ defined by $x^\star_{i,\alpha_i(I_0)} = 1$ for all $i \in \Gamma(I_0)$, $x^\star_{i,\mu_i(I_0)} = 1$ for all $i \in [p] \setminus \Gamma(I_0)$, and all other components being zero, is contained in $\mathrm{O}^{=}_{p,q} \cap F$.*

Proof. Due to $\alpha_i(I_0) \leq \alpha_{i-1}(I_0) + 1$ for all $i \in [p]$ with $i \geq 2$, the point $x^\star$ is contained in $\mathrm{O}^{=}_{p,q}$. It follows from the definitions that $x^\star$ does not have a 1-entry at a position in I_0. Thus, by Remark 2, we have $x^\star \in F$. □

Fig. 2(b) shows an example for the point constructed in Lemma 3.

We now characterize the case $\mathrm{O}^{=}_{p,q} \cap F = \varnothing$ (leading to pruning the corresponding node in the branch-and-cut tree) and describe the set $I_0^\star$.

Proposition 1.

(1) *We have $\mathrm{O}^{=}_{p,q} \cap F = \varnothing$ if and only if there exists $i \in [p]$ with $\mu_i(I_0) > \alpha_i(I_0)$.*
(2) *If $\mu_i(I_0) \leq \alpha_i(I_0)$ holds for all $i \in [p]$, then the following is true.*
  (a) *For all $i \in [p] \setminus \Gamma(I_0)$, we have*
  $I_0^\star \cap \mathrm{row}_i = \{(i, j) \in \mathrm{row}_i : (i, j) \in I_0 \text{ or } j > \alpha_i(I_0)\}$.
  (b) *For all $i \in [p]$ with $\mu_i(I_0) = \alpha_i(I_0)$, we have*
  $I_0^\star \cap \mathrm{row}_i = \mathrm{row}_i \setminus \{(i, \alpha_i(I_0))\}$.
  (c) *For all $s \in \Gamma(I_0)$ with $\mu_s(I_0) < \alpha_s(I_0)$ the following holds: If there is some $i \geq s$ with $\mu_i(I_0) > \alpha_i(I_0 \cup \{(s, \alpha_s(I_0))\})$, then we have*
  $I_0^\star \cap \mathrm{row}_s = \mathrm{row}_s \setminus \{(s, \alpha_s(I_0))\}$.
  Otherwise, we have
  $I_0^\star \cap \mathrm{row}_s = \{(s, j) \in \mathrm{row}_s : (s, j) \in I_0 \text{ or } j > \alpha_s(I_0)\}$.

Proof. Part 1 follows from Lemmas 2 and 3 (see also Fig. 3(a)).

In order to prove Part 2, let us assume that $\mu_i(I_0) \leq \alpha_i(I_0)$ holds for all $i \in [p]$. For Part 2a, let $i \in [p] \setminus \Gamma(I_0)$ and $(i, j) \in \mathrm{row}_i$. Due to $I_0 \subseteq I_0^\star$, we only have to consider the case $(i, j) \notin I_0$. If $j > \alpha_i(I_0)$, then, by Lemma 2, we find $(i, j) \in I_0^\star$. Otherwise, the point that is obtained from $x^\star(I_0)$ (see Lemma 3) by moving the 1-entry in position $(i, \mu_i(I_0))$ to position (i, j) is contained in $\mathrm{O}^{=}_{p,q} \cap F$, proving $(i, j) \notin I_0^\star$.

In the situation of Part 2b, the claim follows from Lemma 2 and $\mathrm{O}^{=}_{p,q} \cap F \neq \varnothing$ (due to Part 1).

For Part 2c, let $s \in \Gamma(I_0)$ with $\mu_s(I_0) < \alpha_s(I_0)$ and define $I_0' := I_0 \cup \{(s, \alpha_s(I_0))\}$. It follows that we have $\mu_i(I_0') = \mu_i(I_0)$ for all $i \in [p]$; compare also Fig. 3(b).

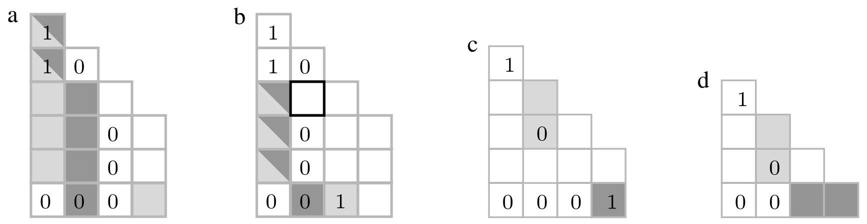


Fig. 3. (a) Example for Proposition 1(1). Light-gray entries indicate the entries $(i, \mu_i(I_0))$ and dark-gray entries indicate entries $(i, \alpha_i(I_0))$. (b) Example of fixing an entry to 1 for Proposition 1(2c). As before light-gray entries indicate entries $(i, \mu_i(I_0))$. Dark-gray entries indicate entries $(i, \alpha_i(I_0 \cup \{(s, \alpha_s(I_0))\}))$ with $s = 3$. (c) and (d) Gray entries show the SCIs used in the proofs of Parts 1(a) and 1(b) of Theorem 3, respectively.

Let us first consider the case that there is some $i \geq s$ with $\mu_i(I_0) > \alpha_i(I_0')$. Part 1 (applied to I_0' instead of I_0) implies that $\mathrm{O}^{=}_{p,q} \cap F$ does not contain a vertex x with $x_{s,\alpha_s(I_0)} = 0$. Therefore, we have $(s, \alpha_s(I_0)) \in I_1^{\star}$, and thus $I_0^{\star} \cap \mathrm{row}_s = \mathrm{row}_s \setminus \{(s, \alpha_s(I_0))\}$ holds (where for "$\subseteq$" we exploit $\mathrm{O}^{=}_{p,q} \cap F \neq \varnothing$ by Part 1, this time applied to I_0).

The other case of Part 2c follows from $s \notin \Gamma(I_0')$ and $\alpha_s(I_0') = \alpha_s(I_0) - 1$. Thus, Part 2a applied to I_0' and s instead of I_0 and i, respectively, yields the claim (because of $(s, \alpha_s(I_0)) \notin I_0^{\star}$ due to $s \in \Gamma(I_0)$ and $\mathrm{O}^{=}_{p,a} \cap F \neq \varnothing$). $\square$

4.2. *Sequential fixing for partitioning orbitopes*

Let us, for some fixed $p \geq q \geq 2$, denote by $\mathcal{S}_{\mathrm{SCI}}$ the system of the non-negativity inequalities, the row-sum equations (each one written as two inequalities, in order to be formally correct) and all shifted column inequalities. Thus, according to Theorem 1, $\mathrm{O}^{=}_{p,q}$ is the set of all $x \in \mathbb{R}^{\mathcal{I}_{p,q}}$ that satisfy $\mathcal{S}_{\mathrm{SCI}}$. Let $\mathcal{S}_{\mathrm{CI}}$ be the subsystem of $\mathcal{S}_{\mathrm{SCI}}$ containing only the column inequalities (and all non-negativity inequalities and row-sum equations).

At first sight, it is not clear whether sequential fixing with the exponentially large system $\mathcal{S}_{\mathrm{SCI}}$ can be done efficiently. A closer look at the problem reveals, however, that one can utilize the linear time separation algorithm for shifted column inequalities (mentioned in Section 2) in order to devise an algorithm for this sequential fixing, whose running time is bounded by $O(\varrho pq)$, where ϱ is the number of variables that are fixed by the procedure.

In fact, one can achieve more: One can compute sequential fixings with respect to the affine hull of the partitioning orbitope. In order to explain this, consider a polytope $P = \{x \in \mathrm{C}^d : Ax \leq b\}$, and let $S \subseteq \mathbb{R}^d$ be some affine subspace containing P. As before, we denote the knapsack relaxations of P obtained from $Ax \leq b$ by $P_1, \ldots, P_m$. Let us define $\mathrm{Fix}_F^S(P_r)$ as the smallest cube face that contains $P_r \cap S \cap \{0, 1\}^d \cap F$. Similarly to the definition of $\mathrm{Fix}_F(Ax \leq b)$, denote by $\mathrm{Fix}_F^S(Ax \leq b)$ the face of C^d that is obtained by setting $G := F$ and then iteratively replacing G by $\mathrm{Fix}_G^S(P_r)$ as long as there is some $r \in [m]$ with $\mathrm{Fix}_G^S(P_r) \subsetneq G$. We call $\mathrm{Fix}_F^S(Ax \leq b)$ the *sequential fixing of* $Ax \leq b$ *at* F *relative to* S. Obviously, we have $\mathrm{Fix}_F(P) \subseteq \mathrm{Fix}_F^S(Ax \leq b) \subseteq \mathrm{Fix}_F(Ax \leq b)$. In contrast to sequential fixing, sequential fixing relative to affine subspaces *in general* is NP-hard (as it can be used to decide whether a linear equation has a 0/1-solution).

Theorem 3.

(1) *There are cube faces F^1, F^2, F^3 with the following properties:*
   (a) $\mathrm{Fix}_{F^1}(\mathcal{S}_{\mathrm{SCI}}) \subsetneq \mathrm{Fix}_{F^1}(\mathcal{S}_{\mathrm{CI}})$
   (b) $\mathrm{Fix}_{F^2}^{\mathrm{aff}(\mathrm{O}^{=}_{p,q})}(\mathcal{S}_{\mathrm{CI}}) \subsetneq \mathrm{Fix}_{F^2}(\mathcal{S}_{\mathrm{SCI}})$
   (c) $\mathrm{Fix}_{F^3}^{\mathrm{aff}(\mathrm{O}^{=}_{p,q})}(\mathcal{S}_{\mathrm{SCI}}) \subsetneq \mathrm{Fix}_{F^3}^{\mathrm{aff}(\mathrm{O}^{=}_{p,q})}(\mathcal{S}_{\mathrm{CI}})$

(2) *For all cube faces F, we have* $\mathrm{Fix}_F^{\mathrm{aff}(\mathrm{O}^{=}_{p,q})}(\mathcal{S}_{\mathrm{SCI}}) = \mathrm{Fix}_F(\mathrm{O}^{=}_{p,q})$.

Proof. For Part (1a), we chose $p = 5, q = 4$, and define the cube face F^1 via $I_0^1 = \{(3, 2), (5, 1), (5, 2), (5, 3)\}$ and $I_1^1 = \{(1, 1), (5, 4)\}$. The shifted column inequality with shifted column $\{(2, 2), (3, 2)\}$ and bar $\{(5, 4)\}$ allows to fix x_{22} to 1 (see Fig. 3(c)), while no column inequality (and no non-negativity constraint and no row-sum equation) allows to fix any variable.

For Part (1b), let $p = 4$, $q = 4$, and define F^2 via $I_0^2 = \{(3, 2), (4, 1), (4, 2)\}$ and $I_1^2 = \{(1, 1)\}$. Exploiting that $x_{43}+x_{44} = 1$ for all $x \in \mathrm{aff}(\mathrm{O}^{=}_{p,q}) \cap F^2$, we can use the column inequality with column $\{(2, 2), (3, 2)\}$ and bar $\{(4, 3), (4, 4)\}$ to fix x_{22} to one (see Fig. 3(d)), while no fixing is possible with $\mathcal{S}_{\mathrm{SCI}}$ only.

For Part (1c), we can use $F^3 = F^1$.

In order to prove Part (2), we have to show for every cube face F that

$$\mathrm{Fix}_F^{\mathrm{aff}(\mathrm{O}^{=}_{p,q})}(\mathcal{S}_{\mathrm{SCI}}) \subseteq \mathrm{Fix}_F(\mathrm{O}^{=}_{p,q})$$

holds. We use the notation introduced in Section 4. The crucial fact is that every point $x \in F \cap \mathrm{aff}(\mathrm{O}^{=}_{p,q})$ satisfies $x(B) = 1$ for every $B \subseteq \mathrm{row}_i$ such that $\{(i, \mu_i(I_0)), \ldots, (i, q(i))\} \subseteq B$.

Algorithm 1 Orbitopal Fixing

```
1: Set $I_0^\star \leftarrow I_0, I_1^\star \leftarrow I_1, \mu_1 \leftarrow 1, \alpha_1 \leftarrow 1$, and $\Gamma = \varnothing$.
2: for $i = 2, \ldots, p$ do
3:    compute $\mu_i \leftarrow \min\{j : (i,j) \notin I_0\}$.
4:    if $\alpha_{i-1} = q(i)$ or $(i, \alpha_{i-1}+1) \in I_0$ then
5:       $\alpha_i \leftarrow \alpha_{i-1}$
6:    else
7:       $\alpha_i \leftarrow \alpha_{i-1}+1, \Gamma \leftarrow \Gamma \cup \{i\}$
8:    if $\mu_i > \alpha_i$ then
9:       return "Orbitopal fixing is empty"
10:   Set $I_0^\star \leftarrow I_0^\star \cup \{(i,j) : j > \alpha_i\}$.
11:   if $|I_0^\star \cap \mathrm{row}_i| = q(i) - 1$ then
12:      set $I_1^\star \leftarrow I_1^\star \cup (\mathrm{row}_i \setminus I_0^\star)$.
13: for all $s \in \Gamma$ with $(s, \alpha_s) \notin I_1^\star$ do
14:   Set $\beta_s \leftarrow \alpha_s - 1$.
15:   for $i = s+1, \ldots, p$ do
16:      if $\beta_{i-1} = q(i)$ or $(i, \beta_{i-1}+1) \in I_0$ then
17:         $\beta_i \leftarrow \beta_{i-1}$
18:      else
19:         $\beta_i \leftarrow \beta_{i-1}+1$
20:      if $\mu_i > \beta_i$ then
21:         $I_1^\star \leftarrow I_1^\star \cup \{(s,\alpha_s)\}$ and $I_0^\star \leftarrow \mathrm{row}_s \setminus \{(s,\alpha_s)\}$.
22:         Proceed with the next $s$ in Step 13.
```

Let us first consider the case $\mathrm{Fix}_F(\mathrm{O}^=_{p,q}) = \varnothing$. Due to Part 1 of Proposition 1 there is some $i \in [p]$ with $\mu_i(I_0) > \alpha_i(I_0)$. Therefore, the SCI $x(\tilde{B}) - x(\tilde{S}) \le 0$ constructed in the proof of Lemma 2 with $(i,j) = (i, \mu_i(I_0))$ has $x(\tilde{B}) = 1$ for all $x \in F \cap \mathrm{aff}(\mathrm{O}^=_{p,q})$, but $x(\tilde{S}) = 0$ due to $\tilde{S} \subseteq I_0$. This shows that we indeed have

$$\mathrm{Fix}_F^{\mathrm{aff}(\mathrm{O}^=_{p,q})}(\mathcal{S}_{\mathrm{SCI}}) = \varnothing$$

in this case.

Otherwise (i.e., $\mathrm{Fix}_F(\mathrm{O}^=_{p,q}) \neq \varnothing$), it suffices to show for each $(k,\ell) \in I_0^\star \setminus I_0$ that there is some SCI that can only be satisfied by some $x \in F \cap \mathrm{aff}(\mathrm{O}^=_{p,q})$ if $x_{k\ell} = 0$ holds. Due to Part 2 of Proposition 1, we have to consider two cases.

Case 1: We have $\ell > \alpha_k(I_0)$. Then the SCI $x(\tilde{B}) - x(\tilde{S}) \le 0$ constructed in the proof of Lemma 2 with $(i,j) = (k,\ell)$ implies $x(\tilde{B}) = 0$ for all $x \in F$ (because of $\tilde{S} \subseteq I_0$), which yields $x_{k\ell} = 0$ due to $(k,\ell) \in \tilde{B}$.

Case 2: We have $k \in \Gamma(I_0)$ with $\ell < \alpha_k(I_0)$ and there is some $r \ge k$ with $\mu_r(I_0) > \alpha_r(I_0 \cup \{(k, \alpha_k(I_0))\})$. Then the SCI $x(\tilde{B}) - x(\tilde{S}) \le 0$ constructed in the proof of Lemma 2 with $(i,j) = (r, \mu_r(I_0))$ (and I_0 replaced by $I_0 \cup \{(k, \alpha_k(I_0))\}$) satisfies, for each $x \in F \cap \mathrm{aff}(\mathrm{O}^=_{p,q})$, $x(\tilde{B}) = 1$ and $x(\tilde{S}) = x_{(k,\alpha_k(I_0))}$ (due to $\tilde{S} \subseteq I_0 \cup \{(k, \alpha_k(I_0))\}$), which implies $x_{(k,\alpha_k(I_0))} = 1$, and hence (as $x \in F \cap \mathrm{aff}(\mathrm{O}^=_{p,q})$) $x_{k\ell} = 0$, because of $\ell \neq \alpha_k(I_0)$. □

The different versions of sequential fixing for partitioning orbitopes are dominated by each other in the following sequence:

$$\mathcal{S}_{\mathrm{CI}} \to \{\mathcal{S}_{\mathrm{SCI}}, \text{affine } \mathcal{S}_{\mathrm{CI}}\} \to \text{affine } \mathcal{S}_{\mathrm{SCI}},$$

which finally is as strong as orbitopal fixing. For each of the arrows there exists an instance for which dominance is strict. The examples in the proof of Theorem 3 also show that there is no general relation between $\mathcal{S}_{\mathrm{SCI}}$ and affine $\mathcal{S}_{\mathrm{CI}}$.

In particular, we could compute orbitopal fixings by the polynomial time algorithm for sequential fixing relative to $\mathrm{aff}(\mathrm{O}^=_{p,q})$. It turns out, however, that this is not the preferable choice. In fact, we will describe below a linear time algorithm for solving the orbitopal fixing problem directly.

4.3. An algorithm for orbitopal fixing

Algorithm 1 describes a method to compute the simultaneous fixing $(I_0^\star, I_1^\star)$ from (I_0, I_1) (which are assumed to satisfy Properties P1 and P2). Note that we use β_i for $\alpha_i(I_0 \cup \{(s, \alpha_s(I_0))\})$.

Theorem 4. *The orbitopal fixing problem for partitioning orbitopes can be solved in time* $\mathrm{O}(pq)$ *(by a slight modification of Algorithm 1).*

Proof. The correctness of the algorithm follows from the structural results given in Proposition 1.

In order to prove the statement on the running time, let us assume that the data structures for the sets $I_0, I_1, I_0^\star$, and $I_1^\star$ allow both membership testing and addition of single elements in constant time (e.g., the sets can be stored as bit vectors).

As none of the Steps 3–12 needs more time than $\mathrm{O}(q)$, we only have to take care of the second part of the algorithm starting in Step 13. (In fact, used verbatim as described above, the algorithm might need time $\Omega(p^2)$.)

For $s, s' \in \Gamma$ with $s < s'$ denote the corresponding β-values by β_i $(i \geq s)$ and by β'_i $(i \geq s')$, respectively. We have $\beta_i \leq \beta'_i$ for all $i \geq s'$, and furthermore, if equality holds for one of these i, we can deduce $\beta_k = \beta'_k$ for all $k \geq i$. Thus, as soon as a pair (i, β_i) is used a second time in Step 20, we can break the for-loop in Step 15 and reuse the information that we have obtained earlier.

This can, for instance, be organized by introducing, for each $(i, j) \in \mathcal{I}_{p,q}$, a flag $f(i, j) \in \{\text{red}, \text{green}, \text{white}\}$ (initialized by white), where $f(i, j) = \text{red / green}$ means that we have already detected that $\beta_i = j$ eventually leads to a positive/negative test in Step 20. The modifications that have to be applied to the second part of the algorithm are the following: The selection of the elements in Γ in Step 13 must be done in increasing order. Before performing the test in Step 20, we have to check whether $f(i, \beta_i)$ is green. If this is true, then we can proceed with the next s in Step 13, after setting all flags $f(k, \beta_k)$ to green for $s \leq k < i$. Similarly, we set all flags $f(k, \beta_k)$ to red for $s \leq k \leq i$, before switching to the next s in Step 22. And finally, we set all flags $f(k, \beta_k)$ to green for $s \leq k \leq p$ at the end of the body of the s-loop starting in Step 13.

As the running time of this part of the algorithm is proportional to the number of flags changed from white to red or green, the total running time indeed is bounded by $O(pq)$ (since a flag is never reset). □

5. Fixing for packing and covering orbitopes

The packing orbitope $O^{\leq}_{p,q}$ obviously can be obtained from the partitioning orbitope $O^{=}_{p+1,q+1}$ by projecting out the first column and row, i.e., by orthogonal projection to the coordinate subspace associated with

$$\{(i, j) \in \mathcal{I}_{p+1,q+1} \; : \; i, j > 1\}$$

(and renaming the variables appropriately), see also [14].

In general, for $J \subseteq [d]$, the orthogonal projection $\pi : \mathbb{R}^{[d]} \to \mathbb{R}^J$, a polytope $P \subseteq C^d$, and some face F of the cube $C^J := [0, 1]^J$, we have

$$\mathrm{Fix}_{\pi(P)}(F) = \pi(\mathrm{Fix}_P(\pi^{-1}(F))),$$

since for every face G of C^J

$$(\pi(P) \cap F) \subseteq G \iff (P \cap \pi^{-1}(F)) \subseteq \pi^{-1}(G)$$

holds (simply because taking preimages commutes with taking intersections).

Thus, the following result for packing orbitopes follows readily from Theorem 4.

Corollary 1. *Variable fixing for packing orbitopes $O^{\leq}_{p,q}$ can be done in time $O(pq)$ by reduction to orbitopal fixing for $O^{=}_{p+1,q+1}$.*

In contrast to this, variable fixing for covering orbitopes $O^{\geq}_{p,q}$ cannot be done in polynomial time, unless P = NP, as the following result implies.

Theorem 5. *The problem to decide whether, for given $I_0 \subseteq [p] \times [q]$, the covering orbitope $O^{\geq}_{p,q}$ contains a vertex $x^\star \in O^{\geq}_{p,q}$ with $x^\star_{ij} = 0$ for all $(i, j) \in I_0$, is* NP*-complete.*

Proof. It suffices to show that one can construct, for each graph $G = (V, E)$ and $k \in \mathbb{N}$ with $k \leq |V|$, (in time bounded polynomially in $|V|$) an instance of the decision problem described in the theorem whose answer is "yes" if and only if G has a vertex cover of size at most k.

Toward this end, let $\kappa := \lceil \log_2(k + 1) \rceil$ be the smallest integer such that we have $\tilde{k} := 2^\kappa - 1 \geq k$. Construct a graph $\tilde{G} = (\tilde{V}, \tilde{E})$ by adding $\tilde{k} - k$ new edges (forming a matching) on $2(\tilde{k} - k)$ nodes disjoint from V. Thus $\tilde{G}$ is a graph with $|\tilde{V}| = |V| + 2(\tilde{k} - k)$ nodes and $m := |\tilde{E}| = |E| + \tilde{k} - k$ edges that has a vertex cover of size at most $\tilde{k}$ if and only if G has a vertex cover of size at most k.

For the instance of the decision problem described in the theorem, let $p := \kappa + m$, $q := 2|\tilde{V}|$, and assume $\tilde{V} = \{2, 4, 6, \ldots, q\}$. Numbering the edges of $\tilde{G}$ by

$$\tilde{E} = \{e_1, e_2, \ldots, e_m\} \quad \text{with } e_h = \{v_h, w_h\} \subseteq \tilde{V} \text{ for all } h \in [m],$$

we set

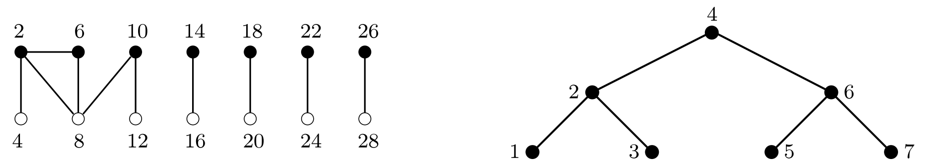
$$I_0 := \{(\kappa + h, j) \; : \; h \in [m], j \in [q] \setminus \{v_h, w_h\}\}.$$

See Fig. 4 for an example.

In order to prove that the answer to the constructed instance is "yes" if and only if $\tilde{G}$ has a vertex cover of size at most $\tilde{k}$, let us call, for $x^\star \in \{0, 1\}^{[p]\times[q]}$, a pair (i, j) an *alibi* (for column j of $x^\star$), if $x^\star_{i,j-1} = 1$ and $x^\star_{ij} = 0$ hold.

If $x^\star \in O^{\geq}_{p,q}$ is a vertex of the covering orbitope $O^{\geq}_{p,q}$ (i.e., a 0/1-point in the orbitope) with $x^\star_{ij} = 0$ for all $(i, j) \in I_0$, then

$$C := \{v \in \tilde{V} \; : \; x^\star_{\kappa+h,v} = 1 \text{ for some } h \in [m]\}$$



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**															
	1	**1**	**1**	**1**	**1**									**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**							
	1					**1**	**1**	**1**	**1**					**1**	**1**	**1**	**1**					**1**	**1**	**1**	**1**			
{2,4}	0	**1**	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
{2,6}	0	**1**	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
{2,8}	0	**1**	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
{6,8}	0	0	0	0	0	**1**	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
{8,10}	0	0	0	0	0	0	0		0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
{10,12}	0	0	0	0	0	0	0	0	0	**1**	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
{14,16}	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0		0	0	0	0	0	0	0	0	0	0	0	0
{18,20}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0		0	0	0	0	0	0	0	0
{22,24}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0		0	0	0	0
{26,28}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	

Fig. 4. Example for the construction in the proof of Theorem 5. Let G be the depicted graph on the top left, and let $k = 7$, i.e., $\kappa = \lceil \log_2(k+1) \rceil = 3$ and $\tilde{k} = 7$. Thus, no new edge is needed, and we have $G = \tilde{G}$. In the matrix, 0s correspond to elements of I_0, **1**s have been set in the construction of a feasible solution, and empty entries correspond to 0s set in the construction. The top right shows the binary tree used for the construction with $a_1, \ldots, a_7 = 2, 6, 10, 14, 18, 22, 26$.

is a vertex cover in $\tilde{G}$ (due to $x^\star(\text{row}_{\kappa+h}) \geq 1$ for all $h \in [m]$). Moreover, for every $v \in C$, there is an alibi (i, v) in some row $i \in [\kappa]$, since column v of $x^\star$ is lexicographically not larger than column $v - 1$. Again due to the lexicographical ordering of the columns, every vertex of the orbitope can have at most 2^{i-1} alibis in row i. It follows that

$$|C| \leq \sum_{i=1}^{\kappa} 2^{i-1} = \sum_{i=0}^{\kappa-1} 2^i = 2^\kappa - 1 = \tilde{k}.$$

Conversely, suppose $C \subseteq \tilde{V}$ is a vertex cover with $|C| \leq \tilde{k}$. We construct a 0/1-point $x^\star \in \mathrm{O}^{\geq}_{p,q}$ with $x^\star_{ij} = 0$ for all $(i, j) \in I_0$ as follows. First, for each $h \in [m]$, we set

$$x^\star_{\kappa+h,v_h} := \begin{cases} 1 & \text{if } v_h \in C \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad x^\star_{\kappa+h,w_h} := \begin{cases} 1 & \text{if } w_h \in C \\ 0 & \text{otherwise.} \end{cases}$$

Since C is a vertex cover of $\tilde{G}$, the part of $x^\star$ that we have already constructed has at least one 1-entry in every row $\kappa+1, \ldots, p$. It thus remains to construct the first κ rows such that they contain an alibi for every column v with $v \in C$ (and such that each of these rows contains at least one 1-entry). This can, e.g., be achieved as follows.

First, choose an arbitrary sequence $(a_1, \ldots, a_{\tilde{k}})$ (of length $\tilde{k} = 2^\kappa - 1$) of numbers in $\tilde{V} = \{2, 4, 6, \ldots, q\}$ with $C \subseteq \{a_1, \ldots, a_{\tilde{k}}\}$ (which is possible due to $|C| \leq \tilde{k}$).

Then a complete rooted binary tree (embedded into the plane) of height κ (having $2^\kappa - 1 = \tilde{k}$ nodes) is constructed in which the nodes receive pairwise different labels $1, 2, \ldots, \tilde{k}$. Furthermore, the labels have to be assigned in such a way that for every node labeled t, we have $a_\ell \leq a_t$ for all labels ℓ in the *left* subtree and $a_t \leq a_r$ for all labels r in the *right* subtree rooted at t.

Then we complete $x^\star$ to a vertex of $\mathrm{O}^{\geq}_{p,q}$ by putting in each row $i \in [\kappa]$ alibis at all positions (i, a_t) for t running through all labels of nodes at distance $i - 1$ from the root of the tree and filling the remaining components of $x^\star$ accordingly. □

Of course, Theorem 5 implies that optimization over covering orbitopes is NP-hard. In particular, in contrast to the packing and partitioning orbitopes, we cannot expect to find a tractable linear description of $\mathrm{O}^{\geq}_{p,q}$, unless NP = coNP.

In fact, using ideas of the proof of Theorem 5, one can also establish other similar statements, in which $\mathrm{O}^{\geq}_{p,q}$ is replaced by the convex hull of all 0/1-matrices (whose columns are in lexicographically non-increasing order) with *at least* k one-entries per row for each $k \geq 1$, or with *exactly* k one-entries per row for each $k \geq 2$.

6. Comparison with isomorphism pruning and orbital branching

In [21,22] Margot developed a related, but more general approach to symmetry breaking, called *isomorphism pruning*. The main components are a setting rule for variables and a pruning rule for nodes in a branch-and-bound tree to avoid consideration of equivalent (partial) solutions. In this section we outline the differences and similarities between Margot's and our approach when specialized to the type of symmetries addressed by partitioning orbitopes.

Isomorphism pruning deals with arbitrary symmetries in any binary program $\min\{c^T x : Ax \leq b, x \in \{0, 1\}^d\}$, or even integer program [22]. Let G be a group of permutations of the variables (inducing permutations of the components of c, the columns of A, and the set of feasible solutions) such that for every $g \in G$ we have $g(c) = c$ and $g(A) = \sigma_g(A)$, $g(b) = \sigma_g(b)$

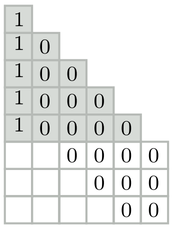


Fig. 5. Example of a branch-and-bound node in which the variables in the first five rows have been fixed by minimum index branching. The zeros below the fifth row have been set by orbitopal fixing after processing the current node, but not by isomorphism pruning.

for some permutation σ_g of the rows of A resp. components of b. In particular, G acts on the set of feasible solutions (via coordinate permutations) with the property that the objective function $c^T x$ is constant on every orbit. Given an order of the variables (a *rank vector* R), Margot's approach then assures that only (partial) solutions that are lexicographically minimal in their orbit under G are explored in the branch-and-bound tree. More precisely, a partial solution in some branch-and-bound node N is identified with the sets I_1^N and I_0^N of variables that have been fixed to 1 and 0, respectively, in the path from the root to N. A partial solution is lex-min in its orbit, i.e., it is a *representative*, if the set I_1^N is lexicographically minimal with respect to the rank vector R. Thus, N can be pruned if it is not a representative.

Note that the definition of the lexicographical order relies upon a particular total order of the variables defined by the rank vector R. It should be mentioned that in [22] this limitation was relaxed by using an arbitrary order that can be determined during the branch-and-bound process. Still, the same rank vector has to be used throughout. Essentially, this means that whenever a branching is to be performed at some level of the branch-and-bound tree for the first time, the branching variable for this level can be freely chosen, and the rank vector is extended. In [31] (see also [30]), the restriction of a global rank vector is dropped, too. The rank vector at some node N of the branch-and-bound tree is now given by the branching decisions from the root node to N.

At every node N of the branch-and-bound tree, two 0-*setting operations* are performed. If N was created by fixing the variable x_f to 0 then all variables in the (sub)orbit of x_f under the stabilizer of I_1^N are set to 0, too, since a 1 for any of these variables would lead to a partial solution lex-greater than I_1^N. Furthermore, in case the index h of the next branching variable is known (e.g., if a global rank vector is available), it is repeatedly tested whether any representative can be reached from N by checking whether the current representative together with h is also a representative, i.e., whether $I_1^N \cup \{h\}$ is lexicographically minimal under G.

For a comparison with orbitopal fixing, we consider isomorphism pruning specialized for partitioning problems with 0/1-variables x_{ij} satisfying $\sum_j x_{ij} = 1$ for all i, where the symmetry group G is assumed to be the group of all permutations of the columns of the variable matrix x. Note that such problems do not require the elaborate machinery of group theoretic algorithms developed by Margot for the general case of more complicated symmetry groups.

We assume that the *canonical rank vector* R is used, i.e., the one that describes the row-wise ordering of the variables x_{ij}. For this ordering, the representatives used by isomorphism pruning are in one-to-one correspondence with faces of the cube $[0, 1]^{\mathcal{I}_{p,q}}$ having nonempty intersections with $O_{p,q}^{=}$.

The row-wise ordering of the variables is natural choice. It moreover turned out from our computational experiments with graph partitioning problems that we could not find any alternative ordering yielding better results for isomorphism pruning, at least for this application. This even holds true for all variants we tested without a global rank vector (see Section 7).

If branch-and-bound trees for both methods are obtained by minimum index branching, orbitopal fixing can be well compared to isomorphism pruning. Indeed (provided that the nodes are also processed in the same order and no cutting planes are added), orbitopal fixing will visit only branch-and-bound nodes that isomorphism pruning visits as well. In any of these nodes the orbitopal fixing algorithm does not perform 1-fixings in loop 13 in Algorithm 1. However, it may do more zero-fixings in loop 2 than isomorphism pruning. For an example see Fig. 5. Thus, the main advantage of orbitopal fixing for our special case of symmetry can be seen as deriving as early as possible conclusions that hold at every child node.

If branching rules different from minimum index branching are applied, the trees produced by using orbitopal fixing and isomorphism pruning are not comparable, in the sense that, in general, both variants will visit branch-and-bound nodes (partial solutions) not visited by the other one. But even at nodes that appear in both trees the behavior of the two methods cannot really be compared unless both use the same representatives. This, however, is only the case if the isomorphism pruning variant does minimum index branching, yielding (locally) the canonical rank vector. Hence, we are in the situation already discussed above (no matter by which branching rule the orbitopal fixing variant has arrived at the node).

Another approach for avoiding symmetrical solutions is orbital branching [24]. It handles the same type of symmetry as isomorphism pruning, but in a local manner: the symmetry group of the current LP is computed on-the-fly for every node of the branch-and-bound tree after removing fixed variables and inequalities that are satisfied regardless of the unfixed variables. The set of unfixed variables then decomposes into orbits of equivalent variables under the symmetry group of the current LP. Then for some orbit of equivalent variables a two-way branching is done, where in one branch the case is considered that all variables in the orbit are zero, and in the other branch a chosen variable from the orbit is fixed to one. Orbital branching is thus not comparable to either orbitopal fixing nor isomorphism pruning, as it can consider symmetries that only arise after fixing variables. On the other hand a problem usually looses global symmetry when variables are removed. As the authors of [24] already point out, it is quite time consuming to compute the symmetry group for every branch-and-bound node. In [32] the authors report that a variant of orbital branching exploiting global symmetry is computationally superior, i.e., instead of computing the symmetry group of every local LP in the branch-and-bound tree, the global symmetry group of the root LP is used. In this variant, the symmetry group used in a branch-and-bound node is the subgroup of the global symmetry group that setwise stabilizes the variables already fixed to one. This, however, is very similar to isomorphism pruning. If there is some orbit O of equivalent variables at some node N, then orbital branching on O is the same as pruning by isomorphism and 0-setting *relative to the branching decisions leading to N*. In this sense orbital branching with global symmetry can be seen as isomorphism pruning with local rank vectors.

In the case of partitioning orbitopes, the symmetry has very simple structure. Note that any symmetry considered here stems from the permutation of columns of a matrix variable, which is a reasonable restriction in particular with regard to the example application presented in our graph partitioning formulation (1). At every branch-and-bound node, the columns of the matrix variable x decompose into one set of columns that are fixed elementwise, and one set of columns which still can be permuted arbitrarily.

The special structure of partitioning problems considered here implies that the orbits described above are the same in both variants of orbital branching, at least as long as symmetry among rows of the matrix variable (graph automorphisms in an instance of graph partitioning) are ignored.

The main advantage of orbital branching is its flexibility, e.g., orbital branching can be used at any node regardless how branching was performed on other nodes of the branch-and-bound tree.

7. Computational experiments

We performed computational experiments for the graph partitioning problem mentioned in the introduction. The code is based on the SCIP 1.2.0 branch-and-cut framework [33], originally developed by Achterberg [19]. We use CPLEX 11.00 as the underlying LP solver. The computations were performed on a 3.2 GHz Pentium 4 machine with 4 GB of main memory and 2 MB cache running Linux. All computation times are CPU seconds and are subject to a *time limit of four hours*. Since in this paper we are not interested in the performance of heuristics, we initialized all computations with the *optimal primal solution*.

We compare different variants of the code by counting *winning* instances. An instance is a winner for variant A compared to variant B, if A finished within the time limit and B did not finish or needed a larger CPU time; if A did not finish, then the instance is a winner for A in case that B did also not finish, leaving, however, a larger gap than A. If the difference between the times or gaps are below 1 s and 0.1%, respectively, the instance is not counted.

In all variants, we fix the variables x_{ij} with $j > i$ to zero. Furthermore, we heuristically separate general clique inequalities

$$\sum_{i,j \in C, i \neq j} y_{ij} \geq b,$$

where

$$b = \frac{1}{2}t(t-1)(q-r) + \frac{1}{2}t(t+1)r$$

and $C \subseteq V$ is a clique of size $tq + r > q$ with integers $t \geq 1,\ 0 \leq r < q$ (see [2]). The separation heuristic for a fractional point $y^\star$ follows ideas of Eisenblätter [1]. We generate the graph $G' = (V, E')$ with $\{i, k\} \in E'$ if and only if $\{i, k\} \in E$ and $y^\star_{ik} < b(b+1)/2$, where $y^\star$ is the y-part of an LP solution. We search for maximum cliques in G' with the specialized branch-and-bound method implemented in SCIP (with a node limit of 10 000), as well as with simple tabu search and greedy strategies. We then check whether the corresponding inequality is violated. We also separate triangle inequalities and both kinds of cycle inequalities as given in [2].

After extensive testing, we decided to branch by default on the *first index*, i.e., we branch on the first fractional x-variable in the row-wise variable order used for defining orbitopes. A side-effect of this choice is that this branching rule makes orbitopal fixing more comparable to isomorphism pruning, in particular, to the variant using an *a priori* fixed variable order. It should be noted that this branching rule is superior only when the vertices are ordered (i.e., the rows of x are permuted) as follows: sort the vertices in descending order of their star weight, i.e., the sum of the weights of incident edges.

We generated 36 random instances with $p = 40$ vertices and m edges of the following types. We used $m = 360$ (*sparse*), 540 (*medium*), and 720 (*dense*). For each type, we generated three instances by picking edges uniformly at random (without recourse) until the specified number of edges is reached. The edge weights are drawn independently and uniformly at random from [1000]. For each instance we computed results for $q = 3, 6, 9$, and 12.

In a first experiment we tested the speedup that can be obtained by performing orbitopal fixing. For this we compare the variant (*basic*) without symmetry breaking (except for the zero-fixing of the upper right x-variables) and the version in which we use orbitopal fixing (*OF*); see Table 1 for the results. Columns "nsub" give the number of nodes in the branch-and-bound tree and "#OF" the number of fixings within OF. The results show that orbitopal fixing is clearly superior (OF winners: 30, basic winners: 0), see also Fig. 6.

Table 1
Results of the branch-and-cut algorithm. All entries are rounded averages over three instances. CPU times are given in seconds.

n	m	q	basic		Iso pruning		OF		
			nsub	CPU	nsub	CPU	nsub	CPU	#OF
40	360	3	677	112	708	100	**516**	**86**	4
40	360	6	1 072	76	655	25	**157**	**15**	97
40	360	9	**1**	**0**	**1**	**0**	**1**	**0**	0
40	360	12	**1**	**0**	**1**	**0**	**1**	**0**	0
40	540	3	288	184	257	180	**219**	**146**	4
40	540	6	57 606	13 915	48 786	7024	**32 347**	**5548**	5 750
40	540	9	62 053	14 400	162 871	7182	**43 434**	**4768**	25 709
40	540	12	40 598	6 018	2 187	69	**174**	**31**	166
40	720	3	488	1 399	393	1325	**366**	**1080**	5
40	720	6	6 888	11 139	3 507	3957	**2 563**	**3263**	756
40	720	9	21 746	14 400	20 743	8220	**12 753**	**6820**	10 656
40	720	12	24 739	14 400	68 920	9164	**21 067**	**6209**	23 532

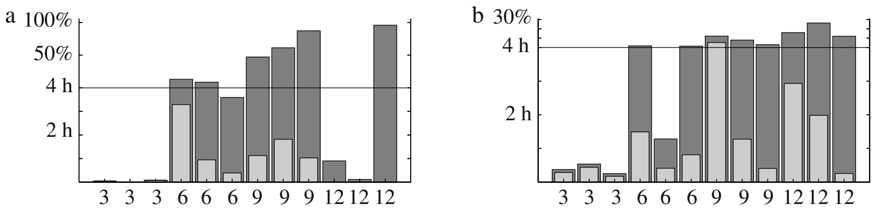


Fig. 6. Computation times/gaps for the basic version (dark gray) and the version with orbitopal fixing (light gray). Left: instances with $n = 40$, $m = 540$. Right: instances for $n = 40$, $m = 720$. The number of partitions q is indicated on the x-axis. Values above 4 h indicate the gap in percent.

Table 1 shows that the sparse instances are extremely easy, the instances with $m = 540$ are quite easy, while the dense instances are hard. A situation that often occurs for small m and large q is that the optimal solution is 0, and hence no work has to be done. For $m = 720$, the hardest instances arise when $q = 9$. It seems that for $q = 3$ the small number of variables helps, while for $q = 12$ the small objective function values help. Of course, symmetry breaking methods become more important when q gets larger.

In addition, we report results for instances used in [13] in Table 2. These instances arise from grid graphs. Due to their sparsity these instances are already approachable for the basic variant, but symmetry breaking yields huge performance gains here, too. As sparsity is particularly exploited in our formulation, the running times turn out to be much smaller than in the SDP based approach of [13].

To compare orbitopal fixing to the isomorphism pruning approach of Margot, we implemented the basic variant with fixed canonical variable order, the *ranked branching rule* (see [34]), as well as the variant with free branching decisions, each adapted to the special symmetry we exploit, which simplifies Margot's algorithm significantly. We decided to use the canonical order variant, as it gave the best results. Other than that, the same implementation and settings were used. It can be seen from Table 1 (columns *Iso Pruning*) that isomorphism pruning is inferior to both orbitopal fixing (OF winners: 30, isomorphism pruning winners: 0) and shifted column inequalities (30:0), but is still a big improvement over the basic variant (28:2). Table 2 yields a similar conclusion: orbitopal fixing outperforms the basic variant, as well as isomorphism pruning both in terms of CPU time (OF winners: 22, isomorphism pruning winners: 12, basic winners: 19), and in terms of branch-and-bound nodes (OF: isomorphism pruning: basic = 27: 9: 12). We additionally report the total values over all instances in Table 2. Note that there are many easy instances in this instance set. All in all, also in these experiments orbitopal fixing turns out to be superior and isomorphism pruning still shows advantages over the basic variant.

We did not implement orbital branching, since this method when using global symmetry only is very similar to isomorphism pruning in our context as pointed out in Section 6. It should, however, be noted that, in contrast to orbitopal fixing, both isomorphism pruning as well as orbital branching could exploit symmetries of the instance graphs, too. However, no nontrivial graph automorphisms in our test instances were found by nauty, see [35].

In a second experiment, we investigated the symmetry breaking capabilities built into CPLEX. We suspect that it breaks symmetry within the tree, but no detailed information was available. We first ran CPLEX 12.1 on the IP formulation stated in Section 1. In one variant, we fixed variables x_{ij} with $j > i$ to zero, but turned symmetry breaking off. In a second variant, we turned symmetry breaking on and did not fix variables to zero (otherwise CPLEX seems not to recognize the symmetry). The level was set to most aggressive (5), although the default setting yields the same results. The symmetry breaking variant turned out to be effective: it was always faster than the basic version without symmetry breaking. The black box use of CPLEX was always inferior to our code. However, of course this is partially due to our use of specialized cutting planes.

We then compared the built-in symmetry treatment of CPLEX to orbitopal fixing. We implemented orbitopal fixing as a branching callback in CPLEX that passed all fixings with the branching decision. In order to obtain a fair comparison, all

Table 2
Results of the branch-and-cut algorithm. CPU times are given in seconds.

graph	n	m	q	basic		Iso pruning		OF		
				CPU	nsub	CPU	nsub	CPU	nsub	#OF
data_2g_10_1001	100	200	3	**9**	**7**	11	17	**9**	15	1
data_2g_10_1001	100	200	5	84	674	**31**	**148**	48	346	98
data_2g_10_1001	100	200	7	71	89	**30**	233	52	**35**	42
data_2g_6_66	36	72	3	**1**	**11**	**1**	27	**1**	**11**	3
data_2g_6_66	36	72	5	5	483	5	667	**3**	**175**	67
data_2g_6_66	36	72	7	8	623	9	1 259	**5**	**226**	232
data_2g_7_1034	49	98	3	**1**	23	4	28	**1**	**13**	1
data_2g_7_1034	49	98	5	10	613	**8**	534	11	**491**	51
data_2g_7_1034	49	98	7	50	3 541	19	1 229	**11**	**359**	110
data_2g_8_648	64	128	3	**4**	21	5	23	5	**15**	2
data_2g_8_648	64	128	5	15	383	**7**	**10**	12	159	18
data_2g_8_648	64	128	7	**9**	95	13	**12**	72	1 879	530
data_2g_9_9211	81	162	3	**5**	**48**	22	77	11	61	1
data_2g_9_9211	81	162	5	13	**1**	**10**	**1**	15	**1**	0
data_2g_9_9211	81	162	7	**14**	**1**	17	**1**	15	**1**	0
data_3g_234_234	24	60	3	**0**	**5**	**0**	16	**0**	**5**	1
data_3g_234_234	24	60	5	**0**	**1**	**0**	**1**	**0**	**1**	0
data_3g_234_234	24	60	7	**0**	**1**	**0**	**1**	**0**	**1**	0
data_3g_244_244	32	80	3	**1**	61	**1**	100	**1**	**57**	2
data_3g_244_244	32	80	5	**0**	**1**	**0**	**1**	**0**	**1**	0
data_3g_244_244	32	80	7	**1**	**1**	**1**	**1**	**1**	**1**	0
data_3g_333_333	27	81	3	**0**	**5**	1	11	1	**5**	1
data_3g_333_333	27	81	5	**1**	71	46	129	**1**	**37**	19
data_3g_333_333	27	81	7	5	493	1984	463	**2**	**131**	136
data_3g_334_334	36	108	3	**1**	41	2	56	**1**	**29**	3
data_3g_334_334	36	108	5	**3**	139	9	352	**3**	**51**	10
data_3g_334_334	36	108	7	36	3 697	795	1 245	**14**	**527**	566
data_3g_344_344	48	144	3	**2**	47	5	64	**2**	**29**	3
data_3g_344_344	48	144	5	24	887	35	830	**17**	**300**	53
data_3g_344_344	48	144	7	182	7 747	494	2 889	**50**	**1 032**	556
data_3g_444_444	64	192	3	**5**	**37**	14	73	7	39	3
data_3g_444_444	64	192	5	274	5 491	184	3 695	**117**	**1 377**	272
data_3g_444_444	64	192	7	5503	114 011	528	13 938	**412**	**4 815**	2500
Total				6338	139 349	4292	28 131	898	12 225	

advanced features like preprocessing, primal heuristics, and cuts were turned off. Moreover, we did not provide an optimal solution. The branching was performed as a first-index branching along the x-variables. The results are shown in Table 3 for several different graph densities. The comparison is performed between CPLEX without symmetry handling (*CPLEX-basic*), with aggressive symmetry handling (*CPLEX-sym5*), and with orbitopal fixing (*CPLEX-OF*). For the reasons discussed above, we only fixed the upper triangle to zero in variants *CPLEX-basic* and *CPLEX-OF*. It can be seen that in these experiments orbitopal fixing performs slightly better than symmetry breaking in CPLEX. Note that the comparison is skewed in favor of CPLEX's own symmetry handling: orbitopal fixing relies on the ability to identify the fixed variables in every node of the branch-and-bound tree. However, apart from the branching decisions no fixings are reported by the CPLEX-API. In contrast, SCIP is much better suited for orbitopal fixing, since in SCIP the strengthening of variable bounds, called propagation, is an essential concept. Besides this, SCIP can also perform conflict analysis, which makes use of the information collected via propagation, see [36].

In another experiment, we turned off orbitopal fixing and separated shifted column inequalities in every node of the tree. The results on the original test set of random instances are that the OF-version is slightly better than the SCI variant (OF winners: 19, SCI winners: 11), but the results are quite close (OF average time: 2330 s, SCI average time: 2288 s). Although by Part 2 of Theorem 3, orbitopal fixing is as strong as fixing with SCIs (with the same branching decisions), the LPs get harder and the process slows down a bit. On the other hand, the SCIs are already active in the root node, which in general yields a better root bound. This may result in fewer branch-and-bound nodes due to potentially more fixings in the root node.

8. Concluding remarks

The main contribution of this paper is the development of an algorithm that handles orbitopal symmetry for binary programs with assignment structure by fixing values in partial solutions to exclude symmetric branches from exploration. The algorithm is proven to be optimal in the sense that as many such fixings, as early as possible are made. Moreover, it is shown that the algorithm can be implemented to run in linear time. The considered assignment structure occurs frequently in standard IP formulations.

Table 3
Comparison between the CPLEX base variant (CPLEX-basic), CPLEX symmetry breaking (CPLEX-sym5), and OF (CPLEX-OF), averaged over three instances.

n	*m*	*q*	CPLEX-basic		CPLEX-sym5		CPLEX-OF	
			CPU	nsub	CPU	nsub	CPU	nsub
24	190	2	0	837	**0**	**758**	0	774
24	190	3	5	7 534	4	5 824	4	**5 745**
24	190	4	13	23 445	8	13 274	**8**	**12 054**
24	190	5	12	20 284	6	9 211	**4**	**5 819**
24	190	6	7	9 749	5	6 610	**2**	**2 336**
24	250	2	**2**	**2 768**	2	2 806	3	3 056
24	250	3	42	51 085	**37**	**40 281**	47	42 385
24	250	4	263	341 511	**134**	**146 861**	139	156 559
24	250	5	540	749 898	397	472 858	**191**	**220 726**
24	250	6	927	1 224 157	226	265 439	**154**	**171 314**
30	200	2	1	**898**	**1**	990	1	953
30	200	3	5	7 134	**4**	**5 668**	5	5 899
30	200	4	4	6 450	3	3 697	3	**3 100**
30	200	5	2	1 981	**1**	750	1	**706**
30	200	6	0	359	0	180	0	**151**
30	233	2	1	1 732	**1**	**1 663**	2	1 823
30	233	3	15	18 346	**11**	**11 936**	15	13 293
30	233	4	20	29 774	14	17 736	**12**	**14 387**
30	233	5	26	34 350	11	13 466	**9**	**9 014**
30	233	6	7	7 295	8	7 180	**2**	**1 957**
30	266	2	**3**	3 177	3	**3 120**	3	3 317
30	266	3	49	46 779	**39**	**36 642**	46	36 840
30	266	4	119	144 671	90	94 227	**77**	**75 152**
30	266	5	181	224 103	110	120 573	**72**	**71 303**
30	266	6	263	267 457	101	97 577	**54**	**39 315**
30	300	2	**7**	**6 176**	7	6 495	8	7 026
30	300	3	204	186 858	**175**	**145 598**	222	157 670
30	300	4	738	766 173	427	396 841	**413**	**378 413**
30	300	5	1283	1 357 270	456	441 720	**408**	**364 906**
30	300	6	1002	913 069	362	326 654	**145**	**113 401**
Total			5741	6 455 320	2643	2 696 635	2051	1 919 394

The effectiveness of our approach is demonstrated by applying our algorithm to the graph partitioning problem, where it is compared with other known methods to handle symmetry.

In the future, extensions in several directions might be possible: other group actions, different restrictions on the number of 1's in each row, symmetries acting on both rows and columns. Moreover, more extensive studies for other applications are desirable to further explore the practical implications of our symmetry handling approach.

Acknowledgments

We are grateful to Andreas Loos for discussions on the complexity of optimizing linear functions over covering orbitopes that lead to the basic idea for the proof of Theorem 5. We also thank Bissan Ghaddar for providing us with the instances reported on in Table 2 and an anonymous referee for comments that helped to improve the computational results of this paper.

Part of the research was conducted while the first author was a visiting professor at TU Berlin.

References

[1] A. Eisenblätter, Frequency assignment in GSM networks: models, heuristics, and lower bounds, Ph.D. Thesis, TU Berlin, 2001.
[2] S. Chopra, M. Rao, The partition problem, Math. Program. 59 (1993) 87–115.
[3] S. Chopra, M. Rao, Facets of the *k*-partition polytope, Discrete Appl. Math. 61 (1995) 27–48.
[4] J. Falkner, F. Rendl, H. Wolkowicz, A computational study of graph partitioning, Math. Program. 66 (1994) 211–239.
[5] G. Kochenberger, F. Glover, B. Alidaee, H. Wang, Clustering of microarray data via clique partitioning, J. Comb. Optim. 10 (2005) 77–92.
[6] C. Ferreira, A. Martin, C. de Souza, R. Weismantel, L. Wolsey, Formulations and valid inequalities of the node capacitated graph partitioning problem, Math. Program. 74 (1996) 247–266.
[7] C. Ferreira, A. Martin, C. de Souza, R. Weismantel, L. Wolsey, The node capacitated graph partitioning problem: a computational study, Math. Program. 81 (1998) 229–256.
[8] A. Mehrotra, M.A. Trick, Cliques and clustering: a combinatorial approach, Oper. Res. Lett. 22 (1998) 1–12.
[9] M.M. Sørensen, Polyhedral computations for the simple graph partitioning problem, Working Paper L-2005-02, Århus School of Business, 2005.
[10] M. Grötschel, Y. Wakabayashi, A cutting plane algorithm for a clustering problem, Math. Program. 45 (1989) 59–96.
[11] M. Grötschel, Y. Wakabayashi, Facets of the clique partitioning polytope, Math. Program. 47 (1990) 367–387.
[12] A. Eisenblätter, The semidefinite relaxation of the *k*-partition polytope is strong, in: W.J. Cook, A.S. Schulz (Eds.), Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization, IPCO'02, in: Lecture Notes in Computer Science, vol. 2337, Springer-Verlag, Berlin, Heidelberg, 2002, pp. 273–290.

[13] B. Ghaddar, M. Anjos, F. Liers, A branch-and-cut algorithm based on semidefinite programming for the minimum *k*-partition problem, Ann. Oper. Res. 188 (1) (2011) 155–174.
[14] V. Kaibel, M.E. Pfetsch, Packing and partitioning orbitopes, Math. Program. 114 (2008) 1–36.
[15] Y. Faenza, V. Kaibel, Extended formulations for packing and partitioning orbitopes, Math. Oper. Res. 34 (2009) 686–697.
[16] K.R. Apt, Principles of Constraint Programming, Cambridge University Press, 2003.
[17] P.V. Hentenryck, Constraint Satisfaction in Logic Programming, MIT Press, Cambridge, 1989.
[18] K. Marriott, P.J. Stuckey, Programming with Constraints: An Introduction, MIT Press, Cambridge, 1998.
[19] T. Achterberg, Constraint integer programming, Ph.D. Thesis, TU Berlin, 2007.
[20] V. Kaibel, M. Peinhardt, M.E. Pfetsch, Orbitopal fixing, in: M. Fischetti, D.P. Williamson (Eds.), Integer Programming and Combinatorial Optimization, 12th International IPCO Conference, Ithaca, NY, USA, June 25–27, 2007, Proceedings, in: Lecture Notes in Computer Science, vol. 4513, Springer, 2007, pp. 74–88.
[21] F. Margot, Pruning by isomorphism in branch-and-cut, Math. Program. 94 (2002) 71–90.
[22] F. Margot, Exploiting orbits in symmetric ILP, Math. Program. 98 (2003) 3–21.
[23] F. Margot, Symmetric ILP: coloring and small integers, Discrete Optim. 4 (2007) 40–62.
[24] J. Ostrowski, J. Linderoth, F. Rossi, S. Smriglio, Orbital branching, in: M. Fischetti, D. Williamson (Eds.), Proceedings of IPCO XII, in: LNCS, vol. 4513, Springer-Verlag, 2007, pp. 106–120.
[25] J. Ostrowski, J. Linderoth, F. Rossi, S. Smriglio, Constraint orbital branching, in: A. Lodi, A. Panconesi, G. Rinaldi (Eds.), Integer Programming and Combinatorial Optimization, Proc. of the 13th IPCO Conference, Bertinoro, in: Lecture Notes in Computer Science, vol. 5035, Springer, 2008, pp. 225–239.
[26] E.J. Friedman, Fundamental domains for integer programs with symmetries, in: Y. Zu, B. Zhu, A. Dress (Eds.), COCOA 2007, in: LNCS, vol. 4616, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 146–153.
[27] T. Fahle, S. Schamberger, M. Sellmann, Symmetry breaking, in: T. Walsh (Ed.), Principles and Practice of Constraint Programming – CP 2001: 7th International Conference, in: LNCS, vol. 2239, Springer-Verlag, Berlin, Heidelberg, 2001, pp. 93–107.
[28] J.-F. Puget, Symmetry breaking revisited, Constraints 10 (2005) 23–46.
[29] M. Sellmann, P.V. Hentenryck, Structural symmetry breaking, in: Proccedings of the 19th International Joint Conference on Artificial Intelligence, IJCAI, 2005, pp. 298–303.
[30] F. Margot, Symmetry in integer linear programming, in: M. Jünger, T. Liebling, D. Naddef, G.L. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, L. Wolsey (Eds.), 50 Years of Integer Programming 1958–2008, Berlin, Heidelberg, Springer-Verlag, 2010, pp. 647–681 (Chapter 17).
[31] J. Ostrowski, Solving integer programs with large degrees of symmetry, Ph.D. Thesis, Lehigh University, 2009.
[32] J. Ostrowski, J. Linderoth, F. Rossi, S. Smriglio, Orbital branching, Math. Program. 126 (2011) 147–168.
[33] SCIP, Solving constraint integer programs. http://scip.zib.de/.
[34] F. Margot, Small covering designs by branch-and-cut, Math. Program. 94 (2003) 207–220.
[35] B.D. McKay, Practical graph isomorphism, Congr. Numer. 30 (1981) 45–87.
[36] T. Achterberg, Conflict analysis in mixed integer programming, Discrete Optim. 4 (2007) 4–20.