Presolving for Mixed-Integer Semidefinite Optimization

Frederic Matter* Marc E. Pfetsch*

March 2022

Abstract

This paper provides a discussion and evaluation of presolving methods for mixed-integer semidefinite programs. We generalize methods from the mixed-integer linear case and introduce new methods that depend on the semidefinite condition. The methods considered include adding linear constraints, deriving bounds relying on 2×2 minors of the semidefinite constraints, tightening of variable bounds based on solving a semidefinite program with one variable, and scaling of the matrices in the semidefinite constraints. Tightening the bounds of variables can also be used in a node presolving step. Along the way, we discuss how to solve semidefinite programs with one variable using a semismooth Newton method and the convergence of iteratively applying bound tightening. We then provide an extensive computational comparison of the different presolving methods, demonstrating their effectiveness with an improvement in running time of about 22 % on average. The impact depends on the instance type and varies across the methods.

1 Introduction

Presolving is one of the cornerstones of generic mathematical optimization solvers. It changes an instance into an equivalent one that is hopefully easier to solve. This can often be achieved by removing variables or constraints as well as tightening coefficients or bounds of variables. As in the literature, we use the terms presolving and preprocessing interchangeably.

Presolving can have an impressive impact, especially if the underlying solution process can in principle result in an exponential runtime behavior. For instance, Bixby and Rothberg [13] report a slowdown factor of 10.8 when solving Mixed-Integer Programs (MIPs) with disabled root node presolving for CPLEX 8.0; this factor was confirmed by Achterberg and Wunderling [4] for CPLEX 12.5. For Mixed-Integer Nonlinear Programs (MINLPs), Puranik and Sahinidis [50] demonstrate the importance of presolving and bound tightening, in particular: not using presolving significantly slows down the solution process and decreases the number of solved instances within the time limit for the solvers BARON, Couenne, and SCIP. It turns out that bound tightening is essential for strengthening relaxations of non-convex problems. Note that the instances in all of these publications come from publicly available benchmark libraries and are quite diverse and generic. Indeed, presolving is very useful for instances that have been generated by modeling languages. The impact of presolving of course depends on the particular instances and might be less effective for instances that come from a less generic source or are tuned ("presolved") by humans.

In this paper, we consider presolving for general Mixed-Integer Semidefinite Programs

*Department of Mathematics, TU Darmstadt, Germany

(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 $k \in [m]_0 := \{0, \ldots, m\}$, $b \in \mathbb{R}^m$, $\ell_i \in \mathbb{R} \cup \{-\infty\}$, $u_i \in \mathbb{R} \cup \{\infty\}$ for all $i \in [m] := \{1, \ldots, m\}$. The set of indices of integer variables is given by $I \subseteq [m]$. The notation $M \succeq 0$ indicates that some matrix M is positive semidefinite. Throughout this paper, we use the notation $A(y) := \sum_{k=1}^{m} A^k y_k - A^0$ for $y \in \mathbb{R}^m$. Note that in some applications, e.g., reformulations of combinatorial optimization problems, it is more natural to have a positive semidefinite matrix variable $X \succeq 0$, which leads to an equivalent "primal" version of (1). In Appendix A, we outline the equivalence and also explain how to reformulate an MISDP in one form into the other. Our presentation and implementation, however, is based on the form in (1).

While for specific types of MISDPs, several presolving methods are known, this paper focuses on presolving for generic MISDPs. We introduce several new techniques and provide a computational evaluation of different variants. Often, these methods can be seen as a generalization of presolving for mixed-integer programs. We note that several methods that we describe can not only be performed at the root node, but also at further nodes in the tree, which leads to node presolving. In particular, this includes propagation (of variable bounds), which refers to the tightening of some variable bounds based on the bounds of other variables.

In more detail, our contributions are as follows. We start with a brief description of how (1) can be solved in Section 1.1. After a literature review in Section 1.2, we summarize standard presolving methods in Section 1.3 and discuss their relation to solving MISPDs. We then present several valid linear inequalities in Section 2, which can be added during and are then used in presolving. In Section 3, we turn our attention to presolving based on 2×2 minors of positive semidefinite matrices $A(y)$. This involves variable bounds derived from upper bounds on diagonal entries in Section 3.1. Using bounds on off-diagonal entries, further variable bounds are derived in Section 3.2. As a next step, we present a method to tighten variable bounds in Section 4. We prove that iteratively applying this bound tightening converges to a best bound, which can also be computed by solving a single SDP (Section 4.2). The underlying optimization problems for computing a single bound tightening correspond to SDPs with one variable and can be solved by using a semismooth Newton method, see Section 4.3. With similar techniques, one can also compute the tightest scaling of the constraint matrices A^k that does not change the feasible region; this generalizes coefficient tightening, see Section 4.4. Then, as one of the main contributions of this paper, our computational results in Section 5 compare the different presolving methods and their combination. The results show that, for the considered instances, presolving in the root node has limited effect, but node presolving – and bound tightening in particular – can result in a significant speed-up of up to 22 % in comparison to no presolving. Moreover, on the one hand, presolving has a different impact on different types of instances. On the other hand, since the methods only take a negligible amount of time, they can easily be applied without much overhead. In conclusion, the techniques investigated in this paper provide a very good basis for future applications of generic MISDP.

1.1 Solving MISDPs

We start with a brief review of the three main techniques for solving (1):

1. *SDP-based branch-and-bound:* One can adapt the general nonlinear branch-and-bound process, as already proposed by Dakin [21] in 1965, by solving SDPs in each node. Two of the first solvers based on this idea are YALMIP [40] and SCIP-SDP, which was introduced by Mars [43] and continued by Gally [28]. See [30] for an analysis of subproblem properties in the tree.

2. *LP-based branch-and-bound:* The second technique was proposed by Sherali and Fraticelli [53] (see also Krishnan and Mitchell [37]) and applies a linear programming (LP) based cutting-plane algorithm for solving the subproblems in each node of the tree, see the next paragraph for more details. This LP-based approach is also implemented in SCIP-SDP (see [43, 28] for computational results) and YALMIP. Kobayashi and Takano [36] explicitly prove that this cutting-plane method converges to an optimal point for each SDP in the tree.

3. *Outer approximation:* Outer approximation, proposed by Duran and Grossmann [22], was investigated for mixed-integer conic problems by Lubin et al. [41] and is implemented in the solver Pajarito [19]. We will not investigate this approach in this paper, but will present results for the first two.

Notes on the LP-based Approach In the following corollary, we highlight that approximating certain SDPs requires exponentially many linear inequalities, which can be seen by combining results from the literature. This is in contrast to second-order cone programs, for which ε-approximate extended formulations of polynomial size in the input and $\log(1/\varepsilon)$ exist, see [8].

Corollary 1. *There are SDPs of dimension $n \times n$ for which any polyhedral approximation is of size $2^{\Omega(n)}$.*

Proof sketch. Braun et al. [16] proved that one may need polyhedral extended formulations with extension complexity $2^{\Omega(n)}$ to construct tight approximations of the feasible regions of SDPs in $\mathbb{R}^{n\times n}$. The proof is based on a nonnegative rank of size $2^{\Omega(n)}$ for particular instances. Braun et al. [17] showed that the nonnegative rank deviates from the minimal number of inequalities in a polyhedral description in the original dimension $n \times n$ by at most 1. □

When solving general MISDPs of the form (1) with a cutting-plane approach, the positive semidefiniteness of $A(y)$ needs to be enforced through linear cuts. To do so, it is possible to use the following characterization of positive semidefiniteness.

$$\sum_{k=1}^{m} A^k y_k - A^0 \succeq 0 \quad \Leftrightarrow \quad v^\top A(y)\, v = v^\top \Big(\sum_{k=1}^{m} A^k y_k - A^0 \Big) v \geq 0 \quad \forall\, v \in \mathbb{R}^n.$$

Thus, if a given relaxation solution y^* does not satisfy the SDP-constraint $A(y^*) \succeq 0$, there exists $v^* \in \mathbb{R}^n$ such that $(v^*)^\top A(y^*)\, v^* < 0$. Consequently, the valid linear inequality

$$(v^*)^\top \Big(\sum_{k=1}^{m} A^k y_k - A^0 \Big) v^* \geq 0$$

cuts the relaxation solution y^* off. These cuts are sometimes called *eigenvector cuts* or *eigencuts*. A simple choice for v^* is an eigenvector for the smallest eigenvalue of $A(y^*)$, which is negative if the SDP-constraint is violated. Of course, it is also possible to directly add several eigenvector cuts, for example, one for each negative eigenvalue of $A(y^*)$. In our implementation, there are two possibilities to add eigenvector cuts. The first variant *separates* eigenvector cuts during the solution of the LP-relaxation, that is, eigenvector cuts are added

whenever a feasible solution of the LP-relaxation does not satisfy the positive semidefiniteness constraint. This setting will be denoted by "LPA" in our experiments. The second variant, denoted by "LPE", only *enforces* eigenvector cuts, that is, these cuts are only added, if an optimal solution of the LP-relaxation satisfies the integrality constraints, but still violates the positive semidefiniteness constraint (a "lazy-cut" approach).

Although it is not the focus of this paper, let us comment on the computations in [36], who compare the SDP-based approach with their own implementation of an LP-based algorithm. The best performing method in [36] is to use LP-relaxations in which eigenvalue cuts are only generated if all integer variables attain integral values (the lazy-cut approach). This method is quite similar to our method of only enforcing integral solutions (LPE-MIX2), see Section 5. The results of our computations differ in several aspects from [36]: For the LP-based approach, it turns out that it is faster to also separate eigenvector cuts for fractional solutions and not only for integer valued solutions. Our implementation based on SDP-relaxations is much faster on average than the LP-based approach. Note that [36] used an older version of SCIP-SDP with DSDP on the NEOS server. Here, we compare on the same machines, use an improved implementation, and use Mosek as an SDP-solver. Moreover, we test on similar but larger instances compared to [36], see Section 5.1.

1.2 Literature Overview

We first note that SDP-relaxations can be preprocessed to improve their numerical stability, for example by facial reduction techniques, see, e.g, [45, 46, 47]. However, such features so far have neither been implemented into the SDP-solver Mosek, which we use in our computations, nor in our code.

In the following literature review, we concentrate on presolving techniques for problems containing integer variables, since this is the main focus of this paper.

For MIPs, many presolving methods are known, see for instance Brearley et al. [18] and Crowder et al. [20]. We note that details are not needed for understanding our contributions. We will, however, add some pointers to MIP-presolving techniques later and refer to the following literature for more information. An overview and new techniques were presented by Savelsbergh [52]. For a more recent overview see Mahajan [42]. Achterberg [2] discusses the implementation of presolving in detail. Further recent contributions are introduced in Achterberg et al. [3] and Gemander et al. [31]. The last three publications describe the methods implemented in the framework SCIP. Our implementation is SCIP-SDP 4.0.0, which is publicly available at `https://wwwopt.mathematik.tu-darmstadt.de/scipsdp/` and is based on SCIP, available at `https://scipopt.org/`. We refer to [12] for more information on the current SCIP-SDP 4.0.0 release.

Presolving is even more important for MINLPs, see, e.g., Vigerske [56], Belotti et al. [7], Vigerske and Gleixner [57], and Puranik and Sahinidis [50].

Several presolving methods for MISDPs have been proposed by Mars [43], Gally et al. [30], and Gally [28]; we explain the most relevant ones in the following section. Beyond the mentioned references, we are not aware of any other presolving techniques for MISDPs.

1.3 Standard Presolving

Several known presolving steps are (relatively) straightforward to perform. For instance, any linear inequality that might be present in (1) can be presolved as for MIPs (see above for references). The following basic MISDP-specific methods have been introduced by Mars [43, Section 3.3.2] and partly extended by Gally [28]: Fixed variables can be removed by appropriately adjusting the constant matrix A^0. Similarly, (multi-)aggregated variables, i.e., variables that affinely depend on other variables, can be substituted, possibly adjusting the affected matrices A^k, $k \in [m]_0$. Furthermore, one can check whether all matrices A^k for $k \in [m]_0$ are

diagonal. In this case, the SDP-constraint $A(y) \succeq 0$ can be replaced by corresponding linear inequalities. All these steps are automatically performed in our implementation.

Further presolving steps are the following, although they treat rather rare cases and are therefore not implemented. Zero matrices A^k and their corresponding variables y_k can be removed. Moreover, duplicate constraints $A(y) \succeq 0$ or duplicated blocks within $A(y) \succeq 0$ can be detected and removed. Redundant constraints $A(y) \succeq 0$ can be detected in several special cases, e.g., if all variables are binary, all A^k, $k \in [m]$, are positive semidefinite and A^0 is negative semidefinite. If $m = 1$ in the SDP-constraint $A(y) \succeq 0$, i.e., there is only one variable, the feasible region is an interval (see Section 4.3); thus, the SDP-constraint can be removed and the variable bounds can be adjusted. Furthermore, if all matrices A^k, $k \in [m]_0$, contain the same 0 rows and columns, the dimension can be reduced. This last step is automatically performed in our implementation each time an instance is passed to an SDP-solver. Furthermore, Mars [43, Section 3.3.2] discusses methods to detect block structures in the SDP-constraint. Under certain conditions, one can also apply dual presolving. For example, if for some $k \in [m]$ the matrix A^k is positive semidefinite and disjoint from the rest (i.e., A^k has no common nonzero with the other matrices), one can fix y_k to its upper or lower bound, depending on the objective coefficient.

More expensive presolving includes so-called probing, see, e.g., Savelsbergh [52]. Probing tentatively fixes binary variables to 0 and 1 and then checks whether propagation of variable bounds leads to infeasibilities. If this happens, one can fix the binary variable to the opposite value. Moreover, implications between binary variables can be detected. Probing is automatically performed in our implementation, but the propagation methods often do not seem to be strong enough to allow for many probing reductions.

One further method is *optimality based bound tightening* (OBBT) in which one maximizes/minimizes variables over a relaxation of the problem to determine lower and upper variable bounds, see, e.g., Gleixner et al. [32] for a recent variant. This method was adapted for MISDPs by Gally [28] and usually reduces the number of nodes in the tree, but increases running times. It is therefore not considered in our analysis.

Dual fixing is a node presolving method, which is a generalization of reduced cost fixing, and is always used in our implementation, see [30].

We finally note that node presolving has secondary effects. For instance, it affects conflict analysis, which in this context summarizes techniques that derive so-called conflict constraints, i.e., linear, set covering or more general disjunctive constraints, based on the information that a certain node in the branch-and-bound tree is infeasible. For more information, we refer to [1], [59], and [58]. If we use SDP-relaxations, the generated conflicts only arise from so-called conflict graph analysis, which applies if the infeasibility of the node has been determined by propagation of variable bounds. In the LP-based approach, however, conflict analysis also uses LP infeasibility proofs. The computational results in Section 5 briefly treat conflict analysis.

2 Implied Linear Inequalities

The following inequalities are known from the literature and can be added to (1) as linear inequalities. All these inequalities are implied by the SDP-relaxation of (1), but might be useful for standard presolving w.r.t. linear constraints or when solving a linear relaxation.

- Mars [43, Section 3.3.2] observed that the constraint $A(y) \succeq 0$ implies that the diagonal entries of $A(y)$ are nonnegative (*Diagonal Greater equal Zero, DGZ*), i.e., for all $i \in [n]$:

$$\sum_{k=1}^{m} (A^k)_{ii}\, y_k - (A^0)_{ii} \geq 0. \tag{DGZ}$$

- If $A^k_{ii} = A^k_{jj} = 0$ for all $k \in [m]$ and $A^0_{ii}A^0_{jj} \geq 0$ for some $i \neq j \in [n]$, then the following inequality based on products of 2×2 minors (*2-Minor Product, 2MP*) is valid, see Gally [28, Prop. 5.11]:

$$\sum_{k=1}^{m} A^k_{ij}\, y_k \geq A^0_{ij} - \sqrt{A^0_{ii}A^0_{jj}}. \tag{2MP}$$

  Furthermore, if exactly one $A^k_{ij} \neq 0$, then this yields upper or lower bounds for the corresponding variable y_k, depending on the sign of A^k_{ij}. Further similar inequalities can be found in Gally [28, Prop. 5.13].

We also obtain the following slight generalization of the "diagonal-zero-implication cuts (DZI)" introduced by Gally [28], based on an observation of Mars [43]. These inequalities build on the presence of integral variables.

Lemma 2. *Let $i, j \in [n]$ with $i \neq j$ and $A^0_{ij} \neq 0$ as well as $A^0_{ii} \geq 0$. If $A^k_{ij} = 0$ for all $k \in [m]$, $A^k_{ii} = 0$ for all continuous variables $k \in [m] \setminus I$, and $\ell_k \geq 0$ for all integer variables $k \in I$, the following inequality is valid:*

$$\sum_{\substack{k\in I:\\ A^k_{ii}>0}} y_k \geq 1. \tag{DZI}$$

Proof. Any y feasible for (1) satisfies $A(y) \succeq 0$ and therefore also $A(y)_{ii} \geq 0$ and $A(y)_{jj} \geq 0$. The 2×2 minor w.r.t. i and j yields $A(y)_{ii} \cdot A(y)_{jj} - (A(y)_{ij})^2 \geq 0$. By assumption $A(y)_{ij} = A^0_{ij} \neq 0$. This implies that $A(y)_{ii} \cdot A(y)_{jj} > 0$ and therefore $A(y)_{ii} > 0$ (and $A(y)_{jj} > 0$). Since $A^k_{ii} = 0$ for all $k \in [m] \setminus I$ and $\ell_k \geq 0$ for all $k \in I$, we obtain:

$$0 < A(y)_{ii} = \sum_{k=1}^{m} A^k_{ii}\, y_k - A^0_{ii} = \sum_{k\in I} A^k_{ii}\, y_k - A^0_{ii} \leq \sum_{\substack{k\in I:\\ A^k_{ii}>0}} A^k_{ii}\, y_k - A^0_{ii}.$$

Since $A^0_{ii} \geq 0$, this implies that at least one variable y_k with $k \in I$ and $A^k_{ii} > 0$ has to be positive, i.e., at least 1. □

The following inequalities, called *2-Minor Linear Constraints (2ML)*, are a special case of eigenvector cuts (see Section 1.1). For all $Z \succeq 0$, we have

$$Z_{ii} + Z_{jj} - 2\, Z_{ij} \geq 0, \tag{2}$$
$$Z_{ii} + Z_{jj} + 2\, Z_{ij} \geq 0. \tag{3}$$

This follows by restricting to the 2×2 minor w.r.t. i and j and multiplying from left and right by $(1,-1)^\top$ and $(1,1)^\top$, respectively. If $Z = A(y)$, we obtain for the first inequality

$$\begin{aligned} & \sum_{k=1}^{m} A^k_{ii}\, y_k - A^0_{ii} + \sum_{k=1}^{m} A^k_{jj}\, y_k - A^0_{jj} - 2\Big(\sum_{k=1}^{m} A^k_{ij}\, y_k - A^0_{ij}\Big) \geq 0 \\ \Leftrightarrow \quad & \sum_{k=1}^{m} \Big(A^k_{ii} + A^k_{jj} - 2\, A^k_{ij}\Big)\, y_k \geq A^0_{ii} + A^0_{jj} - 2A^0_{ij}, \end{aligned} \tag{2ML}$$

and similarly for the second inequality. As above, these inequalities are implied by the SDP-constraint $A(y) \succeq 0$, but might be used for propagation of variable bounds.

3 Presolving for 2×2 minors

In this section, we develop methods that are based on taking 2×2 minors of a positive semidefinite matrix.

3.1 Using Bounds on the Diagonal

Lemma 3. *Consider $Z \succeq 0$ with $0 \leq Z_{ii} \leq U_{ii}$ for all $i \in [n]$. Then*

$$-\sqrt{U_{ii}\, U_{jj}} \leq Z_{ij} \leq \sqrt{U_{ii}\, U_{jj}} \tag{4}$$

holds for all i, $j \in [n]$.

Proof. Since Z is positive semidefinite, we have $Z_{ii}Z_{jj} - Z_{ij}^2 \geq 0$. Rewriting this inequality yields $Z_{ij}^2 \leq Z_{ii}Z_{jj} \leq U_{ii}\, U_{jj}$. Taking the square root shows the claim. □

Remark 4.

- *The bounds in Lemma 3 are tight, even for a rank-1 matrix Z: consider the rank-1 all-ones matrix.*
- *Inequality (2) yields $Z_{ij} \leq \frac{1}{2}(Z_{ii} + Z_{jj}) \leq \frac{1}{2}(U_{ii} + U_{jj})$. This derived bound is dominated by (4), because*

$$Z_{ij} \leq \sqrt{U_{ii} \cdot U_{jj}} \leq \tfrac{1}{2}\big(U_{ii} + U_{jj}\big),$$

using the inequality between the arithmetic and geometric mean.

Lemma 3 can partly be translated to the matrix pencil format $A(y)$ by defining

$$\tilde{U}_{ij} \coloneqq \sum_{k \in [m] : A^k_{ij} > 0} A^k_{ij}\, u_k + \sum_{k \in [m] : A^k_{ij} < 0} A^k_{ij}\, \ell_k - A^0_{ij}.$$

Thus, for any $\ell \leq y \leq u$ we have $A(y)_{ij} \leq \tilde{U}_{ij}$. This directly yields:

Lemma 5. *For any solution $y \in \mathbb{R}^m$ of (1), we have*

$$-\sqrt{\tilde{U}_{ii}\, \tilde{U}_{jj}} \leq A(y)_{ij} \leq \sqrt{\tilde{U}_{ii}\, \tilde{U}_{jj}} \tag{5}$$

for all i, $j \in [n]$.

The downside of Inequalities (5) is that they can be quite weak if $A(y)_{ij}$ depends on many variables. We therefore concentrate on the case in which each entry $A(y)_{ij}$ depends on one variable only, i.e., there exists $k = k(i,j) \in [m]$ such that $A(y)_{ij} = A^k_{ij}\, y_k - A^0_{ij}$ with $A^k_{ij} \neq 0$. In this case, Inequalities (5) are equivalent to

$$\frac{-\sqrt{\tilde{U}_{ii}\, \tilde{U}_{jj}} + A^0_{ij}}{A^k_{ij}} \leq y_k \leq \frac{\sqrt{\tilde{U}_{ii}\, \tilde{U}_{jj}} + A^0_{ij}}{A^k_{ij}}, \tag{PropUB}$$

if $A^k_{ij} > 0$ and similarly if $A^k_{ij} < 0$. If $k \in I$, i.e., variable y_k is integral, the lower bound can be rounded up and the upper bound down. In our implementation, these inequalities are used in presolving and possibly for propagation of variable bounds in every node, which is denoted by *Propagate Upper Bounds (PropUB)*. Again, since Inequalities (PropUB) are valid for the SDP-relaxation, integral variables have to be present or a linear relaxation has to be solved in order for Inequalities (PropUB) to be computationally useful.

By using trace constraints, one can also compute different bounds on the off-diagonal elements as follows; this slightly strengthens Lemma 1 of [29].

Lemma 6. *Consider $Z \succeq 0$ with $\operatorname{tr}(Z) \leq \alpha$. Then*

$$-\tfrac{\alpha}{2} \leq Z_{ij} \leq \tfrac{\alpha}{2} \tag{6}$$

holds for all $i, j \in [n]$ with $i \neq j$.

Proof. Since $Z \succeq 0$, we again have $Z_{ij}^2 \leq Z_{ii} Z_{jj}$. Using the trace constraint and the fact that the diagonal entries are nonnegative, we obtain $Z_{ii} + Z_{jj} \leq \alpha$. This implies

$$Z_{ii}\, Z_{jj} \leq Z_{ii}(\alpha - Z_{ii}) = \alpha Z_{ii} - Z_{ii}^2.$$

Taking the derivative and equating 0 yields a maximal point $Z_{ii}^\star = \frac{\alpha}{2}$. Consequently,

$$Z_{ij}^2 \leq Z_{ii}\, Z_{jj} \leq \alpha Z_{ii}^\star - (Z_{ii}^\star)^2 = \tfrac{\alpha^2}{2} - \tfrac{\alpha^2}{4} = \tfrac{\alpha^2}{4}.$$

Taking the square root shows the claim. □

Inequalities (6) can again be transferred to $A(y) \succeq 0$, but with the same disadvantages. Therefore, we only use these inequalities in the case that $A(y)_{ij}$ only depends on a single variable. As before, integrality of variables can be exploited for rounding the bounds.

3.2 Using Bounds on the Off-Diagonal

We now derive affine inequalities that depend on 2×2 minors. The following result is motivated by and generalizes the special case in Nohra et al. [44].

Lemma 7. *Consider a positive semidefinite matrix $Z \in \mathbb{R}^{n \times n}$ with $L \leq Z \leq U$, where the inequalities are meant componentwise. Then for all i and $j \in [n]$:*

$$U_{jj} Z_{ii} \geq 2 L_{ij} Z_{ij} - L_{ij}^2 \qquad \text{and} \qquad U_{jj} Z_{ii} \geq 2 U_{ij} Z_{ij} - U_{ij}^2. \tag{7}$$

Proof. We first obtain

$$(Z_{ij} - L_{ij})^2 \geq 0 \quad \Leftrightarrow \quad Z_{ij}^2 \geq 2\, L_{ij} Z_{ij} - L_{ij}^2.$$

The 2×2 minor for i and j implies $Z_{jj}\, Z_{ii} - Z_{ij}^2 \geq 0$. Together with $Z_{ii} \geq 0$, this yields

$$2\, L_{ij} Z_{ij} - L_{ij}^2 \leq Z_{ij}^2 \leq Z_{jj}\, Z_{ii} \leq U_{jj}\, Z_{ii}.$$

The second inequality arises similarly. □

Remark 8.

- *Inequalities (7) are implied by the SDP-constraint and thus can only be useful when solving LPs or in the presence of integral variables. Moreover, assume that $L_{ij} < 0$ and $U_{ij} > 0$, which is typical for $i \neq j$. Then these inequalities are non-trivial, that is, the right-hand-side is nonnegative, if $Z_{ij} \leq L_{ij}/2$ and $Z_{ij} \geq U_{ij}/2$, respectively.*
- *Note that cuts like (2) or (3) do not take the lower and upper bounds into account. Thus, Inequalities (7) might further strengthen an LP-relaxation.*
- *However, if we use $Z_{ii} \leq U_{ii}$, the last Inequality in (7) yields (if $U_{ij} > 0$):*

$$Z_{ij} \leq \frac{U_{jj} U_{ii} + U_{ij}^2}{2 U_{ij}}. \tag{8}$$

The right hand-side is stronger than $Z_{ij} \leq U_{ij}$ if $U_{ii} U_{jj} \leq U_{ij}^2$. If U is positive semidefinite, this never happens. Thus, one should use Inequalities (7) instead of (8).

We transfer Inequalities (7) to the form $A(y) \succeq 0$ as in Section 3.1. For the second inequality in (7), this yields:

$$\begin{aligned} & 2\,\tilde{U}_{ij} A(y)_{ij} - \tilde{U}_{jj} A(y)_{ii} \le \tilde{U}_{ij}^2 \\ \Leftrightarrow \quad & \sum_{k=1}^{m} 2\,\tilde{U}_{ij}\, A_{ij}^k\, y_k - \sum_{k=1}^{m} \tilde{U}_{jj}\, A_{ii}^k\, y_k \le \tilde{U}_{ij}^2 + \sum_{k=1}^{m} 2\,\tilde{U}_{ij}\, A_{ij}^0 - \sum_{k=1}^{m} \tilde{U}_{jj}\, A_{ii}^0, \end{aligned} \tag{2MV}$$

where $\tilde{U}_{ij}$ and $\tilde{U}_{jj}$ are defined as in Section 3.1. These inequalities are referred to as *2-Minor Variable Bounds (2MV)*.

A particular case in which Inequalities (7) might be useful arises in SDP-relaxations of quadratic programs or in truss topology optimization as considered in the following corollary. For a short description of truss topology optimization see Section B.5.

Corollary 9. *Consider* $(X, x, t) \in \mathbb{R}^{n\times n} \times \mathbb{R}^n \times \mathbb{R}$ *satisfying*

$$\begin{pmatrix} t & x^\top \\ x & X \end{pmatrix} \succeq 0, \quad \ell \le x \le u, \quad t \le \beta,$$

where t *is a scalar variable. Then for all* $i \in [n]$*:*

$$\beta\, X_{ii} \ge 2\ell_i x_i - \ell_i^2 \qquad \text{and} \qquad \beta\, X_{ii} \ge 2u_i x_i - u_i^2.$$

4 Tightening Procedures

In this section, we investigate how SDP-constraints $A(y) \succeq 0$ can be used to tighten variable bounds and scale matrices A^k.

4.1 Bound Tightening

For an index $k \in [m]$, define

$$P_k := \{i \in [m] \setminus \{k\} \,:\, A^i \succeq 0\}, \qquad N_k := \{i \in [m] \setminus \{k\} \,:\, A^i \preceq 0\},$$

as well as

$$\underline{\mu}_k := \begin{cases} \inf \left\{ \mu \,:\, A^k\,\mu + \sum_{i\in P_k} A^i\, u_i + \sum_{j\in N_k} A^j\, \ell_j - A^0 \succeq 0 \right\} & \text{if } \begin{array}{ll} u_i < \infty & \forall i \in P_k, \\ \ell_i > -\infty & \forall i \in N_k, \end{array} \\ -\infty & \text{otherwise,} \end{cases} \tag{9}$$

$$\overline{\mu}_k := \begin{cases} \sup \left\{ \mu \,:\, A^k\,\mu + \sum_{i\in P_k} A^i\, u_i + \sum_{j\in N_k} A^j\, \ell_j - A^0 \succeq 0 \right\} & \text{if } \begin{array}{ll} u_i < \infty & \forall i \in P_k, \\ \ell_i > -\infty & \forall i \in N_k, \end{array} \\ +\infty & \text{otherwise.} \end{cases} \tag{10}$$

Both $\underline{\mu}_k$ and $\overline{\mu}_k$ might be $\pm\infty$, even if all bounds are finite, for instance, if A^k is negative or positive definite, respectively. Moreover, both might simultaneously be finite. The two SDPs in (9) and (10) only contain a single variable and can be solved with the technique discussed in Section 4.3 below.

The following lemma shows that the lower or upper bounds of the variables can be tightened, depending on the semidefiniteness of the coefficient matrices. This procedure is referred to as *Tighten Bounds (TB)* in our experiments.

Lemma 10 (Tighten Bounds (TB))**.** *Let all* A^k*,* $k \in [m]$*, be (positive or negative) semidefinite. Then,* $A(y) \succeq 0$ *implies that* $\underline{\mu}_k \le y_k \le \overline{\mu}_k$ *for all* $k \in [m]$*. Finite bounds can be rounded for integral variables.*

Proof. Suppose that $y_k < \underline{\mu}_k$ or $y_k > \overline{\mu}_k$. Then, by definition of $\underline{\mu}_k$ and $\overline{\mu}_k$, there exists $x \in \mathbb{R}^n$ with

$$\begin{aligned}
0 &> x^\top \Big(A^k\, y_k + \sum_{i \in P_k} A^i\, u_i + \sum_{i \in N_k} A^i\, \ell_i - A^0 \Big) x \\
&= x^\top A^k x\, y_k + \sum_{i \in P_k} \underbrace{x^\top A^i x}_{\geq 0}\, u_i + \sum_{i \in N_k} \underbrace{x^\top A^i x}_{\leq 0}\, \ell_i - x^\top A^0 x \\
&\geq x^\top A^k x\, y_k + \sum_{i \in P_k} x^\top A^i x\, y_i + \sum_{i \in N_k} x^\top A^i x\, y_i - x^\top A^0 x \\
&= x^\top \Big(\sum_{i=1}^{m} A^i\, y_i - A^0 \Big) x,
\end{aligned}$$

which is a contradiction to $A(y) \succeq 0$. Thus, $\underline{\mu}_k \leq y_k \leq \overline{\mu}_k$. □

Remark 11.

- *The conditions of Lemma 10 are frequently fulfilled for instances that we consider in this paper; namely for 75 out of 185 instances in our testset, all matrices A^k are positive semidefinite, see Section 5.1. If some matrix A^k is indefinite, one could write $A^k = B^k - C^k$ with B^k, $C^k \succeq 0$ and duplicate y_k.*
- *One could also explicitly add the constraint $\ell_k \leq \mu \leq u_k$ to (9) and (10). For instance, this makes the problems bounded if the bounds are finite, see Section 4.3.*
- *If all A^k, $k \in [m]_0$, are diagonal matrices, $A(y) \succeq 0$ specializes to a linear inequality $a^\top y - a_0 \geq 0$ with $a \in \mathbb{R}^m$ and $a_0 \in \mathbb{R}$. If $a_k > 0$, we obtain*

$$y_k \geq \mu_k = \frac{1}{a_k} \Big(a_0 - \sum_{\substack{i : a_i > 0 \\ i \neq k}} a_i\, u_i - \sum_{j : a_j < 0} a_j\, \ell_j \Big),$$

  which is exactly linear bound tightening, i.e., Lemma 10 generalizes the linear case.
- *We note that Inequalities (4) are implied by Lemma 10. This can be seen as follows: Assume that we have a matrix $Z \succeq 0$ with some finite lower bounds $L \in \mathbb{R}^{n \times n}$ (the exact values are not important, but they make (10) finite). Write $Z = \sum_{i,j=1}^{n} E^{ij}\, Z_{ij} \succeq 0$, where $E^{ij} \in \mathbb{R}^{n \times n}$ is 0 except for positions (i,j) and (j,i), where it is 1. Then the optimal value $\bar{\mu}$ of (10) for variable Z_{ij} yields that the 2×2 minor for i and j is nonnegative, i.e., $U_{ii}\, U_{jj} - \bar{\mu}^2 \geq 0$, which is (4). In comparison to the bounds of Lemma 10, the ones in (4) (or (5)) can be computed more efficiently and depend on fewer variable bounds.*

4.2 Convergence of Bound Tightening

Lemma 10 can be applied iteratively and we investigate the convergence of this process. This section uses similar arguments as in Belotti et al. [6].

Let $\underline{\mu}(\ell, u)$ and $\overline{\mu}(\ell, u) \in \mathbb{R}^m \cup \{\pm\infty\}$ be the lower and upper bounds derived from Lemma 10 for each variable, where the constraint $\ell_k \leq \mu \leq u_k$ is added to (9) and (10). Define the interval set $\mathcal{I} := \{(\ell, u) \in \mathbb{R}^n \times \mathbb{R}^n \,:\, \ell \leq u\}$ with the following ordering:

$$(\ell, u) \leq_{\mathcal{I}} (\ell', u') \quad \Leftrightarrow \quad \ell' \leq \ell,\ u \leq u'$$

for (ℓ, u), $(\ell', u') \in \mathcal{I}$. Thus, bounds (ℓ, u) are at least as tight as (ℓ', u'), if $(\ell, u) \leq_{\mathcal{I}} (\ell', u')$. Let

$$F \colon \mathcal{I} \to \mathcal{I},\ (\ell, u) \mapsto \big(\max(\ell, \underline{\mu}(\ell, u)), \min(u, \overline{\mu}(\ell, u)) \big),$$

where min/max is applied componentwise. Thus, F represents one step of bound tightening according to Lemma 10 and makes sure that the bounds do not get weaker.

Lemma 12. *F is a contraction, i.e., $F(\ell,u) \leq_{\mathcal{I}} (\ell,u)$ for all $(\ell,u) \in \mathcal{I}$, and monotone, i.e., $(\ell,u) \leq_{\mathcal{I}} (\ell',u')$ implies $F(\ell,u) \leq_{\mathcal{I}} F(\ell',u')$.*

Proof. The fact that F is a contraction follows by definition of the max and min operations.

For monotonicity, we concentrate on the upper bounds (the lower bounds are similar). Let $f(\ell,u) \coloneqq \min(u, \overline{\mu}(\ell,u))$ and similarly for $f(\ell',u')$. Assume for a contradiction that $(\ell,u) \leq_{\mathcal{I}} (\ell',u')$ (and thus $\ell' \leq \ell$, $u \leq u'$), but $\mu \coloneqq f(\ell,u)_k > f(\ell',u')_k =: \mu'$ for some $k \in [m]$. Thus, by definition of μ', the matrix $A^k\,\mu + \sum_{i\in P_k} A^i\,u'_i + \sum_{j\in N_k} A^j\,\ell'_j - A^0$ is not positive semidefinite. Therefore, there exists $x \in \mathbb{R}^n$ with

$$\begin{aligned}
0 &> x^\top \Big(A^k\,\mu + \sum_{i\in P_k} A^i\,u'_i + \sum_{i\in N_k} A^i\,\ell'_i - A^0\Big)x \\
&= x^\top A^k x\,\mu + \sum_{i\in P_k} \underbrace{x^\top A^i x}_{\geq 0}\,u'_i + \sum_{i\in N_k} \underbrace{x^\top A^i x}_{\leq 0}\,\ell'_i - x^\top A^0 x \\
&\geq x^\top A^k x\,\mu + \sum_{i\in P_k} x^\top A^i x\,u_i + \sum_{i\in N_k} x^\top A^i x\,\ell_i - x^\top A^0 x \\
&= x^\top \Big(A^k\,\mu + \sum_{i\in P_k} A^i\,u_i + \sum_{i\in N_k} A^i\,\ell_i - A^0\Big)x,
\end{aligned}$$

which is a contradiction to the last matrix in parentheses being positive semidefinite by definition of μ. □

Theorem 13. *The operator F has a unique greatest fixed point* $\operatorname{gfix}(F) \coloneqq \sup\{(\ell,u) \in \mathcal{I} : (\ell,u) \leq_{\mathcal{I}} F(\ell,u)\}$.

Proof. Note that $\mathcal{I}$ forms a complete lattice. Since F is a contraction, we always have $F(\ell,u) \leq_{\mathcal{I}} (\ell,u)$, thus $\{(\ell,u) \in \mathcal{I} : (\ell,u) \leq_{\mathcal{I}} F(\ell,u)\}$ contains all fixed points. The result then follows by the Knaster-Tarski Theorem [55] (see, e.g., Fritz [27, Theorem 20.4]). □

As in [6], we define the size $|(\ell,u)|$ of the interval $(\ell,u) \in \mathcal{I}$ as $\sum_{i=1}^m u_i - \ell_i$. Then [6] shows that $|\operatorname{gfix}(F)| \geq |(\ell,u)|$ for all fixed points (ℓ,u) of F. Thus, $\operatorname{gfix}(F)$ is the solution of

$$\begin{aligned}
\max \quad & |(\ell,u)| \\
\text{s.t.} \quad & (\ell,u) \leq_{\mathcal{I}} F(\ell,u), \\
& (\ell,u) \leq_{\mathcal{I}} (\ell^0,u^0),
\end{aligned}$$

where (ℓ^0,u^0) denote the initial bounds. This can be written as the following SDP:

$$\begin{aligned}
\max \quad & \sum_{i=1}^m u_i - \ell_i \\
\text{s.t.} \quad & A^k\,\ell_k + \sum_{i\in P_k} A^i\,u_i + \sum_{j\in N_k} A^j\,\ell_j - A^0 \succeq 0 \quad \forall k \in [m], \\
& A^k\,u_k + \sum_{i\in P_k} A^i\,u_i + \sum_{j\in N_k} A^j\,\ell_j - A^0 \succeq 0 \quad \forall k \in [m], \\
& \ell^0 \leq \ell,\ u \leq u^0, \\
& \ell \leq u.
\end{aligned} \tag{11}$$

Let $(\ell^\star,u^\star)$ be an optimal solution of (11). Then this solution is a fixed point: By the constraints, we have $\ell^\star \geq \underline{\mu}$ and $u^\star \leq \overline{\mu}$. Thus, these bounds would not be tightened by F. Moreover, consider the sequence of bounds $\{(\ell_k,u_k)\}$ produced by iteratively applying F as long as this changes some bounds. Since F is monotone, $|(\ell_k,u_k)|$ is decreasing. Thus, $\{(\ell_k,u_k)\}$ will converge to $(\ell^\star,u^\star)$.

In our implementation, we iteratively apply Lemma 10 as long as this changes bounds of variables, instead of solving the SDP (11), because (11) is quite expensive to solve and we can round bounds of integer variables after each iteration. Note that rounding for integer variables complicates the analysis of fixed points. Indeed, [14] show that deciding the existence of an integral fixed point is NP-complete.

As we shall see, bound tightening is often successful deeper in the tree using bounds tightened by other components of the solver.

4.3 Computing Tightening Scalings

While in the linear case the values $\underline{\mu}_k$ and $\overline{\mu}_k$ can be computed easily, in the general case, it amounts to solving an SDP with one variable. For this, let us rewrite (9) and (10) with scalar lower and upper bounds ℓ and u, respectively, objective direction $\gamma \in \{\pm 1\}$, and appropriate $A, B \in \mathbb{R}^{n\times n}$ as

$$\mu^\star \coloneqq \inf \{\gamma\, \mu \,:\, \mu\, A - B \succeq 0,\ \ell \leq \mu \leq u\}. \tag{12}$$

Problem (12) can be solved in different ways. In fact, there are several special cases in which (12) – with infinite bounds – is easy to solve, for instance, if $A = 0$ or $B = 0$. If A is positive definite, there exists an invertible matrix V with $V^\top A V = I_n$, where I_n is the $n \times n$ identity matrix. It is then easy to see that $\mu^\star = \lambda_{\max}(V^\top B V)$, the maximal eigenvalue of $V^\top B V$. If there exists $\hat{\mu}$ with $\hat{\mu}A - B \succ 0$, Pong and Wolkowicz [49] or Jiang and Li [35] ([49] cites Lancaster and Rodman [38]) describe an algorithm based on Cholesky decomposition; these articles arise in the context of generalized trust region problems. One final special case is the one in which A and B are simultaneously diagonizable: In this case there exists an invertible matrix V with $V^\top(\mu A - B)V = \mu\, C - D$, where C and D are diagonal matrices; then after computing this decomposition, the problem is easy to solve.

Here, we are interested in the general case of Problem (12). Inspired by [54, 34], we consider a semismooth Newton method. We state and prove the following for completeness.

Lemma 14. *For any two symmetric matrices $A,\ B \in \mathbb{R}^{n\times n}$, the function $f\colon \mathbb{R} \to \mathbb{R}$, $\mu \mapsto \lambda_{\min}(\mu\, A - B)$ is concave and hence continuous.*

Proof. For a symmetric matrix $C \in \mathbb{R}^{n\times n}$, $\lambda_{\min}(C) = \min\{x^\top C x \,:\, \|x\|_2 = 1\}$, see, e.g., [15]. Consequently, $C \mapsto \lambda_{\min}(C)$ is the minimum of linear functions and thus concave. Therefore, $\mu \mapsto \lambda_{\min}(\mu\, A - B)$ is concave as the composition with an affine function. □

Lemma 14 implies that Problem (12) is convex. Moreover, if the optimal value of (12) is finite, it is attained: Otherwise, assume $\gamma = 1$ and that there exists a sequence (μ_k) of feasible points with $\mu_k \to \mu^\star$, where $\mu^\star$ is the value of (12). Since f is continuous, we obtain $f(\mu_k) \to f(\mu^\star)$ and hence $f(\mu^\star) \geq 0$, i.e., $\mu^\star$ is feasible.

To describe the semismooth Newton method, we state the following for completeness.

Lemma 15. *Let $\hat{\mu} \in \mathbb{R}$ and $\hat{v}$ be a unit eigenvector for $\lambda_{\min}(\hat{\mu}\, A - B)$. Then $\hat{v}^\top A \hat{v}$ is a supergradient, i.e.,*

$$\lambda_{\min}(\mu\, A - B) \leq \lambda_{\min}(\hat{\mu}\, A - B) + (\mu - \hat{\mu})\, \hat{v}^\top A \hat{v}$$

for all $\mu \in \mathbb{R}$. In particular, if $\hat{v}^\top A \hat{v} = 0$, then $\lambda_{\min}(\hat{\mu}\, A - B)$ is maximal.

Proof. By definition of $\hat{v}$, we have $\lambda_{\min}(\hat{\mu}\, A - B) = \hat{v}^\top(\hat{\mu}\, A - B)\hat{v}$. This implies

$$\hat{v}^\top(\mu\, A - B)\hat{v} = \hat{v}^\top(\hat{\mu}\, A - B)\hat{v} + (\mu - \hat{\mu})\, \hat{v}^\top A \hat{v} = \lambda_{\min}(\hat{\mu}\, A - B) + (\mu - \hat{\mu})\, \hat{v}^\top A \hat{v}.$$

Since $\lambda_{\min}(\mu\, A - B) \leq \hat{v}^\top(\mu\, A - B)\hat{v}$, the claim follows. □

Algorithm 1: Semismooth Newton method

Input: Matrices A and B, scalar lower and upper bounds $\ell < u$

Output: Solution of $\min\{\mu : \mu A - B \succeq 0,\ \ell \le \mu \le u\}$ or "infeasible"

1 compute unit eigenvector w for minimal eigenvalue λ of $A u - B$;
2 **if** $\lambda < 0$ *and* $w^\top A w > 0$ **then**
3 | return "infeasible";
4 compute unit eigenvector v for minimal eigenvalue λ of $A \ell - B$;
5 **if** $\lambda \ge 0$ **then**
6 | return ℓ;
7 **if** $v^\top A v \le 0$ **then**
8 | return "infeasible";
9 $\mu_0 \leftarrow \ell$, $\lambda_0 \leftarrow \lambda$, $v_0 \leftarrow v$, $k \leftarrow 0$;
10 **while** $\lambda_k < 0$ *and* $(v^k)^\top A v^k > 0$ **do**
11 | $\mu_{k+1} = \mu_k - \frac{\lambda_k}{(v^k)^\top A v^k}$;
12 | **if** $\mu_{k+1} > u$ **then**
13 | | break;
14 | compute unit eigenvector v^{k+1} for minimal eigenvalue λ_{k+1} of $A \mu_{k+1} - B$;
15 | $k \leftarrow k + 1$;
16 **if** $\lambda_k < 0$ **then**
17 | return "infeasible"
18 **else**
19 | return μ_k

Algorithm 1 provides the details of the resulting algorithm for the case $\gamma = 1$, using the following considerations; the algorithm for $\gamma = -1$ is very similar.

- In the case of Step 3, we use Lemma 15 for $\hat{\mu} = u$, $\hat{v} = w$ to get

$$\lambda_{\min}(\mu A - B) \le \underbrace{\lambda}_{<0} + \underbrace{(\mu - u)}_{\le 0} \underbrace{\hat{v}^\top A v}_{>0} < 0$$

  for every μ. Therefore the problem is infeasible.
- In Step 6, ℓ is clearly the optimal solution.
- In Step 8, we have $\lambda < 0$ and $v^\top A v \le 0$. Again using Lemma 15 for $\hat{\mu} = \ell$, $\hat{v} = v$, we get

$$\lambda_{\min}(\mu A - B) \le \underbrace{\lambda}_{<0} + \underbrace{(\mu - \ell)}_{\ge 0} \underbrace{\hat{v}^\top A v}_{<0} < 0$$

  for all μ, and the problem is infeasible.
- Step 11 computes μ_{k+1} such that $\lambda_k + (\mu_{k+1} - \mu_k)(v^k)^\top A v^k = 0$, i.e., the eigenvalue estimation via Lemma 15 becomes 0 (this is akin to the Newton iteration).
- Note that because of the while conditions, the sequence (μ_k) is strictly monotonously increasing.

Remark 16. *We can apply general convergence theory, for instance, Theorem 7.5.3 in [25] (see also Qi and Sun [51]), which proves that the semismooth Newton method converges Q-superlinearly to a zero $\mu^\star$ of $f(\mu) = \lambda_{\min}(\mu A - B)$, given that $\partial f(\mu^\star)$ is nonsingular and the starting point lies near $\mu^\star$. Since f is concave, f is semismooth and the theorem can be applied.*

As noted above, since we start with $\mu_0 = \ell$, after Steps 6 and 8, the sequence (μ_k) is strictly monotonously increasing. Therefore, the process always globally converges. However, if $\partial f(\mu_k)$ or $\partial f(\mu^\star)$ becomes singular, we cannot rely on Q-superlinear convergence.

4.4 Coefficient Tightening

We now consider ways to "tighten" matrices A_k, which we denote by (TM) in our experiments. To this end, define

$$\tilde{\mu}_k = \min\{\mu \,:\, A^k\,\mu - A^0 \succeq 0,\ \ell_k \le \mu \le u_k\} \tag{13}$$

and $\hat{\mu}_k = \min\{\tilde{\mu}_k, 1\}$ for $k \in [m]$.

Lemma 17 (Tighten Matrices (TM))**.** *Let $A^k \succeq 0$ for all $k \in [m]$ and $y \in \mathbb{R}^m$ with $y_k \in \{0,1\}$ for all integral variables $k \in I$ and $\ell_k \ge 0$ for $k \notin I$. Then for all $\ell \le y \le u$:*

$$A(y) \succeq 0 \quad \Leftrightarrow \quad \sum_{k=1}^{m} \hat{\mu}_k\, A^k\, y_k - A^0 \succeq 0,$$

where we define $\hat{\mu}_k = 1$ for $k \notin I$.

Proof. First assume that $\sum_{k=1}^m \hat{\mu}_k\, A^k\, y_k - A^0 \succeq 0$. Since by assumption $\ell_k \ge 0$ for all $k \in [m]$, we get $0 \le \hat{\mu}_k \le 1$. Then for every $x \in \mathbb{R}^n$

$$0 \le x^\top \Big(\sum_{k=1}^{m} \hat{\mu}_k\, A^k\, y_k - A^0 \Big) x = \sum_{k=1}^{m} \hat{\mu}_k \underbrace{x^\top A^k x\, y_k}_{\ge 0} - x^\top A^0 x \le x^\top \Big(\sum_{k=1}^{m} A^k\, y_k - A^0 \Big) x,$$

which implies that $A(y) \succeq 0$.

We now assume that $A(y) \succeq 0$. By removing terms with $y_k = 0$ for $k \in I$, we can assume that $y_k = 1$ for all $k \in I$. Thus, $\sum_{k=1}^m A^k - A^0 \succeq 0$. If $\hat{\mu}_k = 1$ for all $k \in [m]$ then the statement is directly clear. Therefore assume that there exists $k \in I$ with $\hat{\mu}_k = \tilde{\mu}_k < 1$. But then already $\hat{\mu}_k\, A^k - A^0 \succeq 0$. Adding the positive semidefinite matrices A^ℓ for $\ell \in [m] \setminus \{k\}$ does not change this, which shows the claim. □

Remark 18. *In the linear case (see Remark 11) with a linear inequality $a^\top y - a_0 \ge 0$, where $a \in \mathbb{R}^n_+$, $a_0 \in \mathbb{R}$, and the variables y are binary, coefficient tightening would tighten coefficient a_j to $\min\{a_j, a_0\}$. If $a_j > a_0 \ge 0$, then $\tilde{\mu}_j = a_0/a_j < 1$. Thus, Lemma 17 would change coefficient a_j to $\tilde{\mu}_j \cdot a_j = a_0$, i.e., the same tightening. In this sense, Lemma 17 generalizes coefficient tightening from the linear case.*

5 Computational Experiments

In this section, we empirically demonstrate the impact of the presented presolving routines for the SDP-based branch-and-bound approach and the LP-based cutting-plane approach.

We use SCIP-SDP 4.0.0 for solving the MISDPs, where all the routines mentioned in the previous sections are implemented. SCIP-SDP interfaces with SCIP 7.0.4, and we use Mosek 9.2.40 for solving the continuous SDP-relaxations in the SDP-based approach, and SoPlex 5.0.2 for the continuous LP-relaxations in the cutting-plane approach. 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.

The code, an online supplement, and the instances can be obtained via the webpage of the second author.

5.1 Instances

We use a testset consisting of 185 instances for different applications, which are very briefly described in Appendix B. Namely, 43 instances are Cardinality Least Squares (CLS) problems, 32 instances are Min-k-Partitioning (MkP) problems, 38 instances are Truss Topology

Design (TTD) problems, and 46 instances are RIP problems. Moreover, there are 26 random MISDPs in the testset. For 24 CLS problems, all 38 TTD problems, and 13 random MISDPs, all matrices A^k are positive semidefinite. Thus, for these 75 instances, the two tightening procedures from Section 4 can be applied. Note that the random MISDPs and the RIP instances in our testset are larger than the random MISDPs and RIP instances used by [36]. Statistics for each instance such as the number of SDP- and LP-constraints, the maximal dimension of the SDP-constraints, the number of binary and continuous variables, and whether all matrices $A^k \succeq 0$ can be obtained from Tables 47 and 84 in the online supplement.

5.2 Settings

We use the following names for the algorithmic variants in which each different presolving routine described above is active and all other routines are deactivated.

- Basic linear inequalities:
  - DGZ: add (DGZ) in presolving;
  - DZI: add (DZI) in presolving;
- Tightening procedures only in presolving:
  - TM: use Lemma 17 in presolving;
  - TB-Pre: apply Lemma 10 only in presolving;
- Linear inequalities based on 2×2 minors:
  - 2ML: add (2ML) in presolving;
  - 2MP: add (2MP) in presolving;
  - 2MV: add (2MV) in presolving;
- Propagation (of variable bounds) and tightening procedures:
  - PropUB-Pre: apply (PropUB) only in presolving;
  - PropUB: apply (PropUB) every time propagation is called;
  - PropTB: apply Lemma 10 every time propagation is called.

Furthermore, we use the following combinations of presolving routines:

- nopresol: none;
- MIX1: DZI, TB-Pre, 2MV, PropUB-Pre, PropUB, PropTB;
- MIX2: DGZ, DZI, PropUB-Pre, PropUB.
- allpresol: all routines are activated in presolving, but not in propagation, i.e., PropUB, PropTB are deactivated;
- allprop: PropUB, PropUB-Pre, PropTB, TB-Pre;
- allprop-DGZ: DGZ, TB-Pre, PropUB, PropUB-Pre, PropTB;
- allpresol-prop: all routines are activated in presolving and in propagation;

Note that "MIX1" is the default setting for SCIP-SDP 4.0.0 when using the SDP-based approach. If there is no additional prefix, then the SDP-based approach is used for solving the MISDPs. The prefixes "LPA" and "LPE" denote that the LP-based cutting-plane approach is used instead of the SDP-based approach, in the following two variants: In "LPA", eigenvector cuts are separated, and in "LPE", eigenvector cuts are only enforced, see Section 1.1. For the settings "MIX1-NoCA" and "LPA-MIX2-NoCA" we additionally deactivated conflict analysis. Finally, we also used the concurrent mode of SCIP, where the instances are solved in parallel with settings "MIX1" and "LPA-MIX2", and solving stops, once the first setting reports an optimal solution. Note that our settings "LPA-DGZ" and "LPE-DGZ" roughly correspond to the branch-and-cut algorithm and the cutting-plane algorithm from [36], respectively.

5.3 Results

Table 1 shows the results using the described testset for various settings listed in Section 5.2. Shown are the number of instances that were solved to optimality within the time limit of one hour out of all 185 instances (# Opt), the number of instances which ran into the time limit (# Limit), and the shifted geometric means[1] of the number of nodes (# Nodes) as well as the CPU time in seconds (Time). The next columns list the shifted geometric mean of the CPU time in seconds used for presolving (Time), the arithmetic mean of the number of domain reductions (# Reds), i.e., changed bounds, and added constraints (# AddCons) in presolving for SDP-constraints. The section "SDP Constraints" in Table 1 shows the arithmetic means of the number of propagation calls (# Prop), domain reductions (# Reds), applied cuts (# Cuts) and cutoffs (# CutOff) from SDP-constraints. The last section "SDP Timings" shows the shifted geometric means of the the total time (Total) and the propagation time (Prop) spent for SDP-constraints. For the shifted geometric means, we used a shift of $s = 100$ nodes and $s = 1$ seconds for time, respectively. Tables 2–6 present the results for each class of instances. Note that when comparing the number of used nodes for two settings, we only take into account instances which have been solved to optimality by both settings, whereas the numbers in Tables 1–6 also take into account instances which ran into the time limit.

First of all, it turns out that Constraints (2MP) and coefficient tightening in Lemma 17 (TM), as well as bound tightening in Lemma 10 (TB-Pre) were never active in presolving throughout our testset. All other routines added constraints and/or changed bounds in presolving and produced domain reductions deeper within the branch-and-bound tree. In comparison with the setting "nopresol" in which all presolving routines are deactivated, adding the constraints (DGZ) or (2ML) has a negative effect on the running time, whereas adding the constraints (DZI) results in a speed-up of about 5 %. The latter is in line with the results reported by Mars [43] and Gally [28]. Using Lemma 7, i.e., adding (2MV) in presolving yields a minor improvement of the overall running time. Using Lemma 5 in propagation and/or in presolving (PropUB, PropUB-Pre) also speeds up the solution process by 6 % and reduces the number of used nodes by 11 %. The highest impact of all routines alone is achieved by using bound tightening from Lemma 10 in propagation (PropTB), resulting in a 15 % reduction of the solution time. Interestingly, it solves one instance less than using no presolving at all. Using all presolving routines (allpresol) yields only a minor further improvement over the best pure presolving routine (DZI). If all propagation methods are activated as well (allpresol-prop), we obtain a major improvement in terms of overall running time (13 % faster) and processed nodes (28 % fewer nodes). Using only bound tightening and propagation (allprop) results in a further speed-up, and using the combination MIX1 turns out to be the best setting in terms of overall running times, which is about 22 % faster and processes about 23 % fewer nodes than using no presolving.

We also conducted experiments where the optimal objective value was set as objective limit and all primal heuristics are turned off in order to remove the impact of primal solutions. In this case, propagation via PropUB and PropTB reduces the number of nodes by 9 % and 10 %, respectively, compared to using no presolving or propagation (nopresol). Activating all propagation routines (allprop) results in a decrease of the number of nodes of 19 %. The propagation routines typically cut off nodes deeper in the tree. Thus, the speed-up of the solution process when using propagation routines can at least partly be explained by the fact that fewer nodes are needed to close the gap between the dual bound and the optimal (primal) objective value.

For all considered settings, the time spent for presolving or propagation is neglectable, so that all routines presented in this paper can safely be activated without needing a significant

[1] The shifted geometric mean of values $t_1, \ldots, t_n$ is defined as $\left(\prod_{i=1}^{n}(t_i+s)\right)^{1/n} - s$, where s is an appropriate shift.

Table 1: Comparison of presolving routines with SDP- and LP-solving: Given are the number of instances solved to optimality (# Opt), the number of instances which ran into the time limit (# Limit), the shifted geometric means of the number of branch-and-bound nodes (# Nodes) and CPU time in seconds (Time), the shifted geometric mean of the CPU time in seconds used for presolving SDP-constraints (Presolving Time), the arithmetic means of the number of domain reductions in presolving (# Reds), the added constraints in presolving (# AddCons), as well as the arithmetic means of the number of propagation calls from SDP-constraints (# Prop), domain reductions from SDP-constraints (# Reds), applied cuts from SDP-constraints (# Cuts) and cutoffs from SDP-constraints (# CutOff), and the shifted geometric means of the the total time (Total) and the propagation time (Prop) spent for SDP-constraints.

					SDP Presolving			SDP Constraints				SDP Timings	
Setting	#Opt	#Limit	#Nodes	Time	Time	#Reds	#AddCons	#Prop	#Reds	#Cuts	#CutOff	Total	Prop
nopresol	168|185	17	1405.3	180.23	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.25	0.08
DGZ	168|185	17	1395.9	187.41	0.00	11.0	13.6	0.0	0.0	0.0	0.0	0.25	0.08
DZI	168|185	17	1313.7	171.47	0.00	0.0	0.7	0.0	0.0	0.0	0.0	0.23	0.07
TM	168|185	17	1404.6	180.63	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.26	0.09
TB-Pre	167|185	18	1403.3	180.86	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.26	0.09
2ML	168|185	17	1388.0	184.19	0.02	0.0	940.6	0.0	0.0	0.0	0.0	0.25	0.08
2MP	168|185	17	1404.6	180.23	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.25	0.08
2MV	167|185	18	1373.6	177.54	0.05	0.0	10 083.2	0.0	0.0	0.0	0.0	0.25	0.08
PropUB-Pre	168|185	17	1246.2	168.73	0.01	494.4	0.0	0.0	0.0	0.0	0.0	0.25	0.08
PropUB	168|185	17	1246.2	169.08	0.01	494.4	0.0	649 386.8	0.1	0.0	0.0	0.75	0.54
PropTB	167|185	18	1297.6	152.43	0.00	0.0	0.0	24 418.5	3378.6	0.0	330.5	1.62	1.57
MIX1	167|185	18	1085.2	139.52	0.06	494.4	10 083.9	443 191.6	2211.1	0.0	148.2	2.73	2.69
MIX1-NoCA	167|185	18	1083.7	139.31	0.06	494.4	10 083.9	444 250.1	2224.3	0.0	149.3	2.68	2.64
MIX2	168|185	17	1152.2	166.72	0.01	505.4	14.4	526 541.0	1365.6	0.0	0.0	0.70	0.51
allpresol	168|185	17	1176.8	168.12	0.08	505.4	11 038.2	0.0	0.0	0.0	0.0	0.22	0.07
allprop	166|185	19	1050.2	156.29	0.01	494.4	0.0	209 689.0	4135.1	0.0	2988.7	4.54	4.50
allprop-DGZ	166|185	19	1039.3	164.10	0.01	505.4	13.6	205 861.1	5427.9	0.0	2969.1	4.72	4.67
allpresol-prop	168|185	17	984.1	156.01	0.08	505.4	11 038.2	181 104.9	5612.6	0.0	1865.4	4.39	4.35
LPA-nopresol	104|185	81	386.5	346.38	0.00	0.0	0.0	0.0	0.0	46 014.9	108.7	4.24	0.01
LPA-DGZ	103|185	82	388.9	361.43	0.00	11.0	13.6	0.0	0.0	46 317.6	109.4	4.35	0.02
LPA-DZI	99|185	86	363.9	332.25	0.00	0.0	0.7	0.0	0.0	45 825.5	46.1	4.16	0.01
LPA-TM	104|185	81	387.0	347.25	0.00	0.0	0.0	0.0	0.0	45 977.5	108.7	4.28	0.02
LPA-TB-Pre	101|185	84	389.8	350.96	0.00	0.0	0.0	0.0	0.0	47 209.5	113.7	4.31	0.02
LPA-2ML	103|185	82	386.7	348.31	0.02	0.0	940.6	0.0	0.0	46 126.8	108.7	4.31	0.01
LPA-2MP	103|185	82	387.1	348.52	0.01	0.0	0.0	0.0	0.0	46 023.2	108.7	4.29	0.01
LPA-2MV	101|185	84	369.1	355.30	0.05	0.0	10 083.2	0.0	0.0	44 719.7	45.3	4.27	0.01
LPA-PropUB-Pre	103|185	82	381.3	347.90	0.01	494.4	0.0	0.0	0.0	46 073.0	108.7	4.26	0.01
LPA-PropUB	103|185	82	381.3	348.54	0.01	494.4	0.0	78 845.4	0.0	46 190.6	108.9	4.49	0.10
LPA-PropTB	103|185	82	372.0	469.86	0.00	0.0	0.0	15 194.2	4708.4	49 122.7	416.8	9.48	2.55
LPA-MIX1	101|185	84	335.0	414.18	0.06	494.4	10 083.9	67 464.5	3490.7	42 739.7	152.6	8.18	2.13
LPA-MIX2	98|185	87	360.4	345.39	0.01	505.4	14.4	77 337.6	0.0	46 421.8	47.2	4.52	0.10
LPA-MIX2-NoCA	102|185	83	352.7	329.60	0.01	505.4	14.4	64 138.4	0.0	41 606.0	52.7	4.37	0.09
LPA-allpresol	99|185	86	367.0	353.88	0.08	505.4	11 038.2	0.0	0.0	45 664.9	48.3	4.32	0.01
LPA-allprop	105|185	80	357.0	564.16	0.01	494.4	0.0	65 408.9	3194.6	39 697.9	907.7	15.34	4.11
LPA-allpresol-prop	99|185	86	327.8	641.33	0.08	505.4	11 038.2	57 647.1	4215.2	42 549.8	885.8	17.54	5.20
LPE-MIX2	84|185	101	91 093.1	622.82	0.01	505.4	14.4	1 403 044.8	89 356.6	150 384.9	7665.8	22.71	2.36
CONC: MIX1+LPA-MIX2	160|185	25	702.0	133.20	0.01	505.4	14.4	89.7	0.0	0.0	0.0	0.00	0.00

Table 2: SDP Presolving CLS.

					SDP Presolving			SDP Constraints				SDP Timings	
Setting	#Opt	#Limit	#Nodes	Time	Time	#Reds	#AddCons	#Prop	#Reds	#Cuts	#CutOff	Total	Prop
nopresol	41|43	2	382.5	201.05	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.01
DGZ	41|43	2	362.4	206.48	0.00	1.0	0.0	0.0	0.0	0.0	0.0	0.03	0.01
DZI	41|43	2	382.3	200.63	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.01
TM	41|43	2	382.5	201.18	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.01
TB-Pre	41|43	2	382.4	200.02	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.01
2ML	41|43	2	382.5	200.80	0.03	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.01
2MP	41|43	2	382.2	200.59	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.01
2MV	41|43	2	381.1	206.32	0.15	0.0	39 672.1	0.0	0.0	0.0	0.0	0.03	0.01
PropUB-Pre	41|43	2	382.7	201.29	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.01
PropUB	41|43	2	382.6	200.65	0.01	0.0	0.0	36 176.0	0.0	0.0	0.0	0.05	0.02
PropTB	41|43	2	334.2	99.87	0.00	0.0	0.0	3813.9	6.9	0.0	0.2	1.45	1.44
MIX1	41|43	2	334.1	111.51	0.16	0.0	39 672.1	35 818.3	7.3	0.0	0.3	2.21	2.21
MIX1-NoCA	41|43	2	334.1	111.45	0.16	0.0	39 672.1	35 830.5	7.3	0.0	0.3	2.21	2.19
MIX2	41|43	2	362.4	206.34	0.02	1.0	0.0	30 259.4	0.0	0.0	0.0	0.04	0.02
allpresol	41|43	2	380.6	205.72	0.19	1.0	39 672.1	0.0	0.0	0.0	0.0	0.03	0.01
allprop	41|43	2	334.0	110.15	0.01	0.0	0.0	35 772.5	6.9	0.0	0.7	3.12	3.11
allprop-DGZ	41|43	2	317.2	118.62	0.01	1.0	0.0	30 023.5	7.4	0.0	0.7	3.65	3.63
allpresol-prop	41|43	2	334.3	121.31	0.19	1.0	39 672.1	35 934.0	7.3	0.0	0.8	3.66	3.65
LPA-nopresol	43|43	0	201.1	7.53	0.00	0.0	0.0	0.0	0.0	3197.3	2.0	3.65	0.00
LPA-DGZ	43|43	0	207.2	9.08	0.01	1.0	0.0	0.0	0.0	2992.2	4.8	3.90	0.00
LPA-DZI	43|43	0	201.1	7.70	0.00	0.0	0.0	0.0	0.0	3197.3	2.0	3.70	0.00
LPA-TM	43|43	0	201.1	7.56	0.00	0.0	0.0	0.0	0.0	3197.3	2.0	3.71	0.00
LPA-TB-Pre	43|43	0	201.1	7.76	0.00	0.0	0.0	0.0	0.0	3197.3	2.0	3.72	0.00
LPA-2ML	43|43	0	201.1	7.69	0.03	0.0	0.0	0.0	0.0	3197.3	2.0	3.73	0.00
LPA-2MP	43|43	0	201.1	7.68	0.01	0.0	0.0	0.0	0.0	3197.3	2.0	3.74	0.00
LPA-2MV	43|43	0	214.5	10.53	0.15	0.0	39 672.1	0.0	0.0	3246.3	4.7	4.05	0.00
LPA-PropUB-Pre	43|43	0	201.1	7.69	0.01	0.0	0.0	0.0	0.0	3197.3	2.0	3.71	0.00
LPA-PropUB	43|43	0	201.1	7.68	0.01	0.0	0.0	4036.7	0.0	3197.3	2.0	3.71	0.00
LPA-PropTB	43|43	0	211.3	28.09	0.00	0.0	0.0	901.7	10.3	3100.8	2.1	18.05	9.89
LPA-MIX1	43|43	0	197.8	21.75	0.16	0.0	39 672.1	5643.1	25.7	3188.0	5.3	10.32	4.95
LPA-MIX2	43|43	0	207.2	9.08	0.02	1.0	0.0	6658.1	0.0	2992.2	4.8	3.94	0.01
LPA-MIX2-NoCA	43|43	0	198.6	8.89	0.02	1.0	0.0	6009.3	0.0	2815.1	3.7	3.93	0.00
LPA-allpresol	43|43	0	211.4	10.57	0.20	1.0	39 672.1	0.0	0.0	3249.3	4.6	4.06	0.00
LPA-allprop	43|43	0	197.8	48.07	0.01	0.0	0.0	3974.3	13.6	3059.2	15.9	31.38	10.19
LPA-allpresol-prop	41|43	2	183.6	100.89	0.19	1.0	39 672.1	5564.5	16.4	3107.2	10.9	64.23	29.34
LPE-MIX2	32|43	11	19 781.8	106.76	0.02	1.0	0.0	1 447 705.2	0.0	103 867.4	19 388.7	31.46	0.51
CONC: MIX1+LPA-MIX2	43|43	0	181.3	46.04	0.02	1.0	0.0	100.0	0.0	0.0	0.0	0.00	0.00

Table 3: SDP Presolving MKP.

					SDP Presolving			SDP Constraints				SDP Timings	
Setting	#Opt	#Limit	#Nodes	Time	Time	#Reds	#AddCons	#Prop	#Reds	#Cuts	#CutOff	Total	Prop
nopresol	32|32	0	181.5	63.23	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.05	0.03
DGZ	32|32	0	181.5	63.23	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.03
DZI	32|32	0	181.5	63.23	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.03
TM	32|32	0	181.5	63.20	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.03
TB-Pre	32|32	0	181.5	63.18	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.02
2ML	32|32	0	181.5	63.22	0.02	0.0	110.6	0.0	0.0	0.0	0.0	0.04	0.02
2MP	32|32	0	181.5	63.19	0.02	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.02
2MV	32|32	0	181.5	63.42	0.03	0.0	2052.2	0.0	0.0	0.0	0.0	0.04	0.02
PropUB-Pre	32|32	0	181.5	63.33	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.03
PropUB	32|32	0	181.5	64.14	0.01	0.0	0.0	209 081.2	0.0	0.0	0.0	0.89	0.88
PropTB	32|32	0	181.5	63.18	0.00	0.0	0.0	499.4	0.0	0.0	0.0	0.04	0.03
MIX1	32|32	0	181.5	63.98	0.04	0.0	2052.2	209 081.2	0.0	0.0	0.0	0.91	0.90
MIX1-NoCA	32|32	0	182.0	64.68	0.04	0.0	2052.2	209 119.7	0.0	0.0	0.0	0.90	0.89
MIX2	32|32	0	181.5	63.86	0.01	0.0	0.0	209 081.2	0.0	0.0	0.0	0.89	0.88
allpresol	32|32	0	181.5	63.48	0.07	0.0	2162.9	0.0	0.0	0.0	0.0	0.04	0.03
allprop	32|32	0	181.5	63.94	0.01	0.0	0.0	209 081.2	0.0	0.0	0.0	0.88	0.87
allprop-DGZ	32|32	0	181.5	64.01	0.01	0.0	0.0	209 081.2	0.0	0.0	0.0	0.90	0.89
allpresol-prop	32|32	0	181.5	63.92	0.08	0.0	2162.9	209 081.2	0.0	0.0	0.0	0.88	0.87
LPA-nopresol	5|32	27	67.3	2737.16	0.00	0.0	0.0	0.0	0.0	24 934.4	0.0	1.42	0.00
LPA-DGZ	5|32	27	67.3	2740.72	0.00	0.0	0.0	0.0	0.0	24 940.7	0.0	1.42	0.01
LPA-DZI	4|32	28	67.5	2734.94	0.00	0.0	0.0	0.0	0.0	24 995.8	0.0	1.42	0.00
LPA-TM	5|32	27	67.3	2734.44	0.00	0.0	0.0	0.0	0.0	24 910.2	0.0	1.43	0.00
LPA-TB-Pre	4|32	28	67.5	2736.70	0.00	0.0	0.0	0.0	0.0	24 988.4	0.0	1.45	0.00
LPA-2ML	5|32	27	67.5	2735.28	0.02	0.0	110.6	0.0	0.0	24 979.4	0.0	1.46	0.00
LPA-2MP	5|32	27	67.5	2735.11	0.02	0.0	0.0	0.0	0.0	25 004.1	0.0	1.45	0.00
LPA-2MV	5|32	27	67.7	2734.81	0.03	0.0	2052.2	0.0	0.0	24 892.2	0.0	1.44	0.00
LPA-PropUB-Pre	4|32	28	67.3	2735.38	0.01	0.0	0.0	0.0	0.0	24 936.4	0.0	1.42	0.01
LPA-PropUB	4|32	28	67.2	2736.93	0.01	0.0	0.0	26 071.9	0.0	24 942.1	0.0	1.77	0.31
LPA-PropTB	5|32	27	67.6	2735.21	0.00	0.0	0.0	194.1	0.0	25 012.2	0.0	1.41	0.00
LPA-MIX1	4|32	28	67.4	2738.49	0.04	0.0	2052.2	26 065.8	0.0	24 915.2	0.0	1.77	0.30
LPA-MIX2	4|32	28	67.3	2737.89	0.01	0.0	0.0	26 046.8	0.0	24 934.2	0.0	1.79	0.29
LPA-MIX2-NoCA	5|32	27	57.5	2408.40	0.01	0.0	0.0	24 964.9	0.0	22 153.9	0.0	1.61	0.28
LPA-allpresol	4|32	28	67.5	2736.88	0.07	0.0	2162.9	0.0	0.0	24 965.7	0.0	1.44	0.00
LPA-allprop	5|32	27	67.4	2735.64	0.01	0.0	0.0	26 083.1	0.0	24 958.2	0.0	1.80	0.30
LPA-allpresol-prop	5|32	27	67.3	2739.24	0.08	0.0	2162.9	26 056.8	0.0	24 913.6	0.0	1.77	0.30
LPE-MIX2	1|32	31	731 548.2	3074.35	0.01	0.0	0.0	2 856 042.9	0.0	62 998.1	0.0	30.51	27.70
CONC: MIX1+LPA-MIX2	29|32	3	168.8	93.52	0.01	0.0	0.0	100.0	0.0	0.0	0.0	0.00	0.00

Table 4: SDP Presolving RIP.

Setting	#Opt	#Limit	#Nodes	Time	SDP Presolving Time	SDP Presolving #Reds	SDP Presolving #AddCons	SDP Constraints #Prop	SDP Constraints #Reds	SDP Constraints #Cuts	SDP Constraints #CutOff	SDP Timings Total	SDP Timings Prop
nopresol	36|46	10	4376.2	259.70	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.08	0.04
DGZ	36|46	10	4443.0	296.45	0.00	43.3	0.0	0.0	0.0	0.0	0.0	0.08	0.05
DZI	36|46	10	4376.3	259.95	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.08	0.05
TM	36|46	10	4369.4	260.17	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.09	0.05
TB-Pre	36|46	10	4376.1	259.73	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.08	0.05
2ML	36|46	10	4173.6	281.78	0.02	0.0	994.1	0.0	0.0	0.0	0.0	0.07	0.04
2MP	36|46	10	4371.5	259.52	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.07	0.05
2MV	36|46	10	4376.2	261.18	0.04	0.0	1988.3	0.0	0.0	0.0	0.0	0.08	0.05
PropUB-Pre	36|46	10	2755.9	198.09	0.01	1988.3	0.0	0.0	0.0	0.0	0.0	0.07	0.04
PropUB	36|46	10	2756.7	199.05	0.01	1988.3	0.0	184 929.1	0.2	0.0	0.0	1.58	1.56
PropTB	36|46	10	4377.1	259.67	0.00	0.0	0.0	25 049.6	0.0	0.0	0.0	0.07	0.05
MIX1	36|46	10	2759.5	194.11	0.04	1988.3	1988.3	186 004.4	0.2	0.0	0.0	1.62	1.60
MIX1-NoCA	36|46	10	2761.9	193.71	0.04	1988.3	1988.3	186 254.2	0.2	0.0	0.0	1.58	1.56
MIX2	36|46	10	2757.6	225.43	0.01	2031.5	0.0	183 489.4	5492.2	0.0	0.0	1.58	1.56
allpresol	36|46	10	2768.3	227.89	0.06	2031.5	2982.4	0.0	0.0	0.0	0.0	0.06	0.03
allprop	36|46	10	2755.0	199.16	0.01	1988.3	0.0	184 592.3	0.2	0.0	0.0	1.59	1.57
allprop-DGZ	36|46	10	2755.7	225.17	0.01	2031.5	0.0	183 360.7	5492.2	0.0	0.0	1.59	1.57
allpresol-prop	36|46	10	2757.7	225.37	0.06	2031.5	2982.4	183 484.5	5017.5	0.0	0.2	1.59	1.57
LPA-nopresol	0|46	46	35.7	3600.32	0.00	0.0	0.0	0.0	0.0	12 779.8	0.0	0.61	0.00
LPA-DGZ	0|46	46	36.7	3600.51	0.00	43.3	0.0	0.0	0.0	13 240.9	0.0	0.64	0.00
LPA-DZI	0|46	46	35.9	3600.40	0.00	0.0	0.0	0.0	0.0	12 829.6	0.0	0.60	0.00
LPA-TM	0|46	46	36.3	3600.42	0.00	0.0	0.0	0.0	0.0	12 991.6	0.0	0.64	0.00
LPA-TB-Pre	0|46	46	35.9	3600.58	0.00	0.0	0.0	0.0	0.0	12 895.0	0.0	0.61	0.00
LPA-2ML	0|46	46	37.4	3600.38	0.02	0.0	994.1	0.0	0.0	13 218.0	0.0	0.65	0.00
LPA-2MP	0|46	46	36.2	3600.51	0.00	0.0	0.0	0.0	0.0	12 924.5	0.0	0.63	0.00
LPA-2MV	0|46	46	36.3	3600.70	0.03	0.0	1988.3	0.0	0.0	12 928.0	0.0	0.62	0.00
LPA-PropUB-Pre	0|46	46	30.1	3600.77	0.01	1988.3	0.0	0.0	0.0	13 507.7	0.0	0.62	0.00
LPA-PropUB	0|46	46	29.9	3600.90	0.01	1988.3	0.0	10 355.9	0.0	13 433.2	0.0	0.74	0.10
LPA-PropTB	0|46	46	35.8	3600.48	0.00	0.0	0.0	122.1	0.0	12 870.5	0.0	0.62	0.00
LPA-MIX1	0|46	46	29.5	3600.40	0.04	1988.3	1988.3	10 927.8	0.0	13 673.1	0.0	0.75	0.10
LPA-MIX2	0|46	46	29.9	3600.64	0.01	2031.5	0.0	10 646.2	0.0	13 778.3	0.0	0.77	0.10
LPA-MIX2-NoCA	0|46	46	30.0	3600.70	0.01	2031.5	0.0	10 652.8	0.0	13 777.7	0.0	0.76	0.10
LPA-allpresol	0|46	46	32.2	3600.56	0.06	2031.5	2982.4	0.0	0.0	14 095.3	0.0	0.69	0.00
LPA-allprop	0|46	46	29.8	3600.60	0.01	1988.3	0.0	10 393.3	0.0	13 402.5	0.0	0.74	0.09
LPA-allpresol-prop	0|46	46	32.3	3600.76	0.06	2031.5	2982.4	10 920.9	0.0	14 131.2	0.0	0.81	0.11
LPE-MIX2	33|46	13	41 373.9	366.98	0.01	2031.5	0.0	532 210.1	359 368.7	131 484.5	64.8	7.02	4.27
CONC: MIX1+LPA-MIX2	30|46	16	1371.2	513.69	0.01	2031.5	0.0	79.6	0.0	0.0	0.0	0.00	0.00

Table 5: SDP Presolving RND.

					SDP Presolving			SDP Constraints				SDP Timings	
Setting	#Opt	#Limit	#Nodes	Time	Time	#Reds	#AddCons	#Prop	#Reds	#Cuts	#CutOff	Total	Prop
nopresol	25|26	1	98.4	268.58	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.00
DGZ	25|26	1	98.3	268.85	0.00	0.0	96.9	0.0	0.0	0.0	0.0	0.04	0.00
DZI	25|26	1	98.3	269.73	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.00
TM	25|26	1	98.3	268.78	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.00
TB-Pre	25|26	1	98.3	268.44	0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.04	0.00
2ML	25|26	1	98.3	270.37	0.03	0.0	4797.7	0.0	0.0	0.0	0.0	0.04	0.00
2MP	25|26	1	98.3	268.78	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.00
2MV	25|26	1	98.3	269.02	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.00
PropUB-Pre	25|26	1	98.3	269.05	0.01	0.0	0.0	0.0	0.0	0.0	0.0	0.03	0.00
PropUB	25|26	1	98.3	268.69	0.01	0.0	0.0	2903.5	0.0	0.0	0.0	0.03	0.00
PropTB	25|26	1	98.3	269.00	0.00	0.0	0.0	120.6	0.0	0.0	0.0	0.03	0.00
MIX1	25|26	1	98.3	268.99	0.02	0.0	0.0	2903.5	0.0	0.0	0.0	0.03	0.00
MIX1-NoCA	25|26	1	98.3	269.15	0.02	0.0	0.0	2903.5	0.0	0.0	0.0	0.03	0.00
MIX2	25|26	1	98.4	269.03	0.01	0.0	96.9	2903.6	0.0	0.0	0.0	0.03	0.00
allpresol	25|26	1	98.3	270.26	0.05	0.0	4894.6	0.0	0.0	0.0	0.0	0.03	0.00
allprop	25|26	1	98.4	268.81	0.01	0.0	0.0	2903.6	0.0	0.0	0.0	0.03	0.00
allprop-DGZ	25|26	1	98.4	268.64	0.01	0.0	96.9	2903.6	0.0	0.0	0.0	0.03	0.00
allpresol-prop	25|26	1	98.3	270.46	0.05	0.0	4894.6	2903.5	0.0	0.0	0.0	0.03	0.00
LPA-nopresol	26|26	0	99.8	413.63	0.00	0.0	0.0	0.0	0.0	41 561.1	0.0	104.20	0.00
LPA-DGZ	25|26	1	97.6	423.27	0.00	0.0	96.9	0.0	0.0	43 599.4	0.0	105.26	0.00
LPA-DZI	26|26	0	99.8	409.12	0.00	0.0	0.0	0.0	0.0	41 561.1	0.0	102.85	0.00
LPA-TM	26|26	0	99.8	412.75	0.00	0.0	0.0	0.0	0.0	41 561.1	0.0	103.56	0.00
LPA-TB-Pre	26|26	0	99.8	412.29	0.00	0.0	0.0	0.0	0.0	41 561.1	0.0	103.50	0.00
LPA-2ML	25|26	1	96.0	417.89	0.03	0.0	4797.7	0.0	0.0	42 159.3	0.0	102.70	0.00
LPA-2MP	26|26	0	99.8	411.54	0.00	0.0	0.0	0.0	0.0	41 561.1	0.0	103.81	0.00
LPA-2MV	26|26	0	99.8	412.07	0.01	0.0	0.0	0.0	0.0	41 561.1	0.0	103.29	0.00
LPA-PropUB-Pre	26|26	0	99.8	413.21	0.00	0.0	0.0	0.0	0.0	41 561.1	0.0	103.36	0.00
LPA-PropUB	26|26	0	99.8	413.63	0.00	0.0	0.0	2193.6	0.0	41 561.1	0.0	104.06	0.00
LPA-PropTB	26|26	0	99.8	412.04	0.00	0.0	0.0	161.3	0.0	41 561.1	0.0	103.65	0.00
LPA-MIX1	26|26	0	99.8	412.45	0.01	0.0	0.0	2193.6	0.0	41 561.1	0.0	103.39	0.00
LPA-MIX2	25|26	1	97.6	422.87	0.01	0.0	96.9	2064.0	0.0	43 527.3	0.0	105.44	0.00
LPA-MIX2-NoCA	25|26	1	98.6	418.66	0.01	0.0	96.9	1828.1	0.0	44 157.1	0.0	106.42	0.00
LPA-allpresol	25|26	1	95.4	419.32	0.05	0.0	4894.6	0.0	0.0	40 530.2	0.0	101.47	0.00
LPA-allprop	26|26	0	99.8	412.99	0.00	0.0	0.0	2193.6	0.0	41 561.1	0.0	103.68	0.00
LPA-allpresol-prop	25|26	1	95.4	418.40	0.05	0.0	4894.6	2191.8	0.0	40 589.2	0.0	101.18	0.00
LPE-MIX2	7|26	19	37 491.6	2232.65	0.01	0.0	96.9	325 125.5	0.0	78 329.4	0.0	133.61	0.09
CONC: MIX1+LPA-MIX2	25|26	1	67.6	179.92	0.01	0.0	96.9	75.0	0.0	0.0	0.0	0.00	0.00

Table 6: SDP Presolving TTD.

Setting	#Opt	#Limit	#Nodes	Time	SDP Presolving Time	SDP Presolving #Reds	SDP Presolving #AddCons	SDP Constraints #Prop	SDP Constraints #Reds	SDP Constraints #Cuts	SDP Constraints #CutOff	SDP Timings Total	SDP Timings Prop
nopresol	34|38	4	23 840.6	187.36	0.00	0.0	0.0	0.0	0.0	0.0	0.0	1.46	0.36
DGZ	34|38	4	23 839.7	187.16	0.00	0.0	0.0	0.0	0.0	0.0	0.0	1.48	0.34
DZI	34|38	4	17 547.3	146.73	0.00	0.0	3.6	0.0	0.0	0.0	0.0	1.23	0.26
TM	34|38	4	23 840.5	188.81	0.00	0.0	0.0	0.0	0.0	0.0	0.0	1.50	0.36
TB-Pre	33|38	5	23 700.8	191.85	0.00	0.0	0.0	0.0	0.0	0.0	0.0	1.52	0.38
2ML	34|38	4	23 842.6	188.11	0.00	0.0	0.0	0.0	0.0	0.0	0.0	1.49	0.34
2MP	34|38	4	23 839.8	188.01	0.00	0.0	0.0	0.0	0.0	0.0	0.0	1.45	0.34
2MV	33|38	5	21 561.9	167.33	0.00	0.0	62.1	0.0	0.0	0.0	0.0	1.43	0.32
PropUB-Pre	34|38	4	23 841.9	188.00	0.00	0.0	0.0	0.0	0.0	0.0	0.0	1.47	0.34
PropUB	34|38	4	23 841.3	187.69	0.00	0.0	0.0	2 718 636.0	0.0	0.0	0.0	1.58	0.50
PropTB	33|38	5	18 695.5	182.75	0.00	0.0	0.0	83 737.6	16 440.6	0.0	1609.0	33.24	32.58
MIX1	33|38	5	14 397.8	147.75	0.00	0.0	65.7	1 713 894.0	10 756.0	0.0	721.2	27.57	27.05
MIX1-NoCA	33|38	5	14 274.5	145.71	0.00	0.0	65.7	1 718 698.6	10 820.2	0.0	726.7	26.68	26.15
MIX2	34|38	4	17 547.1	146.42	0.00	0.0	3.6	2 129 008.7	0.0	0.0	0.0	1.31	0.40
allpresol	34|38	4	18 398.9	151.30	0.00	0.0	65.7	0.0	0.0	0.0	0.0	1.27	0.30
allprop	32|38	6	12 454.4	252.35	0.00	0.0	0.0	578 865.9	20 123.6	0.0	14 549.5	151.89	151.85
allprop-DGZ	32|38	6	12 428.8	253.52	0.00	0.0	0.0	568 226.6	19 768.7	0.0	14 453.9	152.92	152.88
allpresol-prop	34|38	4	9293.6	192.47	0.00	0.0	65.7	440 865.0	21 242.3	0.0	9080.5	115.26	115.22
LPA-nopresol	30|38	8	17 646.4	210.55	0.00	0.0	0.0	0.0	0.0	155 497.6	527.1	5.20	0.07
LPA-DGZ	30|38	8	17 635.4	210.67	0.00	0.0	0.0	0.0	0.0	155 245.6	527.1	5.23	0.07
LPA-DZI	26|38	12	13 931.7	169.27	0.00	0.0	3.6	0.0	0.0	154 463.6	222.1	4.73	0.05
LPA-TM	30|38	8	17 628.5	212.69	0.00	0.0	0.0	0.0	0.0	155 079.4	527.1	5.18	0.07
LPA-TB-Pre	28|38	10	18 168.5	218.12	0.00	0.0	0.0	0.0	0.0	161 128.7	551.1	5.44	0.07
LPA-2ML	30|38	8	17 620.8	210.28	0.00	0.0	0.0	0.0	0.0	155 065.0	527.1	5.20	0.06
LPA-2MP	29|38	9	17 637.8	213.57	0.00	0.0	0.0	0.0	0.0	155 304.2	527.1	5.21	0.06
LPA-2MV	27|38	11	13 937.4	169.64	0.00	0.0	62.1	0.0	0.0	148 992.6	215.3	4.73	0.07
LPA-PropUB-Pre	30|38	8	17 617.8	210.71	0.00	0.0	0.0	0.0	0.0	154 897.7	527.1	5.18	0.07
LPA-PropUB	30|38	8	17 650.9	212.82	0.00	0.0	0.0	343 292.3	0.0	155 555.5	527.7	5.20	0.11
LPA-PropTB	29|38	9	14 602.5	231.70	0.00	0.0	0.0	72 529.8	22 910.9	170 561.6	2026.9	35.27	30.97
LPA-MIX1	28|38	10	10 918.4	165.25	0.00	0.0	65.7	285 380.5	16 965.0	138 497.7	736.9	27.01	23.68
LPA-MIX2	26|38	12	13 983.5	168.97	0.00	0.0	3.6	332 744.0	0.0	155 157.1	224.2	4.81	0.11
LPA-MIX2-NoCA	29|38	9	13 935.7	154.24	0.00	0.0	3.6	270 283.5	0.0	133 823.3	252.4	4.36	0.10
LPA-allpresol	27|38	11	14 548.4	163.72	0.00	0.0	65.7	0.0	0.0	152 821.3	230.0	4.75	0.06
LPA-allprop	31|38	7	13 857.6	311.69	0.00	0.0	0.0	277 894.1	15 537.3	124 126.5	4400.9	139.89	131.06
LPA-allpresol-prop	28|38	10	10 525.4	251.53	0.00	0.0	65.7	237 691.5	20 502.8	137 776.4	4300.1	114.22	106.39
LPE-MIX2	11|38	27	421 128.0	938.85	0.00	0.0	3.6	1 920 622.5	0.0	348 792.6	15 301.8	13.80	0.70
CONC: MIX1+LPA-MIX2	33|38	5	9125.8	93.54	0.00	0.0	3.6	91.5	0.0	0.0	0.0	0.00	0.00

amount of time by themselves.

In case of the LP-based cutting-plane approach, it turns out that DZI is the only setting which improves the running times (around 4 % faster), whereas 2MV and propagating the bound tightening (PropTB) have a negative impact. Moreover, only enforcing eigenvector cuts (LPE-MIX2) is clearly much worse than separating them (LPA-MIX2).

Concerning conflict analysis, it turns out that it has almost no impact when using the SDP-based approach, but it negatively influences the performance of the LP-based cutting-plane approach, regardless of the instance class. For the setting MIX2, deactivating conflict analysis results in a speed-up of almost 5 % for the solution time.

Tables 2–6 present the results for each separate instance class. It turns out that for Min-k-Partitioning and random MISDPs, none of the routines has any impact on the performance, even if some constraints are added during presolving. No bounds are changed in presolving and no domain reductions are found deeper in the tree. For Cardinality Least Squares, using bound tightening from Lemma 10 in presolving and propagation (PropTB) reduces the overall running time by almost a factor of 2. Using bound tightening only in presolving (TB-Pre) or using the propagation from Lemma 5 in propagation and/or presolving (PropUB, PropUB-Pre) has almost no impact. For the RIP, the performance impact is switched. Using bound tightening (PropTB, TB-Pre) has no impact, whereas the propagation of Lemma 5 (PropUB, PropUB-Pre) significantly improves the performance; the solution process is about 23 % faster. Finally, for Truss Topology Design, Inequalities (DZI) turn out to be very effective and reduce the solution time by about 22 %, whereas bound tightening and propagation have no impact.

Interestingly, the winner between SDP- and LP-based approach also heavily depends on the instance class. Namely, for Cardinality Least Squares, the LP-based approach is faster by almost a factor 20, whereas for Min-k-Partitioning, the SDP-approach is almost a factor 35 times faster. For random MISDPs and Truss Topology Design, there is not much difference, but the SDP-approach is slightly faster. Lastly, for the RIP, the LP-based approach only solves a single instance within the time limit for the best setting, whereas the SDP-approach solves 36 out of 46. Interestingly, the RIP instances are the only ones for which enforcing eigenvector cuts is significantly faster than separating eigenvector cuts. Using a concurrent solving mode with the best SDP-based setting (MIX1) and the best LP-based setting (LPA-MIX2) yields the best performance overall on the testset, resulting in 41 % fewer processed nodes and a solution process which is 26 % faster than using no presolving at all.

Overall, it turns out that several of the presented methods have a positive impact on the performance of SCIP-SDP, at almost no additional time spent for executing these methods. Most importantly, the inequalities in (DZI), and in Lemma 7 (2MV) should be added during presolving, and the propagation in Lemma 5 as well as the bound tightening from Lemma 10 should be executed both in presolving and in propagation calls deeper in the tree. Depending on the instance, it is beneficial to turn off one or more of these routines to gain improved performance, and to switch to an LP-based approach. By using the concurrent mode with an SDP and LP solving procedure run in parallel, one can exploit this performance difference between SDP- and LP-approach automatically.

6 Conclusions

In this paper, we extended several presolving methods from mixed-integer linear programs to MISDPs and introduced new methods. On our testset, these methods are effective on average with a decrease of about 22 % in running time compared to using no presolving (variant MIX1 vs. nopresol), when applied in the nodes, i.e., propagation is performed in the whole tree. The impact, however, depends on the type of instance. In the extreme, for partitioning instances presolving has no impact at all. For others, (node) presolving implies a performance

improvement of about 25 % (RIP) or even 44 % (CLS), although in the latter case solving LPs is even better with an improvement of at least one order of magnitude between SDP and LP solving. These numbers illustrate again that the effectiveness of presolving depends on the type of application. However, since the methods only cause a negligible runtime increase, they can easily be used or tested on new instance types. This is true, in particular, if more instances are generated by modeling software in the future. Thus, one could conclude the results of this paper as follows: "The presolving methods are effective if they can be applied; and if not, they only impose a very small overhead."

Open questions for future research include the following: Can one derive effective presolving based on larger minors of the positive semidefinite matrices $A(y)$? Can one predict in which cases which presolving method is effective or when switching to LP solving seems advisable? Can perspective reformulation techniques as, e.g., in [5], be automatically applied if indicator constraints are present?

Acknowledgement

We thank three reviewers for detailed comments that helped to improve the presentation of the paper. This work was supported by the EXPRESS II project within the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) priority program CoSIP (DFG-SPP 1798). It was also partly supported by the DFG within Project A4 in the SFB 805. We thank Stefan Ulbrich for suggesting reference [25] and Andreas Schmitt for beta-testing and debugging.

References

[1] T. Achterberg, "Conflict analysis in mixed integer programming," *Discrete Opt.*, vol. 4, no. 1, pp. 4–20, 2007. 10.1016/j.disopt.2006.10.006

[2] ——, "Constraint integer programming," Dissertation, TU Berlin, 2007, http://opus.kobv.de/tuberlin/volltexte/2007/1611/.

[3] T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger, "Presolve reductions in mixed integer programming," *INFORMS J. Comput.*, vol. 32, no. 2, pp. 473–506, 2020. 10.1287/ijoc.2018.0857

[4] T. Achterberg and R. Wunderling, "Mixed integer programming: Analyzing 12 years of progress," in *Facets of Combinatorial Optimization*, M. Jünger and G. Reinelt, Eds. Springer Berlin Heidelberg, 2013, pp. 449–481. 10.1007/978-3-642-38189-8_18

[5] A. Atamtürk and A. Gómez, "Safe screening rules for L0-regression from perspective relaxations," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. Daumé III and A. Singh, Eds., vol. 119. PMLR, 2020, pp. 421–430.

[6] P. Belotti, S. Cafieri, J. Lee, and L. Liberti, "Feasibility-based bounds tightening via fixed points," in *Combinatorial Optimization and Applications – 4th International Conference, COCOA 2010*, ser. Lecture Notes in Computer Science, W. Wu and O. Daescu, Eds., vol. 6508. Springer, 2010, pp. 65–76. 10.1007/978-3-642-17458-2_7

[7] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, "Mixed-integer nonlinear optimization," *Acta Numer.*, vol. 22, pp. 1–131, 2013. 10.1017/S0962492913000032

[8] A. Ben-Tal and A. Nemirovski, "On polyhedral approximations of the second-order cone," *Mathematics of Operations Research*, vol. 26, pp. 193–205, 2001. 10.1287/moor.26.2.193.10561

[9] ——, "Robust truss topology design via semidefinite programming," *SIAM J. Optim.*, vol. 7, no. 4, pp. 991–1016, 1997. 10.1137/S1052623495291951

[10] D. Bertsimas, R. Cory-Wright, and J. Pauphilet, "Solving large-scale sparse PCA to certifiable (near) optimality," *J. Mach. Learn. Res.*, vol. 23, no. 13, pp. 1–35, 2022.

[11] D. Bertsimas and B. Van Parys, "Sparse high-dimensional regression: exact scalable algorithms and phase transitions," *Ann. Statist.*, vol. 48, no. 1, pp. 300–323, 2020. 10.1214/18-AOS1804

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

[13] R. Bixby and E. Rothberg, "Progress in computational mixed integer programming—A look back from the other side of the tipping point," *Ann. Oper. Res.*, vol. 149, pp. 37–41, 2007. 10.1007/s10479-006-0091-y

[14] L. Bordeaux, G. Katsirelos, N. Narodytska, and M. Y. Vardi, "The complexity of integer bound propagation," *J. Artif. Intell. Res.*, vol. 40, pp. 657–676, 2011. 10.1613/jair.3248

[15] S. Boyd and L. Vandenberghe, *Convex Optimization*, 7th ed. Cambridge University Press, 2009.

[16] G. Braun, S. Fiorini, S. Pokutta, and D. Steurer, "Approximation limits of linear programs (beyond hierarchies)," *Math. Oper. Res.*, vol. 40, no. 3, pp. 756–772, 2015. 10.1287/moor.2014.0694

[17] G. Braun, S. Pokutta, and D. Zink, "Affine reductions for LPs and SDPs," *Math. Program.*, vol. 173, pp. 281–312, 2019. 10.1007/s10107-017-1221-9

[18] A. L. Brearley, G. Mitra, and H. P. Williams, "Analysis of mathematical programming problems prior to applying the simplex algorithm," *Math. Program.*, vol. 8, no. 1, pp. 54–83, 1975. 10.1007/BF01580428

[19] C. Coey, M. Lubin, and J. P. Vielma, "Outer approximation with conic certificates for mixed-integer convex problems," *Mathem. Program. Comput.*, vol. 12, pp. 249–293, 2020. 10.1007/s12532-020-00178-3

[20] H. Crowder, E. L. Johnson, and M. Padberg, "Solving large-scale zero-one linear programming problems," *Oper. Res.*, vol. 31, pp. 803–834, 1983. 10.1287/opre.31.5.803

[21] R. J. Dakin, "A tree-search algorithm for mixed integer programming problems," *Comput. J.*, vol. 8, no. 3, pp. 250–255, 1965. 10.1093/comjnl/8.3.250

[22] M. A. Duran and I. E. Grossmann, "An outer-approximation algorithm for a class of mixed-integer nonlinear programs," *Math. Program.*, vol. 36, pp. 307–339, 1986. 10.1007/BF02592064

[23] A. Eisenblätter, "Frequency assignment in GSM networks: Models, heuristics, and lower bounds," Dissertation, TU Berlin, 2001.

[24] ——, "The semidefinite relaxation of the k-partition polytope is strong," in *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization*, ser. Lecture Notes in Computer Science, W. J. Cook and A. S. Schulz, Eds., vol. 2337. Springer, Berlin Heidelberg, 2002, pp. 273–290. 10.1007/3-540-47867-1_20

[25] F. Facchinei and J.-S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems – Volume II*. Springer, 2003.

[26] S. Foucart and H. Rauhut, *A mathematical introduction to compressive sensing*. Birkhäuser Basel, 2013.

[27] C. Fritz, "Some fixed point basics," in *Automata Logics, and Infinite Games: A Guide to Current Research*, E. Grädel, W. Thomas, and T. Wilke, Eds. Berlin, Heidelberg: Springer, 2002, pp. 359–364. 10.1007/3-540-36387-4_20

[28] T. Gally, "Computational mixed-integer semidefinite programming," Dissertation, TU Darmstadt, 2019.

[29] T. Gally and M. E. Pfetsch, "Computing restricted isometry constants via mixed-integer semidefinite programming," Optimization Online, Tech. Rep., 2016, http://www.optimization-online.org/DB_HTML/2016/04/5395.html.

[30] T. Gally, M. E. Pfetsch, and S. Ulbrich, "A framework for solving mixed-integer semidefinite programs," *Optim. Methods Softw.*, vol. 33, no. 3, pp. 594–632, 2017. 10.1080/10556788.2017.1322081

[31] P. Gemander, W.-K. Chen, D. Weninger, L. Gottwald, A. Gleixner, and A. Martin, "Two-row and two-column mixed-integer presolve using hashing-based pairing methods," *EURO J. Comput. Optim.*, vol. 8, pp. 205–240, 2020. 10.1007/s13675-020-00129-6

[32] A. M. Gleixner, T. Berthold, B. Müller, and S. Weltge, "Three enhancements for optimization-based bound tightening," *Journal of Global Optimization*, vol. 67, no. 4, pp. 731–757, 2017. 10.1007/s10898-016-0450-4

[33] C. Helmberg, "Semidefinite programming for combinatorial optimization," Habilitation, TU Berlin, 2000.

[34] N. J. Higham, N. Strabić, and V. Šego, "Restoring definiteness via shrinking, with an application to correlation matrices with a fixed block," *SIAM Rev.*, vol. 58, no. 2, pp. 245–263, 2016. 10.1137/140996112

[35] R. Jiang, D. Li, and B. Wu, "SOCP reformulation for the generalized trust region subproblem via a canonical form of two symmetric matrices," *Math. Program.*, vol. 169, pp. 531–563, 2018. 10.1007/s10107-017-1145-4

[36] K. Kobayashi and Y. Takano, "A branch-and-cut algorithm for solving mixed-integer semidefinite optimization problems," *Comput. Optim. Appl.*, vol. 75, pp. 493–513, 2020. 10.1007/s10589-019-00153-2

[37] K. Krishnan and J. E. Mitchell, "A unifying framework for several cutting plane methods for semidefinite programming," *Optimization Methods and Software*, vol. 21, no. 1, pp. 57–74, 2006.

[38] P. Lancaster and L. Rodman, "Canonical forms for Hermitian matrix pairs under strict equivalence and congruence," *SIAM Rev.*, vol. 47, no. 3, pp. 407–443, 2005. 10.1137/S003614450444556X

[39] Y. Li and W. Xie, "Exact and approximation algorithms for sparse PCA," Optimization Online, Tech. Rep., 2020, http://www.optimization-online.org/DB_HTML/2020/05/7802.html.

[40] J. Löfberg, "YALMIP: a toolbox for modeling and optimization in MATLAB," in *IEEE International Symposium on Computer Aided Control Systems Design*, 2004, pp. 284–289. 10.1109/CACSD.2004.1393890

[41] M. Lubin, E. Yamangil, R. Bent, and J. P. Vielma, "Polyhedral approximation in mixed-integer convex optimization," *Math. Program.*, vol. 172, pp. 139–168, 2018. 10.1007/s10107-017-1191-y

[42] A. Mahajan, "Presolving mixed–integer linear programs," in *Wiley Encyclopedia of Operations Research and Management Science*. American Cancer Society, 2011. 10.1002/9780470400531.eorms0437

[43] S. Mars, "Mixed-integer semidefinite programming with an application to truss topology design," Dissertation, FAU Erlangen-Nürnberg, 2013.

[44] C. J. Nohra, A. U. Raghunathan, and N. Sahinidis, "Spectral relaxations and branching strategies for global optimization of mixed-integer quadratic programs," *SIAM Journal on Optimization*, vol. 31, no. 1, pp. 142–171, 2021. 10.1137/19M1271762

[45] F. Permenter, H. A. Friberg, and E. D. Andersen, "Solving conic optimization problems via self-dual embedding and facial reduction: A unified approach," *SIAM Journal on Optimization*, vol. 27, no. 3, pp. 1257–1282, 2017. 10.1137/15M1049415

[46] F. Permenter and P. A. Parrilo, "Partial facial reduction: simplified, equivalent SDPs via approximations of the PSD cone," *Mathematical Programming*, vol. 171, no. 1, pp. 1–54, 2018. 10.1007/s10107-017-1169-9

[47] ——, "Dimension reduction for semidefinite programs via Jordan algebras," *Mathematical Programming*, vol. 181, no. 1, pp. 51–84, 2020. 10.1007/s10107-019-01372-5

[48] M. Pilanci, M. J. Wainwright, and L. El Ghaoui, "Sparse learning via Boolean relaxations," *Math. Program. Series B*, vol. 151, no. 1, pp. 62–87, 2015. 10.1007/s10107-015-0894-1

[49] T. K. Pong and H. Wolkowicz, "The generalized trust region subproblem," *Comput. Optim. Appl.*, vol. 58, pp. 273–322, 2014. 10.1007/s10589-013-9635-7

[50] Y. Puranik and N. V. Sahinidis, "Domain reduction techniques for global NLP and MINLP optimization," *Constraints*, vol. 22, no. 3, pp. 338–376, 2017. 10.1007/s10601-016-9267-5

[51] L. Qi and J. Sun, "A nonsmooth version of Newton's method," *Math. Program.*, vol. 58, no. 1, pp. 353–367, 1993. 10.1007/BF01581275

[52] M. W. P. Savelsbergh, "Preprocessing and probing techniques for mixed integer programming problems," *ORSA J. Comput.*, vol. 6, no. 4, pp. 445–454, 1994. 10.1287/ijoc.6.4.445

[53] H. D. Sherali and B. M. Fraticelli, "Enhancing RLT relaxations via a new class of semidefinite cuts," *J. Glob. Optim.*, vol. 22, pp. 233–261, 2002. 10.1023/A:1013819515732

[54] N. Strabić, "Theory and algorithms for matrix problems with positive semidefinite constraints," Ph.D. dissertation, University of Manchaster, 2016.

[55] A. Tarski, "A lattice-theoretical fixpoint theorem and its application," *Pac. J. Math.*, vol. 5, pp. 285–309, 1955. 10.2140/pjm.1955.5.285

[56] S. Vigerske, "Decomposition in multistage stochastic programming and a constraint integer programming approach to mixed-integer nonlinear programming," Dissertation, Humboldt-Universität zu Berlin, 2013.

[57] S. Vigerske and A. Gleixner, "SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework," *Optim. Methods Softw.*, vol. 33, no. 3, pp. 563–593, 2018. 10.1080/10556788.2017.1335312

[58] J. Witzig, "Infeasibilitity analysis for MIP," Dissertation, TU Berlin, 2021.

[59] J. Witzig, T. Berthold, and S. Heinz, "Experiments with conflict analysis in mixed integer programming," in *Integration of AI and OR Techniques in Constraint Programming. CPAIOR 2017*, vol. 10335, 2017, pp. 211–222. 10.1007/978-3-319-59776-8_17

A Primal Form of an MISDP

Apart from the so-called "dual" form (1) of an MISDP, one can also consider the corresponding "primal" form:

$$\begin{aligned} \sup \quad & \langle A^0, X\rangle & \\ \text{s.t.} \quad & \langle A^i, X\rangle = b_i & \forall\, i \in [m], \\ & L_{ij} \le X_{ij} \le U_{ij} & \forall\, i,\, j \in [n], \\ & X_{ij} \in \mathbb{Z} & \forall\, i,\, j \in I \times I, \\ & X \succeq 0, & \end{aligned} \tag{14}$$

where $\langle A, B\rangle \coloneqq \sum_{i,j=1}^{n} A(i,j)B(i,j)$ for two $n \times n$ matrices A and B. The bounds are given by $L_{ij} \in \mathbb{R} \cup \{-\infty\}$, $U_{ij} \in \mathbb{R} \cup \{\infty\}$ for all $i,\, j \in [m]$.

We note that (1) and (14) are equivalent: Indeed, starting from (1), one can define $Z = \sum_{i=1}^{n} A^i\, y_i - A^0$. The "primal" variables are

$$X = \begin{pmatrix} Z & 0 \\ 0 & \operatorname{Diag}(y) \end{pmatrix} \in \mathbb{R}^{(n+m)\times(n+m)},$$

where $\operatorname{Diag}(y)$ denotes a diagonal matrix containing y on the diagonal (possibly y has to be split into two nonnegative variables). The n^2 equations $Z = \sum_{i=1}^{n} A^i\, y_i - A^0$ can then be written in the form $\langle B^i, X\rangle = d_i$ for appropriate matrices B^i and scalars d_i, $i \in [n^2]$. Conversely, given (14), using the Gauss algorithm on the equations $\langle A^i, X\rangle = b_i$, one can express the n^2 variables in X using $r \coloneqq n^2 - m$ variables y as $X = \sum_{i=1}^{r} B^i\, y_i - B^0$ with appropriate matrices B^i, $i \in [r]_0$. In both directions, the objective and variable bounds can be chosen appropriately.

These transformations often simplify for particular problems. Moreover, the relaxations of (1) and (14) are dual to each other.

B Instance Classes

The following applications (with the exception of random MISDPs) are described in more detail in Gally [28]. We only very briefly illustrate the structure of the problems.

B.1 Cardinality Constrained Least Squares

Given is a matrix $A \in \mathbb{R}^{m\times d}$, whose rows represent sample points, and a vector $b \in \mathbb{R}^m$, which contains the corresponding measurements. For a fixed sparsity level $k \in \mathbb{N}$, the (regularized) cardinality constrained least squares problem is

$$\inf_{x\in\mathbb{R}^d} \left\{ \tfrac{1}{2}\|Ax-b\|_2^2 + \tfrac{1}{2}\rho\|x\|_2^2 \,:\, \|x\|_0 \le k \right\},$$

where $\|x\|_0$ is the number of nonzeros in x and $\frac{1}{2}\rho\|x\|_2^2$ is a regularization term for given positive $\rho \in \mathbb{R}$. Pilanci et al. [48] showed that this problem is equivalent to the following MISDP:

$$\begin{aligned} \inf \quad & \tau \\ \text{s.t.} \quad & \begin{pmatrix} I_m + \frac{1}{\rho} A \operatorname{Diag}(z) A^\top & b \\ b^\top & \tau \end{pmatrix} \succeq 0, \\ & \sum_{j=1}^{d} z_j \le k,\ z \in \{0,1\}^d, \end{aligned} \tag{CLS}$$

where I_m is the identity matrix of dimension m. We note that these problems can also be written as mixed-integer second-order cone problems (possibly using a perspective formulation). Moreover, [11] present a very effective method to solve an equivalent convex formulation. We nevertheless add CLS instances to our testset, since they have distinctive features and complement the other problem types.

We used a subset of the instances in [28], namely, 19 of the 20 instances based on real-world data and 24 of the 45 randomly generated instances. See [28, Chapter 3.5] for information on the generation of these instances. These instances are completely dense.

B.2 Minimum k-Partitioning

Given is an undirected graph $G = (V, E)$ with n nodes, edge-weights $c\colon E \mapsto \mathbb{R}$, and a positive integer $k \ge 2$. The minimum k-partitioning problem seeks to find a partitioning of $V := \{1, \dots, n\}$ into k sets $V_1, \dots, V_k$ such that

$$\sum_{i=1}^{k} \sum_{e\in E[V_i]} c(e)$$

is minimized. We use an MISDP formulation of Eisenblätter [23, 24]. Define the costs as $C_{ij} := c(\{i,j\})$ for $\{i,j\} \in E$ and $C_{ij} = 0$ otherwise. This leads to the formulation

$$\begin{aligned} \inf \quad & \sum_{1\le i<j\le n} C_{ij}\, Y_{ij} \\ \text{s.t.} \quad & \tfrac{-1}{k-1}\, J + \tfrac{k}{k-1}\, Y \succeq 0, \\ & Y_{ii} = 1,\ Y \succeq 0,\ Y \in \{0,1\}^{n\times n}, \end{aligned} \tag{Min-k}$$

where J is the all-one matrix. Additionally, using node weights $w \in \mathbb{R}^n$, we add the following constraint with lower and upper bounds ℓ and u on the weights of the parts:

$$\ell \le \sum_{j=1}^{n} w_j\, Y_{ij} \le u \qquad \forall\, i \in [n].$$

We use 32 of the 59 instances in [28, Chapter 3.5], which all contain very sparse SDP-constraints, since every matrix A^k only consists of a single nonzero entry.

B.3 Restricted Isometry Property

Given a matrix $A \in \mathbb{R}^{m\times n}$, the s-th restricted isometry constant (RIC) $\delta_s(A)$ is defined as

$$\delta_s(A) \coloneqq \min\{\delta \geq 0 \,:\, (1-\delta)\|x\|_2^2 \leq \|Ax\|_2^2 \leq (1+\delta)\|x\|_2^2 \quad \forall\, x \in \Sigma_s\},$$

where $\Sigma_s = \{x \in \mathbb{R}^n \,:\, \|x\|_0 \leq s\}$. The RIC plays a crucial role in Compressed Sensing, since it can be used to decide when it is possible to reconstruct an unknown sparse vector $x^{(0)} \in \mathbb{R}^n$ from its measurements $Ax^{(0)}$ by solving $\min\{\|x\|_1 \,:\, Ax = Ax^{(0)}\}$, see, e.g., [26]. For the purpose of computing the RIC, it is more convenient to split the RIC into a lower and an upper constant α_s and β_s as follows.

$$\begin{aligned}\alpha_s &\coloneqq \max\{\alpha \geq 0 \,:\, \alpha^2\|x\|_2^2 \leq \|Ax\|_2^2\, \forall\, x \in \Sigma_s\} = \min\{\|Ax\|_2^2 \,:\, \|x\|_2^2 = 1,\, \|x\|_0 \leq s\},\\ \beta_s &\coloneqq \min\{\beta \geq 0 \,:\, \beta^2\|x\|_2^2 \geq \|Ax\|_2^2\, \forall\, x \in \Sigma_s\} = \max\{\|Ax\|_2^2 \,:\, \|x\|_2^2 = 1,\, \|x\|_0 \leq s\},\end{aligned} \tag{15}$$

where $\|x\|_0 \coloneqq |\mathrm{supp}(x)|$. For more details, see [29]. Let $x^\star$ be an optimal solution of either of the two problems and $S = \mathrm{supp}(x^\star)$ be its support with $k \coloneqq \|x^\star\|_0$. Consider the submatrix $A_S \in \mathbb{R}^{m\times k}$ indexed by columns in S. Then $\tilde{A} = A_S^\top A_S \in \mathbb{R}^{k\times k}$ is symmetric positive semidefinite. By the Rayleigh-Ritz theorem (see, e.g., Helmberg [33, Thm. A.0.4]) we have

$$\begin{aligned}\max_{y\in\mathbb{R}^k}\{\|\tilde{A}y\|_2^2 \,:\, \|y\|_2^2 = 1\} &= \lambda_{\max}(\tilde{A})^2,\\ \min_{y\in\mathbb{R}^k}\{\|\tilde{A}y\|_2^2 \,:\, \|y\|_2^2 = 1\} &= \lambda_{\min}(\tilde{A})^2,\end{aligned}$$

i.e., computing the lower and upper RIC as defined in (15) are sparse eigenvalue problems, which are of interest in their own regard. Moreover, the problem of computing the upper RIC β_s is also known as *sparse principal component analysis* (SPCA); see Bertsimas et al. [10] for a tailored approach to SPCA.

These problems can be formulated as

$$\begin{aligned}\max\,/\,\min \quad & \langle A^\top A, X\rangle && \text{(16a)}\\ \text{s.t.} \quad & \mathrm{tr}(X) = 1, && \text{(16b)}\\ & -z_j \leq X_{ij} \leq z_j \quad \text{for } i,\, j \in [n], && \text{(16c)}\\ & \sum_{i=1}^n z_i \leq k, && \text{(16d)}\\ & X \succeq 0, && \text{(16e)}\\ & z \in \{0,1\}^n. && \text{(16f)}\end{aligned}$$

In [29] (and [39] as well as [10]) it is proved that there exists an optimal rank-1 solution $X^\star$. Thus, $X^\star = x^\star(x^\star)^\top$ for some $x^\star \in \mathbb{R}^n$ with $\|x^\star\|_0 \leq k$. Let $S = \mathrm{supp}(x^\star)$. Then $x_S^\star$ is an eigenvector for a maximal eigenvalue of $A_S^\top A_S$.

We use 46 instances which are created similar to the instances in [29, Section 6]. Namely, the following six types of random matrices $A \in \mathbb{R}^{m\times n}$ are used for generating the instances:

- 0 ± 1: $\mathcal{P}(A_{ij} = \sqrt{1/m}) = \mathcal{P}(A_{ij} = -\sqrt{1/m}) = \frac{1}{6}$ and $\mathcal{P}(A_{ij} = 0) = \frac{2}{3}$;
- band: band matrix, entries uniformly in $\{0,1\}$, bandwidths 3, 5, 7, $m = n$;
- bernoulli: A_{ij} uniformly in $\{\pm\sqrt{1/m}\}$;
- binary: A_{ij} uniformly in $\{0,1\}$;

- normal: $A_{ij} \sim \mathcal{N}(0,1)$;
- scaled normal: $A_{ij} \in \mathcal{N}(0, \frac{1}{m})$.

Here $\mathcal{N}(\mu, \sigma^2)$ denotes the normal distribution with parameters μ and σ^2. The sizes (m, n, k) are given by $(m, n, k) \in \{(15, 30, 5), (25, 35, 4), (30, 40, 3), (40, 60, 5)\}$. The band matrix instances are larger with $(m, n, k) \in \{(40, 40, 3), (60, 60, 5), (70, 70, 4)\}$. As in Minimum k-Partitioning, the coefficient matrices A^k only consist of one single nonzero entry, if (16) is written in form (1).

B.4 Random MISDPs

We also consider random instances of the form

$$\sup\{b^\top y \,:\, \sum_{k=1}^{m} A^k y_k - A^0 \succeq 0,\ y \in \{0,1\}^{m_b} \times \mathbb{R}^{m_c}\},$$

where $m = m_b + m_c$ and $A^k \in \mathbb{R}^{n \times n}$ are symmetric matrices for $k \in [m]_0$. These instances are produced in the same way as done by Kobayashi and Takano [36], that is, we choose $y_k \sim \mathcal{U}(\{0,1\})$ for $k \leq m_b$, $y_k \sim \mathcal{U}([0,1])$ for $k > m_b$, $A^k_{ij} \sim \mathcal{U}([-1,1])$ for $k \in [m]$ and $1 \leq i \leq j \leq n$. Here, $\mathcal{U}(C)$ denotes the uniform distribution on the set C. In order to ensure that there exists a feasible solution, we set $A^0 = \sum_{k=1}^{m} A^k y_k - \alpha \mathbb{1}$, $b_k = \langle A^k, \mathbb{1} \rangle$ for $k \in [m]$, and $\alpha \geq 0$. For half of the instances, the matrices A^k are ensured to be positive semidefinite and to have rank 1 by randomly choosing $a^k \sim \mathcal{U}([-1,1]^n)$ and setting $A^k = a^k (a^k)^\top$. The dimension of A^k as well as the numbers of binary variables m_b and continuous variables m_c vary between $\{60, 90, 120\}$. The nonnegative factor is chosen as $\alpha \in \{0.1, 10\}$.

B.5 Truss Topology Optimization

Truss topology optimization seeks truss structures that are stable with minimal total volume. Given is a ground structure, which is specified by a simple directed graph $D = (V, E)$ with n nodes, n_f of which are free, while the remaining nodes are fixed. The goal is to choose cross-sectional areas coming from a discrete set $\mathcal{A}$ for the bars on the edges. The model includes ellipsoidal robustness with respect to uncertain loads on the free nodes in $\{Qf \,:\, \|f\|_2 \leq 1\}$ for some matrix Q, following [9], and uses binary variables for choosing bars, see [43]. This yields the model:

$$\begin{aligned}
\inf \quad & \sum_{e \in E} \ell_e \sum_{a \in \mathcal{A}} a\, x_e^a \\
\text{s.t.} \quad & \begin{pmatrix} 2\tau I & Q^\top \\ Q & A(x) \end{pmatrix} \succeq 0, \\
& \sum_{a \in \mathcal{A}} x_e^a \leq 1 && \forall\, e \in E, \\
& \tau \leq C_{\max}, \\
& x_e^a \in \{0,1\} && \forall\, e \in E,\ a \in \mathcal{A},
\end{aligned} \tag{TT}$$

where I is the identity matrix of appropriate dimension. The binary variables x_e^a choose a bar on edge e with cross-sectional area $a \in \mathcal{A}$. The stiffness matrix $A(x)$ is given by $A(x) = \sum_{e \in E} \sum_{a \in \mathcal{A}} A_e\, a\, x_e^a$ with appropriate matrices A_e. The length of bar $e \in E$ is ℓ_e and $C_{\max}$ provides an upper bound on the compliance (potential energy in the system). We use 38 of the 60 instances in [28, Chapter 3.5].