Handling Symmetries in Mixed-Integer Semidefinite Programs

Christopher Hojny[1] and Marc E. Pfetsch[2]

[1]*Eindhoven University of Technology, Department of Mathematics and Computer Science, P.O. Box 513, 5600MB Eindhoven,, the Netherlands, email c.hojny@tue.nl*
[2]*TU Darmstadt, Department of Mathematics, Dolivostr. 15, 64293 Darmstadt, Germany email pfetsch@mathematik.tu-darmstadt.de*

December 8, 2022

Abstract

Symmetry handling is a key technique for reducing the running time of branch-and-bound methods for solving mixed-integer linear programs. In this paper, we generalize the notion of (permutation) symmetries to mixed-integer semidefinite programs (MISDPs). We first discuss how symmetries of MISDPs can be automatically detected by finding automorphisms of a suitably colored auxiliary graph. Then known symmetry handling techniques can be applied. We demonstrate the effect of symmetry handling on different types of MISDPs. To this end, our symmetry detection routine is implemented in the state-of-the-art MISDP solver SCIP-SDP. We obtain speed-ups similar to the mixed-integer linear case.

1 Introduction

In this paper, we consider solving general Mixed-Integer Semidefinite Programs (MISDP) of the following form:

$$\begin{aligned} \inf \quad & b^\top y \\ \text{s.t.} \quad & \sum_{k=1}^{m} A^k y_k - A^0 \succeq 0, \\ & \ell_i \leq y_i \leq u_i && \forall\, i \in [m], \\ & y_i \in \mathbb{Z} && \forall\, i \in I, \end{aligned} \tag{1}$$

with symmetric matrices $A^k \in \mathbb{R}^{n\times n}$ for $k \in [m]_0 \coloneqq \{0,\dots,m\}$, $b \in \mathbb{R}^m$, $\ell_i \in \mathbb{R}\cup\{-\infty\}$, $u_i \in \mathbb{R}\cup\{\infty\}$ for all $i \in [m] \coloneqq \{1,\dots,m\}$. The set of indices

of integer variables is given by $I \subseteq [m]$. The notation $M \succeq 0$ indicates that a matrix M is positive semidefinite. Throughout this paper, we use the notation $A(y) \coloneqq \sum_{k=1}^{m} A^k y_k - A^0$ for $y \in \mathbb{R}^m$.

One way to solve (1) is by SDP-based branch-and-bound, a special case of nonlinear branch-and-bound, see Dakin [5]. Here, branching on the integer variables creates a search tree and in each node a semidefinite program (SDP) is solved, which arises from the relaxation of the integrality requirements of that node. For more details on this approach see, e.g., [9, 20].

Optimization problems (1) are quite general and, in particular, contain mixed-integer linear programs (MIPs) as a special case, where one uses only the diagonal entries of the matrices A^k, $k \in [m]_0$. MISDPs also have numerous applications, e.g., robust truss topology optimization with discrete bar diameters [19] and cardinality least squares [24, 8].

The challenges for solving MIPs are inherited for the solution of MISDPs. This includes the presence of symmetries, which are defined as follows. Let

$$X = \{y \in \mathbb{R}^m \,:\, A(y) \succeq 0,\ \ell \leq y \leq u,\ y_i \in \mathbb{Z}\ \forall\, i \in I\}$$

be the feasible region of (1). A *symmetry* of (1) is a bijection $\pi \colon \mathbb{R}^m \to \mathbb{R}^m$ such that $X = \pi(X) \coloneqq \{\pi(x) \,:\, x \in X\}$ and $b^\top \pi(x) = b^\top x$ for every $x \in X$. Thus, π maps feasible solutions of (1) to feasible solutions with the same objective value. The symmetries of (1) form the so-called *symmetry group*.

The presence of symmetries results in an unnecessarily large search tree, since many symmetric solutions have to be treated although they do not contain new information. This effect is well-known for MIPs, and many different techniques for handling symmetries in MIPs have been developed, see, e.g., [18, 23, 12] for an overview. Many of these methods are implemented in the solver SCIP [10, 2].

As far as we know, symmetry handling for mixed-integer semidefinite programming has not been considered so far. For SDPs without integer variables, however, model reformulating techniques can be used to handle symmetries. The main idea is to aggregate a set of symmetric variables to a common variable that represents the average value of all symmetric variables, see, e.g., [11, 13, 1, 6]. Moreover, symmetries in mixed-integer conic programming has been investigated very recently in [26]; note that the SDP cone is not mentioned there.

The goal of this paper is to generalize the techniques for MIPs to MISDPs and to investigate their computational impact. We first discuss how symmetries can be computed. In Section 2 permutations of the variables and so-called formulation symmetries of MISDPs are defined. In Section 3, we discuss how such symmetries can be computed through graph automorphisms. We conclude the paper in Section 4 with a numerical study of the impact of handling symmetries in MISDPs.

2 Computing Symmetries

Note that the definition of a symmetry above is based on the feasible region X, which is hard to handle in general (e.g., it is NP-hard to decide if $X = \varnothing$). In

practice, one therefore often only considers permutations of variables that leave the description of X invariant. In the following, we generalize the corresponding definition for MIPs, see, e.g., Margot [18], to MISDPs.

Denote the (full) *symmetric group*, i.e., the set of all permutations of $[m]$, by $\mathcal{S}_m$. Then a permutation $\pi \in \mathcal{S}_m$ acts on $x \in \mathbb{R}^m$ by permuting its components, i.e., $\pi(x)_i = x_{\pi^{-1}(i)}$ for all $i \in [m]$. Thus, $\pi(x) \coloneqq (x_{\pi^{-1}(1)}, \dots, x_{\pi^{-1}(m)})^\top$. Let $\sigma \in \mathcal{S}_n$ act on a matrix $A \in \mathbb{R}^{n \times n}$ as follows:

$$\sigma(A)_{ij} = A_{\sigma^{-1}(i),\sigma^{-1}(j)} \quad \forall i, j \in [n].$$

Definition 1. *A permutation $\pi \in \mathcal{S}_m$ of variables is a* formulation symmetry *of (1) if there exists a permutation $\sigma \in \mathcal{S}_n$ such that*

(P1) $\pi(I) = I$, $\pi(\ell) = \ell$, $\pi(u) = u$, *and* $\pi(b) = b$ *(i.e., π leaves integer variables, variable bounds, and the objective coefficients invariant),*

(P2) $\sigma(A^0) = A^0$ *and, for all* $i \in [m]$, $\sigma(A^i) = A^{\pi^{-1}(i)}$.

Thus, the variables are permuted by π and the matrices by σ.

Lemma 2. *Every formulation symmetry of (1) is a symmetry.*

Proof. Let $y \in \mathbb{R}^m$ and let $\pi \in \mathcal{S}_m$ be a formulation symmetry with corresponding matrix permutation $\sigma \in \mathcal{S}_n$. Since $\pi(b) = b$ and permutations are orthogonal maps, we find $b^\top \pi(y) = \pi^{-1}(b)^\top y = b^\top y$. Thus, π leaves the objective invariant. It remains to show that π also maps feasible solutions onto feasible solutions.

Note that

$$\begin{aligned}\sigma(A)(\pi(y)) = \sum_{k=1}^{m} \sigma(A^k)\pi(y)_k - \sigma(A^0) &= \sum_{k=1}^{m} A^{\pi^{-1}(k)} y_{\pi^{-1}(k)} - A^0 \\ &\overset{(\star)}{=} \sum_{k'=1}^{m} A^{k'} y_{k'} - A^0 = A(y),\end{aligned}$$

where $(\star)$ follows from π being a permutation of $[m]$. Consequently, since formulation symmetry π maps integer variables onto integer variables and respects the bounds of variables, y is feasible for (1) if and only if $\pi(y)$ is feasible. □

We note that we currently only treat symmetries in the above sense, but do not reduce symmetries in the SDP-formulations as described in the introduction, see, e.g., [13, 1, 6] for details; we leave this to future research.

3 Symmetry Detection

A common strategy for detecting formulation symmetries of MIPs is to construct a suitably colored graph whose color-preserving automorphisms correspond to

formulation symmetries, see, e.g., [23, 25]. We follow this line of research and present a colored graph to detect formulation symmetries of MISDPs.

Recall that each formulation symmetry π admits a permutation $\sigma \in \mathcal{S}_n$ with $\sigma(A^i) = A^{\pi^{-1}(i)}$ for all $i \in [m]$ as well as $\sigma(A^0) = A^0$. To model this matrix invariant, we first introduce some notation. For each matrix A^k, $k \in [m]_0$, let $N_k = \{(i,j)^k \in [n] \times [n] \,:\, A^k_{ij} \neq 0\}$ be the set of its non-zero entries, where the superscript at $(i,j)^k$ is used to distinguish non-zero entries of different matrices. For $p = (i,j)^k \in N_k$, let $r^k(p) = i$ be its row index and $c^k(p) = j$ be its column index. We define the symmetry detection graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as described next.

The graph needs to capture both the permutation of variables $\pi \in \mathcal{S}_m$ and the permutation of the matrix entries $\sigma \in \mathcal{S}_n$. For this reason, we consider the node set $\mathcal{V} = V \cup D \cup \bigcup_{k=0}^m N_k$, where $V \coloneqq \{y_1, \dots, y_m\}$ and $D \coloneqq [n]$ represent the variables and the "dimensions" of the matrices of the MISDP, respectively. Permutation π will then correspond to a permutation of the variable nodes V and σ will correspond to a permutation of the dimension nodes in D. The nodes in $N_0, \dots, N_k$ will make sure that both π and σ are correctly linked, which is achieved by adding appropriate edges.

The edge set $\mathcal{E}$ is partitioned into $E_V \cup E_R \cup E_C \cup E_P$, where

$$
\begin{aligned}
E_V &= \{\{y_k, p\} \,:\, p \in N_k,\ k \in [m]\}\,, \\
E_R &= \big\{\{p, r^k(p)\} \,:\, p \in N_k,\ k \in [m]_0\big\}\,, \\
E_C &= \big\{\{p, c^k(p)\} \,:\, p \in N_k,\ k \in [m]_0\big\}\,, \\
E_P &= \big\{\{(i,j)^k, (j,i)^k\} \,:\, (i,j)^k \in N_k,\ k \in [m]_0\big\}\,.
\end{aligned}
$$

The edges in E_P are not necessary to encode formulation symmetries, however, we think that graph automorphism codes benefit from them since they allow to more easily recognize dependencies between different nodes. With this graph, permuting dimension nodes requires to also permute nodes corresponding to non-zero entries, which in turn might trigger a permutation of variable nodes and vice versa. To make sure that only nodes corresponding to the same problem information are mapped onto each other, we color the nodes and edges.

The variables are partitioned into groups, each with the same objective coefficient, lower & upper bound, and variable type (continuous or integer). Each of the groups is assigned a unique color and all variables within this group receive this color. Similarly, we define a unique color for the sets D and $\bigcup_{k=0}^m N_k$, and assign all nodes within such a set the corresponding color. That is, all nodes modeling a dimension of a matrix receive the same color and all nodes corresponding to matrix entries receive the same color. Finally, to also distinguish the entries of the matrices $A^0, \dots, A^m$, we color the edges in E_R according to their matrix coefficient, i.e., edge $\{p, r^k(p)\}$ gets color A^k_p. Similarly, we color the edges in E_C. To distinguish "row colors" from "column colors", we refer to the color of $\{p, c^k(p)\}$ as $\bar{A}^k_p$. All other edges remain uncolored.

A bijective map $\varphi\colon \mathcal{V} \to \mathcal{V}$ is an *automorphism* of $\mathcal{G}$ if it preserves adjacency, i.e., $\{u, v\} \in \mathcal{E}$ if and only if $\{\varphi(u), \varphi(v)\} \in \mathcal{E}$. We say that φ is *color-preserving*

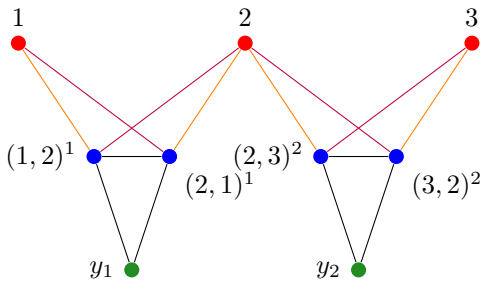

Figure 1: Illustration of symmetry detection graph.

if, for every $v \in \mathcal{V}$, nodes $\varphi(v)$ and v have the same color and, for each $\{u, v\} \in \mathcal{E}$, edges $\{u, v\}$ and $\{\varphi(u), \varphi(v)\}$ have the same color.

Example 3. *Consider the MISDP*

$$\inf \left\{ y_1 + y_2 \,:\, \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} y_1 + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} y_2 \succeq 0,\ 0 \leq y_1,\, y_2 \leq 1,\ y_1,\, y_2 \in \mathbb{Z} \right\}.$$

The corresponding symmetry detection graph is given in Fig. 1, where uncolored edges are drawn in black and $(i,j)^k$ denotes entry (i,j) of A^k. The only non-trivial color-preserving automorphism of the graph exchanges $y_1 \leftrightarrow y_2$, $(1,2)^1 \leftrightarrow (3,2)^2$, $(2,1)^1 \leftrightarrow (2,3)^2$, $1 \leftrightarrow 3$, and keeps node 2 fixed. This leads to the variable permutation π, which exchanges y_1 and y_2, and the matrix permutation σ, which exchanges 1 and 3.

We show that $\mathcal{G}$ captures all information about formulation symmetries. The *restriction* of φ to a set $B \subseteq \mathcal{V}$ is denoted $\varphi|_B$.

Proposition 4. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be constructed as described above.*

- *For each color-preserving automorphism φ of $\mathcal{G}$, $\varphi|_V$ is a formulation symmetry of (1).*
- *For every formulation symmetry π of (1), there is a color-preserving automorphism φ such that $\varphi|_V = \pi$.*

Proof. For the first part, let φ be a color-preserving automorphism of $\mathcal{G}$. Define $\pi \coloneqq \varphi|_V$ and $\sigma \coloneqq \varphi|_D$. We claim that π is a formulation symmetry with corresponding matrix permutation σ. Since φ is a color-preserving automorphism, the image of π is V and the image of σ is D. Moreover, by the choice of colors, π can only map variables of the same type (integer/continuous, objective coefficient, upper and lower bounds) onto each other. Thus, π satisfies (P1). Moreover, for all $k \in [m]$ and $(i,j)^k \in N_k$, we have $\varphi((i,j)^k) = (\sigma(i), \sigma(j))^{\pi(k)}$ because φ preserves adjacency:

- If one element of N_k is mapped to $N_{k'}$, all elements from N_k need to be mapped to $N_{k'}$ by the edges in E_V; in particular, $k' = \pi(k)$.

- If $(i,j)^k$ is mapped to $(i',j')^{k'}$, the edges in E_R (resp. E_C) ensure $i' = \sigma(i)$ (resp. $j' = \sigma(j)$); in particular, $A^k_{i,j} = A^{k'}_{i',j'}$ as φ preserves edge colors.

Since $\varphi((i,j)^k) = (\sigma(i), \sigma(j)^{\pi(k)})$ describes the action of σ on matrix A^k, the second part of (P2) holds. The first part $\sigma(A^0) = A^0$ follows by the same argument, because φ cannot map $(i,j)^0$ to a node $(i',j')^k$ for $k \neq 0$ since the nodes in N_0 are the only matrix-entry nodes not connected to a variable node.

The second part follows from the above discussion by setting

$$\varphi(v) = \begin{cases} \pi(v), & \text{if } v \in V, \\ \sigma(v), & \text{if } v \in D, \\ (\sigma(i), \sigma(j))^{\pi(k)}, & \text{if } v = (i,j)^k \in N_k \text{ for some } k \in [m], \\ (\sigma(i), \sigma(j))^0, & \text{if } v = (i,j)^0 \in N_0 \text{ for some } k \in [m], \end{cases}$$

for each $v \in \mathcal{V}$. □

Remark 5. *Graph automorphism codes like* `bliss` *[15] or* `nauty` *[21] cannot handle edge colors. Modifying $\mathcal{G}$ by replacing colored edges $\{u,v\}$ by a node with the same color that is connected to u and v allows to use these codes.*

4 Computational Results

We implemented the symmetry detection method described in Section 3 in SCIP-SDP 4.1.0. SCIP-SDP is a framework for solving MISDPs and is available at https://wwwopt.mathematik.tu-darmstadt.de/scipsdp/. We compiled SCIP-SDP with a developer version of SCIP 8.0.2 (githash `878b1c5`) and used Mosek 9.2.40 for solving SDP-relaxations. All tests were performed on a Linux cluster with 3.5 GHz Intel Xeon E5-1620 Quad-Core CPUs, having 32 GB main memory and 10 MB cache. All computations were run single-threaded and with a time limit of one hour.

To handle symmetries, we use the variant of the state-of-the-art method orbital fixing [17, 22] as described in [23] and as implemented in SCIP. The idea of orbital fixing is to derive symmetry-based fixings of binary variables that are derived from the branching decisions and already fixed variables.

In the first experiment, we use the same 185 instances as in [20] from a variety of applications; the instances can be downloaded from the second author's web page. Table 1 shows the detected symmetries of all 21 instances that contain symmetry. We can see that most instances admit symmetries of the full symmetric group $\mathcal{S}_k$, i.e., k variables can be permuted arbitrarily. As $|\mathcal{S}_k| = k!$, the symmetry groups become rather large, which indicates that symmetry handling might be beneficial for solving these instances. Note that the action of the symmetries might be nontrivial, i.e., several variables might be moved simultaneously like permuting the rows or columns of a matrix.

Table 2 shows a comparison of the default without symmetry handling and the new version with symmetry handling. Here, we excluded one numerically difficult instance for which both variants computed a wrong solution. The columns

Table 1: Symmetries in the 21 symmetric instances from [20], where $\mathcal{S}_k$ refers to the full symmetric group on k elements and $\mathcal{D}_k$ is the dihedral group.

instance	symmetry group
0+-115305C_MISDPld000010	$\mathcal{S}_2$
0+-115305C_MISDPrd000010	$\mathcal{S}_2$
band60605D_MISDPld000010	$\mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_{10} \times \mathcal{S}_3 \times \mathcal{S}_4$
band60605D_MISDPrd000010	$\mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_{10} \times \mathcal{S}_3 \times \mathcal{S}_4$
band70704A_MISDPld000010	$\mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_3 \times \mathcal{S}_3$
band70704A_MISDPrd000010	$\mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_3 \times \mathcal{S}_3$
clique_60_k10_6_6, clique_60_k15_4_4, clique_60_k20_3_3, clique_60_k4_15_15, clique_60_k5_12_12, clique_60_k6_10_10, clique_60_k7_8_9, clique_60_k8_7_8, clique_60_k9_6_7, clique_70_k3_23_24	$\mathcal{S}_2$
diw_34	$\mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{D}_4 \times \mathcal{S}_4 \times \mathcal{S}_4$
diw_37	$\mathcal{S}_2 \times \mathcal{S}_4 \times \mathcal{S}_3 \times \mathcal{S}_4$
diw_38	$\mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_2 \times \mathcal{S}_3$
diw_43	$\mathcal{S}_3$
diw_44	$\mathcal{S}_3$

Table 2: Results for MISDP instances from [20].

	all (184)			all optimal (168)		only symmetric (21)
variant	time (s)	symtime (s)	# gens	time (s)	#nodes	time (s)
no symmetry	130.6	–	–	95.0	778.3	45.07
orbital fixing	125.3	0.44	99	90.8	760.6	29.84

represent the shifted geometric mean of the CPU time in seconds (with a shift of 1 s), the average time for symmetry handling (including detection), and the number of generators. The next two columns display the shifted geometric mean of the CPU time and number of nodes (with a shift of 100) for the 168 instances that could be solved by both variants. The last column shows the CPU time, but only for those instances for which at least one generator has been found.

One can see a 4 % speed-up in CPU time for all instances and about 34 % for the 21 instances that contain symmetry. Note that symmetry handling does not help to solve more instances (168), but to speed up computation. The time for handling and computing symmetry over all instances is quite small. Similarly, the number 99 of found generators is quite small. This fits well with Table 1 in which the symmetry groups admit a small set of generators.

In a second experiment, we consider the problem of finding a maximum stable set in an unweighted undirected graph $G = (V, E)$. Based on [16, 3], we

Table 3: Results for stable set MISDP instances on Color02 graphs.

	all (54)			all optimal (47)		only symmetric (51)
variant	time (s)	symtime (s)	# gens	time (s)	#nodes	time (s)
no symmetry	89.2	–	–	51.1	11.0	85.35
orbital fixing	81.4	0.32	1028	46.4	7.2	77.54

Table 4: Results for the stable set MISDP instances on flower snark graphs.

	all (20)			all optimal (14)	
variant	time (s)	symtime (s)	# gens	time (s)	#nodes
no symmetry	310.8	–	–	108.3	10.2
orbital fixing	211.7	0.23	80	67.3	7.2

derive the MISDP formulation

$$
\begin{aligned}
\sup \quad & \sum_{v \in V} x_v \\
\text{s.t.} \quad & \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0, \\
& X_{uv} \le x_u,\ X_{uv} \le x_v,\ x_u + x_v \le 1 + X_{uv} \quad \forall \{u,v\} \in \binom{V}{2}, \\
& X_{uv} = 0, \qquad \forall \{u,v\} \in E, \\
& x \in \{0,1\}^V,\ X \in \{0,1\}^{V \times V}.
\end{aligned}
\tag{2}
$$

Model (2) can be easily transformed into a MISDP of type (1). It indeed models the stable set problem, where $x_v = 1$ if and only if v is contained in a stable set. The linear constraints model that $X_{uv} = x_u \cdot x_v$ and that not both endpoints of an edge can be contained in a stable set. The SDP constraint arises from the observation that $xx^\top \succeq 0$ for every incidence vector x of a stable set.

We conducted experiments for two sets of graph instances. The `Color02` test sets consists of the 55 smallest graphs from the Color02 symposium [4]; the `Snark` test sets consists of so-called flower snark graphs [14, 7] with $4n$ nodes, where $n \in \{11, \ldots, 49\} \cap (1 + 2\mathbb{Z})$ nodes. The latter instances are of particular interest, because they admit a large dihedral symmetry group.

Table 3 provides results for the `Color02` graphs. Here we excluded one instance for which the computations produced a wrong result. There is a speed-up of about 9 % each for all instances, the ones the have been solved by all variants, and the ones in which a least one symmetry has been found. Note that the symmetry of the formulation only arises from the symmetry of the graph.

Table 4 shows the result for the flower snark graphs, which all contain symmetries. With symmetry handling one solves 17 instances, compared to 14 for the default. Moreover, there is a speed-up of about 32 % and 38 % on all instances and the ones solved to optimality by both variants, respectively.

Conclusion Our numerical experiments indicate that symmetry handling is an important tool for reducing the running time of solving MISDPs. Our graph automorphism based approach allows to quickly compute symmetries such that the additional time needed to detect symmetries is negligible. Then, using state-of-the-art methods such as orbital fixing, already allows to substantially improve the solution time on a variety of test sets. An interesting question is whether new symmetry handling methods that are tailored for MISDPs allow to improve the running time even further. For instance, one could try to combine handling permutation symmetries and using model reformulations as mentioned in the introduction. We leave this for future research.

References

[1] Bai, Y., de Klerk, E., Pasechnik, D., Sotirov, R.: Exploiting group symmetry in truss topology optimization. Optimization and Engineering **10**(3), 331–349 (2009)

[2] Bestuzheva, K., Besançon, M., Chen, W.K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., Gottwald, L., Graczyk, C., Halbig, K., Hoen, A., Hojny, C., van der Hulst, R., Koch, T., Lübbecke, M., Maher, S.J., Matter, F., Mühmer, E., Müller, B., Pfetsch, M.E., Rehfeldt, D., Schlein, S., Schlösser, F., Serrano, F., Shinano, Y., Sofranac, B., Turner, M., Vigerske, S., Wegscheider, F., Wellner, P., Weninger, D., Witzig, J.: The SCIP Optimization Suite 8.0. Technical report, Optimization Online (2021), http://www.optimization-online.org/DB_HTML/2021/12/8728.html

[3] Burer, S., Monteiro, R.D., Zhang, Y.: Maximum stable set formulationsand heuristics based on continuous optimization. Mathematical Programming **94**, 137–166 (2022). https://doi.org/10.1007/s10107-002-0356-4

[4] Color02 – computational symposium: Graph coloring and its generalizations. Available at http://mat.gsia.cmu.edu/COLOR02 (2002)

[5] Dakin, R.J.: A tree-search algorithm for mixed integer programming problems. Comput. J. **8**(3), 250–255 (1965). https://doi.org/10.1093/comjnl/8.3.250

[6] de Klerk, E., Sotirov, R.: Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem. Mathematical Programming **122**(2), 225–246 (2010)

[7] Fiorini, S., Wilson, R.J.: Edge-colourings of graphs. No. 16 in Research Notes in Mathematics, Pitman Publishing Limited (1977)

[8] Gally, T.: Computational Mixed-Integer Semidefinite Programming. Dissertation, TU Darmstadt (2019)

[9] Gally, T., Pfetsch, M.E., Ulbrich, S.: A framework for solving mixed-integer semidefinite programs. Optimization Methods and Software **33**(3), 594–632 (2017). https://doi.org/10.1080/10556788.2017.1322081

[10] Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., Hendel, G., Hojny, C., Koch, T., Bodic, P.L., Maher, S.J., Matter, F., Miltenberger, M., Mühmer, E., Müller, B., Pfetsch, M.E., Schlösser, F., Serrano, F., Shinano, Y., Tawfik, C., Vigerske, S., Wegscheider, F., Weninger, D., Witzig, J.: The SCIP Optimization Suite 7.0. Tech. rep., Optimization Online (2020), http://www.optimization-online.org/DB_HTML/2020/03/7705.html

[11] Gatermann, K., Parrilo, P.: Symmetry groups, semidefinite programs, and sums of squares. Journal of Pure and Appl. Algebra **192**(1-3), 95–128 (2004)

[12] Hojny, C., Pfetsch, M.E.: Polytopes associated with symmetry handling. Mathematical Programming **175**(1), 197–240 (2019). https://doi.org/10.1007/s10107-018-1239-7

[13] Hu, H., Sotirov, R., Wolkowicz, H.: Facial reduction for symmetry reduced semidefinite and doubly nonnegative programs. Mathematical Programming (2022), to appear

[14] Isaacs, R.: Infinite families of nontrivial trivalent graphs which are not tait colorable. The American Mathematical Monthly **82**(3), 221–239 (1975). https://doi.org/10.1080/00029890.1975.11993805, https://doi.org/10.1080/00029890.1975.11993805

[15] Junttila, T., Kaski, P.: bliss: A tool for computing automorphism groups and canonical labelings of graphs, https://users.aalto.fi/~tjunttil/bliss/

[16] Lovász, L.: On the Shannon capacity of a graph. IEEE Transactions on Information Theory **25**, 1–7 (1979)

[17] Margot, F.: Exploiting orbits in symmetric ILP. Mathematical Programming **98**(1–3), 3–21 (2003). https://doi.org/10.1007/s10107-003-0394-6, http://dx.doi.org/10.1007/s10107-003-0394-6

[18] Margot, F.: Symmetry in integer linear programming. In: Jünger, M., Liebling, T., Naddef, D., Nemhauser, G.L., Pulleyblank, W., Reinelt, G., Rinaldi, G., Wolsey, L. (eds.) 50 Years of Integer Programming 1958–2008, chap. 17, pp. 647–681. Springer-Verlag, Berlin Heidelberg (2010)

[19] Mars, S.: Mixed-Integer Semidefinite Programming with an Application to Truss Topology Design. Dissertation, FAU Erlangen-Nürnberg (2013)

[20] Matter, F., Pfetsch, M.E.: Presolving for mixed-integer semidefinite optimization. INFORMS J. Optimization (2022). https://doi.org/10.1287/ijoo.2022.0079, to appear

[21] McKay, B.D., Piperno, A.: Practical graph isomorphism, II. Journal of Symbolic Computation **60**, 94–112 (2014). https://doi.org/https://doi.org/10.1016/j.jsc.2013.09.003

[22] Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Orbital branching. Mathematical Programming **126**(1), 147–178 (2011). https://doi.org/10.1007/s10107-009-0273-x, http://dx.doi.org/10.1007/s10107-009-0273-x

[23] Pfetsch, M.E., Rehn, T.: A computational comparison of symmetry handling methods for mixed integer programs. Mathematical Programming Computation **11**(1), 37–93 (2019). https://doi.org/10.1007/s12532-018-0140-y

[24] Pilanci, M., Wainwright, M.J., El Ghaoui, L.: Sparse learning via Boolean relaxations. Math. Program. Series B **151**(1), 62–87 (2015). https://doi.org/10.1007/s10107-015-0894-1

[25] Salvagnin, D.: A dominance procedure for integer programming. Master's thesis, University of Padova, Padova, Italy (2005)

[26] Wiese, S.: Symmetry detection in mixed-integer conic programming. https://docs.mosek.com/whitepapers/symmetry.pdf (2022), Mosek Whitepaper