European Journal of Operational Research 331 (2026) 381–395

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Spanning and splitting: Integer semidefinite programming for the quadratic minimum spanning tree problem

Frank de Meijer [a,*], Melanie Siebenhofer [b], Renata Sotirov [c], Angelika Wiegele [b,d]

[a] Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
[b] Institut für Mathematik, Alpen-Adria-Universität Klagenfurt, Universitätstraße 65–67, 9020, Klagenfurt, Austria
[c] Tilburg University, Department of Econometrics & Operations Research, CentER, 5000 LE, Tilburg, The Netherlands
[d] Universität zu Köln, Weyertal 86–90, 50931, Köln, Germany

ARTICLE INFO

Keywords:
Combinatorial optimization
Spanning trees
Integer semidefinite programming
Algebraic connectivity
Projection methods

ABSTRACT

In the quadratic minimum spanning tree problem (QMSTP) one wants to find the minimizer of a quadratic function over all possible spanning trees of a graph. We present a formulation of the QMSTP as a mixed-integer semidefinite program exploiting the algebraic connectivity of a graph. Based on this formulation, we derive a doubly nonnegative relaxation for the QMSTP and investigate classes of valid inequalities to strengthen the relaxation using the Chvátal-Gomory procedure for mixed-integer conic programming.

Solving the resulting relaxations is out of reach for off-the-shelf software. We therefore develop and implement a version of the Peaceman-Rachford splitting method that allows to compute the new bounds for graphs from the literature. The computational results demonstrate that our bounds significantly improve over existing bounds from the literature in both quality and computation time, in particular for graphs with more than 30 vertices.

This work is further evidence that semidefinite programming is a valuable tool to obtain high-quality bounds for problems in combinatorial optimization, in particular for those that can be modelled as a quadratic problem.

1. Introduction

The quadratic minimum spanning tree problem (QMSTP) is the problem of finding a spanning tree of a connected, undirected graph such that the sum of interaction costs over all pairs of edges in the tree is minimized. The QMSTP was introduced by Assad and Xu (1992) in 1992. The adjacent-only quadratic minimum spanning tree problem (AQMSTP), that is, the QMSTP where the interaction costs of all non-adjacent edge pairs are assumed to be zero, is also introduced in (Assad & Xu, 1992). Assad and Xu proved that both the QMSTP and AQMSTP are strongly $\mathcal{NP}$-hard problems. Interestingly, the QMSTP remains $\mathcal{NP}$-hard even when the cost matrix is of rank one (Punnen, 2001).

There are many existing variants of the QMSTP problem, such as the minimum spanning tree problem with conflict pairs, the quadratic bottleneck spanning tree problem, and the bottleneck spanning tree problem with conflict pairs. For a description of those problems, see e.g., Ćustić et al. (Ćustić et al., 2018). The QMSTP has various applications in telecommunication, transportation, energy and hydraulic networks, see e.g., (Assad & Xu, 1992; Chiang et al., 2007; Chou & Kershenbaum, 1974).

There is a lot of research on lower-bounding approaches and exact algorithms for the QMSTP. The majority of lower bounding approaches for the QMSTP may be classified into Gilmore-Lawler (GL) type bounds (Assad & Xu, 1992; Cordone & Passeri, 2012; Öncan & Punnen, 2010; Rostami & Malucelli, 2015) and reformulation linearization technique (RLT) based bounds (Pereira et al., 2015b; Rostami & Malucelli, 2015). The GL procedure is a well-known approach to construct lower bounds for quadratic binary optimization problems, see e.g., (Gilmore, 1962; Lawler, 1963). The RLT is a method to derive a hierarchy of convex approximations of mixed-integer programming problems (Sherali & Adams, 1998) where integer variables are binary. Lower bounding approaches based on an extended formulation of the minimum spanning tree problem are derived in (Sotirov & Verchére, 2024). For an overview of the above-mentioned lower bounding approaches and their comparison, see e.g., (Sotirov & Verchére, 2024). Semidefinite programming (SDP) lower bounds for the QMSTP are considered in (Guimarães et al., 2020). SDP bounds incorporated in a branch-and-bound algorithm provide the best exact solution approach for the problem up to date (Guimarães et al., 2020). Different exact approaches for solving the QMSTP are considered in (Assad & Xu, 1992; Cordone & Passeri,

* Corresponding author.
E-mail addresses: f.j.j.demeijer@tudelft.nl (F. de Meijer), melanie.siebenhofer@aau.at (M. Siebenhofer), R.Sotirov@tilburguniversity.edu (R. Sotirov), angelika.wiegele@aau.at (A. Wiegele).

https://doi.org/10.1016/j.ejor.2025.10.051
Received 8 October 2024; Accepted 31 October 2025
Available online 5 November 2025

2012; Pereira et al., 2015a,b). For a comparison of various heuristic approaches for solving the QMSTP, see Palubeckis et al. (2010).

In this paper, we derive a compact mixed-integer semidefinite (MISDP) formulation for the QMSTP by exploiting the algebraic connectivity of a tree. Algebraic connectivity was also exploited in (Cvetković et al., 1999; De Meijer and Sotirov, 2024a) to derive ISDP formulations for the traveling salesman problem (TSP) and the quadratic TSP, respectively. This formulation contributes to the recent development in mixed-integer semidefinite programming, revealing that many hard combinatorial optimization problems have compact MISDP formulations (De Meijer and Sotirov, 2024b). We prove that the continuous relaxation of our MISDP formulation is not dominated by the continuous relaxation of the cut-set QMSTP formulation, and vice versa. Further, we derive several classes of valid inequalities for our MISDPs by exploiting the Chvátal-Gomory (CG) procedure for mixed-integer conic programming (Çezik & Iyengar, 2005; De Meijer and Sotirov, 2024a). In particular, we show that the classical cut-set constraints and the first-level RLT constraints are CG cuts. Although these cuts have been considered before for the QMSTP, obtaining them as CG cuts provides a measure for how well the continuous relaxation of our MISDP formulation describes the convex hull of feasible solution vectors. The cut-set constraints are derived from the linear matrix inequality (LMI) that is related to the algebraic connectivity of a tree. The RLT-type constraints are derived using two LMIs from the MISDP formulation of the QMSTP.

We exploit our MISDP formulation for the QMSTP to derive a doubly nonnegative (DNN) relaxation of the problem. DNN relaxations have matrix variables that are both positive semidefinite and nonnegative. Our DNN relaxation for the MISDP does not include the connectivity constraint in the form of a linear matrix inequality, because that constraint has only a small impact on the bound. Moreover, preliminary computational results show that the cut-set constraints also have a small impact on the quality of our doubly nonnegative relaxation of the QMSTP. However, the RLT-type constraints improve the DNN bound. Therefore, we add RLT-type constraints to the DNN relaxation of the QMSTP. Although our relaxation is theoretically dominated by the relaxation of Guimarães et al. (2020), we carefully selected which constraints to include in the relaxation in order to achieve an optimal balance between solution quality and computational efficiency. The resulting relaxation has a large number of constraints, and it is difficult to solve using state-of-the-art interior point methods. Therefore, we design a version of the Peaceman-Rachford splitting method (PRSM) that is able to handle a large number of cutting-planes efficiently. In particular, the PRSM algorithm is adding violated RLT-type inequalities iteratively while using warm-starts. The computational results show that our bounds for the QMSTP outperform bounds from the literature in quality, as well as in computational time required to obtain them. Interestingly, this is true even for the bounds that are theoretically stronger than ours, i.e., the ones of Guimarães et al. (2020). We expect that this has to do with the fact that the stronger relaxation of Guimarães et al. (2020) is hard to solve to high precision, whereas our bounds can be computed with high accuracy, resulting in improved bounds. Hence, our approach shows significant improvement over other methods from the literature, particularly for larger instances, specifically, for graphs with more than 30 vertices.

Notation

The set of $n \times n$ real symmetric matrices is denoted by $\mathcal{S}^n$. The space of symmetric matrices is considered with the trace inner product, which for any $X, Y \in \mathcal{S}^n$ is defined as $\langle X, Y \rangle := \operatorname{tr}(XY)$. The associated norm is the Frobenius norm $\|X\|_F := \sqrt{\operatorname{tr}(XX)}$. The cone of symmetric positive semidefinite matrices of order n is defined as $\mathcal{S}^n_+ := \{X \in \mathcal{S}^n : X \succeq \mathbf{0}\}$. We order the eigenvalues of $X \in \mathcal{S}^n$ as follows $\lambda_1(X) \leq \cdots \leq \lambda_n(X)$. If it is clear from the context to which matrix the eigenvalues relate, we denote eigenvalues by λ_i. The operator $\operatorname{diag} : \mathbb{R}^{n\times n} \to \mathbb{R}^n$ maps a square matrix to a vector consisting of its diagonal elements. The adjoint operator of diag is denoted by $\operatorname{Diag} : \mathbb{R}^n \to \mathbb{R}^{n\times n}$.

We denote by $\mathbf{1}_n$ the vector of all ones of length n, and define $\mathbf{J}_n := \mathbf{1}_n\mathbf{1}_n^\top$. The indicator vector of $S \subseteq V$ is denoted by $\mathbb{1}_S$. The all-zero matrix of order n is denoted by $\mathbf{0}_n$. We use $\mathbf{I}_n$ to denote the identity matrix of order n, while its i-th column is given by $\mathbf{u}_i$. In case the dimension of $\mathbf{1}_n$, $\mathbf{0}_n$, $\mathbf{J}_n$ and $\mathbf{I}_n$ is clear from the context, we omit the subscript.

We define the n-simplex as $\Delta_n := \{x \in \mathbb{R}^p : x \geq \mathbf{0},\ \sum_{i=1}^p x_i = n\}$ and the capped n-simplex as $\bar{\Delta}_n := \{x \in \mathbb{R}^p : \mathbf{0} \leq x \leq \mathbf{1},\ \sum_{i=1}^p x_i = n\}$. By $\mathcal{P}_{\mathcal{M}}$ we denote the projection operator onto the set $\mathcal{M}$. We use $[n]$ to denote the set of integers $\{1, \ldots, n\}$. Throughout the paper, we use the (graph) parameters α and β, which are defined as $\beta = 2\left(1 - \cos\left(\frac{\pi}{n}\right)\right)$ and $\alpha = \frac{\beta}{n}$.

Given a subset $S \subseteq V$ of vertices in a graph $G = (V, E)$, we denote the set of edges with both endpoints in S by $E(S) := \{\{i, j\} \in E : i, j \in S\}$ and the cut induced by S by $\delta(S) := \left\{\{i, j\} \in E : i \in S, j \notin S\right\}$. The Laplacian matrix of a graph G with n vertices and the adjacency matrix A is defined as $L := \operatorname{Diag}(A\mathbf{1}_n) - A$.

2. The quadratic minimum spanning tree problem

In this section, we formally introduce the QMSTP. Let $G = (V, E)$ be a connected, undirected graph with $n = |V|$ vertices and $m = |E|$ edges. Let $Q = (q_{ef}) \in \mathcal{S}^m$ be a matrix of interaction costs between edges of G, where q_{ee} represents the cost of edge e.

The QMSTP can be formulated as the following binary quadratic programming problem:

$$\min_{x \in \mathcal{T}} \sum_{e \in E} \sum_{f \in E} q_{ef} x_e x_f,$$

where $\mathcal{T}$ denotes the set of all spanning trees in G. Each spanning tree in $\mathcal{T}$ is represented by its incidence vector of length m, and therefore

$$\mathcal{T} := \left\{ x \in \{0,1\}^m : \sum_{e \in E} x_e = n - 1,\ \sum_{e \in \delta(S)} x_e \geq 1, \forall S \subsetneq V, S \neq \emptyset \right\}. \tag{1}$$

The constraints of the type

$$\sum_{e \in \delta(S)} x_e \geq 1 \tag{2}$$

are known as the *cut-set constraints*, and they ensure connectivity of a subgraph from $\mathcal{T}$. If Q is a diagonal matrix then the QMSTP reduces to the minimum spanning tree problem that is solvable in polynomial time (Kruskal, 1956; Prim, 1957).

Let us now fix an ordering for the edges $E = \{e_1, \ldots, e_m\}$. For $x \in \mathcal{T}$ define $Y := (y_{ef}) \in \mathcal{S}^m$ such that $y_{ef} = 1$ if $x_e = 1$ and $x_f = 1$, and $y_{ef} = 0$ otherwise. Then, the QMSTP can be formulated as the following mixed-integer programming problem, see (Assad & Xu, 1992):

$$\begin{aligned} &\min \langle Q, Y \rangle \\ &\text{s.t. } \operatorname{diag}(Y) = x,\ Y\mathbf{1}_m = (n-1)x \\ &\quad \mathbf{0} \leq Y \leq \mathbf{J}_m,\ Y \in \mathcal{S}^m,\ x \in \mathcal{T}. \end{aligned} \tag{3}$$

One can verify that the constraints, in combination with the binarity of x, are sufficient to obtain the coupling between Y and x. This result was proven in (Assad & Xu, 1992), and this connection between the vector of variables x and the matrix variable Y is also exploited in this paper. Note that each row in Y is an incidence vector of a tree. We refer to (3) as the *cut-set* formulation of the QMSTP.

3. MISDP Formulation for the QMSTP

Fiedler (1973) defined the algebraic connectivity, $a(G)$, of a graph G as the second smallest eigenvalue of the Laplacian matrix of the graph. It is well-known that the algebraic connectivity is greater than zero if and only if G is a connected graph. In this section, we will exploit the algebraic connectivity of a tree to derive a compact MISDP formulation of the QMSTP. Moreover, we derive two classes of cutting planes resulting from this MISDP formulation, that will be exploited later on.

3.1. Compact MISDP formulation for the QMSTP

We start by showing how a lower bound on the algebraic connectivity of a simple graph can be characterized by a linear matrix inequality.

Proposition 1. *Let G be a simple graph on $n \geq 3$ vertices and L be the Laplacian matrix of G. Moreover, let $\widetilde{\beta} \in \mathbb{R}$, $\widetilde{\beta} \geq 0$, and $\widetilde{\alpha} = \frac{\widetilde{\beta}}{n}$. Then $a(G) \geq \widetilde{\beta}$ if and only if $L + \widetilde{\alpha}\mathbf{J}_n - \widetilde{\beta}\mathbf{I}_n \succeq \mathbf{0}$.*

Proof. Let $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of L, the Laplacian matrix of G. The eigenvalues of $L + \widetilde{\alpha}\mathbf{J}$ are $\widetilde{\beta}$ and $a(G) = \lambda_2 \leq \cdots \leq \lambda_n$. If $a(G) \geq \widetilde{\beta}$, then all eigenvalues of $L + \widetilde{\alpha}\mathbf{J}$ are greater or equal than $\widetilde{\beta}$ and therefore $L + \widetilde{\alpha}\mathbf{J} - \beta\mathbf{I} \succeq \mathbf{0}$. Conversely, if $L + \widetilde{\alpha}\mathbf{J} - \beta\mathbf{I} \succeq \mathbf{0}$, then all eigenvalues of $L + \widetilde{\alpha}\mathbf{J}$ greater or equal to $\widetilde{\beta}$ and therefore $a(G) \geq \widetilde{\beta}$. □

We now apply Proposition 1 to the class of trees. It is known that the algebraic connectivity for trees with $n \geq 3$ vertices lies in the interval between $2\left(1 - \cos\left(\frac{\pi}{n}\right)\right)$ and 1, see e.g., (Grone & Merris, 1990). Here, $2\left(1 - \cos\left(\frac{\pi}{n}\right)\right)$ is the algebraic connectivity of the path graph, and 1 is the algebraic connectivity of the star graph. It is also known that a tree on n vertices has exactly $n-1$ edges. Hence, a tree can be characterized as a connected graph with exactly $n-1$ edges, see also (1). By using $\widetilde{\beta} = 2\left(1 - \cos\left(\frac{\pi}{n}\right)\right)$ in Proposition 1, we obtain the following characterization of spanning trees.

Proposition 2. *Let G be a simple graph on $n \geq 3$ vertices with $n-1$ edges. Let L be its Laplacian matrix and let $\beta = 2\left(1 - \cos\left(\frac{\pi}{n}\right)\right)$ and $\alpha = \frac{\beta}{n}$. Then, G is a tree if and only if $L + \alpha\mathbf{J}_n - \beta\mathbf{I}_n \succeq \mathbf{0}$.*

Since linear matrix inequalities are tractable constraints in an optimization model, Proposition 2 can be used to derive an MISDP formulation for the QMSTP. To do so, let us first define the set of adjacency matrices of induced subgraphs of G with n vertices and $n-1$ edges:

$$\mathcal{F} := \left\{X \in \{0,1\}^{n\times n} \cap S^n : \langle X, \mathbf{J}_n\rangle = 2(n-1),\ x_{ij} = 0 \text{ if } \{i,j\} \notin E\right\}. \quad (4)$$

The set of all adjacency matrices of spanning trees induced by G is contained in $\mathcal{F}$, which we denote by

$$\mathcal{T}_M = \mathcal{F} \cap \{X \in S^n : \mathrm{Diag}(X\mathbf{1}) - X + \alpha\mathbf{J} - \beta\mathbf{I} \succeq \mathbf{0}\}, \quad (5)$$

where $\alpha = \frac{\beta}{n}$ and $\beta = 2\left(1 - \cos\left(\frac{\pi}{n}\right)\right)$. Indeed, there is a bijection between the entries in $\mathcal{T}$, see (1), and $\mathcal{T}_M$ given by $\mathcal{B} : \mathcal{T}_M \to \mathcal{T}$, where $\mathcal{B}(X)$ maps X to a column vector containing the entries of X corresponding to E with respect to the fixed ordering of the edge set. We let $\mathcal{B}^* : \mathcal{T} \to \mathcal{T}_M$ denote its adjoint operator, which maps a column vector x to an adjacency matrix containing the elements of x on the positions of E and zeros elsewhere.

Now, for a given $X \in \mathcal{T}_M$, we define the lifted matrix $Y = \mathcal{B}(X)\mathcal{B}(X)^\top$ such that $Y_{ef} = 1$ if and only if $X_{ij} = 1$ and $X_{k\ell} = 1$, where $e = \{i,j\}$ and $f = \{k,\ell\}$. To handle the rank-one constraint $Y = \mathcal{B}(X)\mathcal{B}(X)^\top$, we exploit the following result on binary PSD matrices.

Theorem 1 (De Meijer and Sotirov (2024b)). *Let $Z = \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \succeq \mathbf{0}$ with $\mathrm{diag}(X) = x$. Then, $\mathrm{rank}(Z) = 1$ if and only if $X \in \{0,1\}^{n\times n}$.*

Theorem 1 implies that we can enforce $Y = \mathcal{B}(X)\mathcal{B}(X)^\top$ by requiring $\begin{pmatrix} Y & \mathcal{B}(X) \\ \mathcal{B}(X)^\top & 1 \end{pmatrix} \succeq \mathbf{0}$, $\mathrm{diag}(Y) = \mathcal{B}(X)$ and $Y \in \{0,1\}^m$. Combining Proposition 1 with Theorem 1, we formulate the QMSTP as the following mixed-integer semidefinite program:

$$\min \langle Q, Y\rangle \quad (6a)$$

$$\text{s.t. } \mathrm{diag}(Y) = \mathcal{B}(X) \quad (6b)$$

$$\begin{pmatrix} Y & \mathcal{B}(X) \\ \mathcal{B}(X)^\top & 1 \end{pmatrix} \succeq \mathbf{0} \quad (6c)$$

$$\mathrm{Diag}(X\mathbf{1}) - X + \alpha\mathbf{J}_n - \beta\mathbf{I}_n \succeq \mathbf{0} \quad (6d)$$

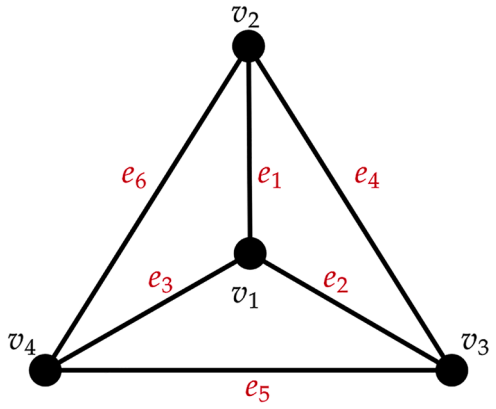

Fig. 1. $G = K_4$ with given edge and vertex labeling.

$$Y \in S^m,\ X \in \mathcal{F}, \quad (6e)$$

where $\mathcal{F}$ is given in (4). In (6), we do not impose integrality on the off-diagonal elements of Y as those follow from the integrality of $\mathcal{B}(X)$, see (De Meijer and Sotirov, 2024b).

The MISDP formulation (6) provides an alternative to the mixed-integer linear formulation (3). The natural question arises which of these formulations is stronger, i.e., what is the relationship between the continuous SDP (or actually DNN) relaxation of (6) and the continuous LP relaxation of (3). We now show that none of these relaxations dominates the other. To do so, we consider the natural mappings between solutions of (3) and (6) given by $(x, Y) \mapsto (\mathcal{B}^*(x), Y)$ and $(X, Y) \mapsto (\mathcal{B}(X), Y)$. We show that for each relaxation there exists a feasible solution whose image under this natural mapping is not feasible for the other problem.

We start by showing that the continuous relaxation of (3) does not dominate the continuous relaxation of (6). Let $G = K_4$ with the edge labeling as presented in Fig. 1 and take

$$x = \begin{pmatrix} \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{pmatrix}^\top,$$

$$Y = \begin{pmatrix} \frac{1}{3}\mathbf{I}_3 + \frac{1}{3}\mathbf{J}_3 & \frac{2}{3}\mathbf{I}_3 \\ \frac{2}{3}\mathbf{I}_3 & \frac{1}{3}\mathbf{I}_3 \end{pmatrix}.$$

It is not difficult to verify that the tuple (x, Y) is feasible for the continuous relaxation of (3). Indeed, we have $\mathrm{diag}(Y) = x$, $Y\mathbf{1}_6 = (n-1)x$, $\mathbf{0} \leq Y \leq \mathbf{J}_6$, $Y \in S^6$ and $x \in \mathcal{T}$, where the last inclusion follows from the fact that x satisfies all cut-set constraints (2). Now, let us define the matrix Z as $Z = \begin{pmatrix} Y & x \\ x^\top & 1 \end{pmatrix}$. It turns out that Z has eigenvalue $-\frac{1}{3}$, thus Z is not positive semidefinite. We conclude that $(\mathcal{B}^*(x), Y)$ is not feasible for (6).

Next, we show that the continuous relaxation of (6) does not dominate the continuous relaxation of (3). Consider again $G = K_4$ with the labeling as depicted in Fig. 1. Now, take

$$X = \begin{pmatrix} 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & 0 & \frac{3}{4} & \frac{3}{4} \\ \frac{1}{4} & \frac{3}{4} & 0 & \frac{3}{4} \\ \frac{1}{4} & \frac{3}{4} & \frac{3}{4} & 0 \end{pmatrix} \quad \text{and} \quad Y = \begin{pmatrix} \frac{1}{16}\mathbf{J}_3 + \frac{3}{16}\mathbf{I}_3 & \frac{3}{16}\mathbf{J}_3 \\ \frac{3}{16}\mathbf{J}_3 & \frac{9}{16}\mathbf{J}_3 + \frac{3}{16}\mathbf{I}_3 \end{pmatrix},$$

for which it holds that $\mathcal{B}(X) = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{3}{4} & \frac{3}{4} & \frac{3}{4} \end{pmatrix}^\top$. One can verify that $\mathrm{diag}(Y) = \mathcal{B}(X)$, $\begin{pmatrix} Y & \mathcal{B}(X) \\ \mathcal{B}(X)^\top & 1 \end{pmatrix} \succeq \mathbf{0}$, $\mathrm{Diag}(X\mathbf{1}_4) - X + \alpha\mathbf{J}_4 - \beta\mathbf{I}_4 \succeq \mathbf{0}$, $Y \in S^6$ and $X \in \mathcal{F}$. Hence, (X, Y) is feasible for (6). Now, let us define $x = \mathcal{B}(X)$. Then, for $S = \{v_1\}$ we have $\sum_{e\in\delta(\{v_1\})} x_e = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} < 1$, so the cut-set constraint for S is not satisfied. We conclude that $x \notin \mathcal{T}$, implying that $(\mathcal{B}(X), Y)$ is not feasible for (3).

We summarize these results in the following corollary.

Corollary 1. *The continuous relaxation of the cut-set QMSTP formulation (3) and the continuous relaxation of (6) do not dominate one another.*

3.2. Valid inequalities

In this section, we derive Chvátal-Gomory cuts from the MISDP formulation of the QMSTP introduced in the previous section. Interestingly, both the cut-set constraints (2) and some of the first-level RLT constraints turn out to be CG cuts resulting from (6).

The Chvátal-Gomory (CG) cuts form a well-known family of cutting planes for integer programs, initially designed for integer linear programs by Chvátal (1973) and Gomory (1958). Similar cuts for integer conic problems and integer semidefinite programs have been investigated by Çezik and Iyengar (2005) and De Meijer and Sotirov (2024a), respectively.

In the sequel, we derive the cut-set constraints (2) as CG cuts. Let $S \subsetneq V$, $S \neq \emptyset$ and X be feasible for (6). Then, for the PSD matrix $\mathbb{1}_S \mathbb{1}_S^\top$ we have that

$$\langle \mathbb{1}_S \mathbb{1}_S^\top, \operatorname{Diag}(X\mathbf{1}) - X + \alpha \mathbf{J} - \beta \mathbf{I} \rangle \geq 0, \tag{7}$$

is a valid inequality for (6). Here, we exploit the fact that the trace inner product of two positive semidefinite matrices is non-negative. After rewriting (7) and exploiting $\langle \mathbb{1}_S \mathbb{1}_S^\top, \operatorname{Diag}(X\mathbf{1}) \rangle = \langle \mathbb{1}_S \mathbf{1}^\top, X \rangle$, we have $\langle \mathbb{1}_S \mathbb{1}_S^\top - \mathbb{1}_S \mathbf{1}^\top, X \rangle \leq \langle \mathbb{1}_S \mathbb{1}_S^\top, \alpha \mathbf{J} - \beta \mathbf{I} \rangle$. Since the left-hand side of this inequality is integer, we may round down the right-hand side to the nearest integer value, which results in the CG cut $\langle \mathbb{1}_S \mathbb{1}_S^\top - \mathbb{1}_S \mathbf{1}^\top, X \rangle \leq \lfloor \langle \mathbb{1}_S \mathbb{1}_S^\top, \alpha \mathbf{J} - \beta \mathbf{I} \rangle \rfloor$. After rewriting the left-hand side, this inequality becomes

$$-\sum_{i \in S} \sum_{j \notin S} x_{ij} \leq \lfloor |S|(|S|\alpha - \beta) \rfloor. \tag{8}$$

For $\alpha = \frac{\beta}{n}$ and $\beta = 2\left(1 - \cos\left(\frac{\pi}{n}\right)\right)$, see Proposition 2, we have $\lfloor |S|(|S|\alpha - \beta) \rfloor = -1$, and thus the above CG cut implies the cut-set constraint $\sum_{e \in \delta(S)} x_e \geq 1$, see also (2). Let us summarize the previous discussion.

Proposition 3. *Let $S \subsetneq V$, $S \neq \emptyset$. Then, the cut-set constraint (2) is a Chvátal-Gomory cut with respect to the MISDP (6).*

Subsequently, we derive other valid inequalities by exploiting the constraint (6c), that may be equivalently reformulated as $Y - \mathcal{B}(X)\mathcal{B}(X)^\top \succeq \mathbf{0}$. Let X, Y be feasible for (6), $i \in V$, and $\mathbb{1}_{\delta(i)}$ be the indicator vector of $\delta(i)$. For $f \in E$, we define the following positive semidefinite matrix $P_f := \mathbf{u}_k \mathbb{1}_{\delta(i)}^\top + \mathbb{1}_{\delta(i)} \mathbf{u}_k^\top + \mathbf{I}_m + (n-1)\mathbf{u}_k \mathbf{u}_k^\top$, where the index k corresponds to the ordering number of the edge f, i.e., $\mathcal{B}(X)_k = x_f$. Since $P_f \succeq \mathbf{0}$, it follows that $\langle Y - \mathcal{B}(X)\mathcal{B}(X)^\top, P_f \rangle \geq 0$. By rewriting the left-hand side, we have

$$\langle Y - \mathcal{B}(X)\mathcal{B}(X)^\top, P_f \rangle = 2 \sum_{e \in \delta(i)} y_{fe} - 2 x_f \mathcal{B}(X) \mathbb{1}_{\delta(i)}^\top \geq 0,$$

from where it follows $\sum_{e \in \delta(i)} y_{fe} \geq x_f \mathcal{B}(X) \mathbb{1}_{\delta(i)}^\top \geq x_f$, since $\mathcal{B}(X) \mathbb{1}_{\delta(i)}^\top \geq 1$ due to the fact that the underlying graph is connected. Observe that $\mathcal{B}(X) \mathbb{1}_{\delta(i)}^\top \geq 1$ is a cut-set constraint, which by Proposition 3 is a CG cut. Therefore, the latter inequality implicitly contains a rounding step. Thus, we have obtained the constraints

$$\sum_{e \in \delta(i)} y_{fe} \geq x_f \qquad \forall f \in E, \forall i \in V. \tag{9}$$

Interestingly, these constraints also follow from the reformulation-linearization technique (Sherali & Adams, 1998) applied to the cut-set constraints (2) with $|S| = 1$. The reformulation-linearization technique (RLT) is a systematic method to derive stronger relaxations for nonconvex programming problems by pairwise multiplication of constraints and/or variables, followed by a linearization step (Sherali & Adams, 1998). Indeed, after multiplying both sides of (2) by x_f and replacing $x_f x_e$ by y_{fe}, one obtains the constraints (9). Although sometimes written in the form of subtour elimination constraints, the inequalities (9) have been considered for the QMSTP in the works of Öncan and Punnen (2010), Rostami and Malucelli (2015), Pereira et al. (2015b), Guimarães et al. (2020), and Sotirov and Verchére (2024). We refer later to the constraints (9) as *RLT-type constraints*.

Proposition 4. *Let $i \in V$ and $f \in E$. Then, the constraint (9) is a Chvátal-Gomory cut with respect to the MISDP (6).*

In the next section, we show how the inequalities (9) are exploited to strengthen our DNN relaxation for the QMSTP, see (11).

4. DNN Relaxation

Here, we derive two doubly nonnegative relaxations for the QMSTP and show that these relaxations are not strictly feasible. We apply facial reduction in order to derive facially reduced equivalent formulations. We emphasize that the relaxations introduced here are not the strongest QMSTP relaxations in the literature. In fact, the SDP relaxation of Guimarães et al. (2020) is theoretically stronger. However, in anticipation of the algorithm for solving our relaxations, see Section 5, we aim at relaxations that are both strong and tractable to solve in order to balance the quality of the bound and the efficiency of computing it.

Starting from the formulation (6), we introduce a vector y that is equal to the diagonal of the matrix variable Y. We drop the linear matrix inequality (6d), and relax the constraint $X \in \mathcal{F}$, see (6e), to $\mathbf{1}^\top y = n - 1$. Furthermore, we add the constraint $Y\mathbf{1} = (n-1)y$ that can be derived from (6). Additionally, we impose nonnegativity constraints on the matrix variable, and obtain the following DNN relaxation:

$$\min \langle Q, Y \rangle \tag{10a}$$

$$\text{s.t. } \operatorname{diag}(Y) = y \tag{10b}$$

$$Y\mathbf{1} = (n-1)y, \ \mathbf{1}^\top y = n - 1 \tag{10c}$$

$$Y \geq \mathbf{0}, \begin{pmatrix} Y & y \\ y^\top & 1 \end{pmatrix} \succeq \mathbf{0}, \tag{10d}$$

where $Y \in \mathcal{S}^m$. The above relaxation does not include the connectivity constraint (6d), because that constraint has only a small impact on the bound and it makes the relaxation more difficult to solve. In order to include a type of connectivity constraints in (10), we consider valid inequalities from Section 3.2. Preliminary computational results show that by adding the cut-set constraints (2), see also Proposition 3, the resulting bound only marginally improves on the DNN bound (10). The RLT-type cuts (9), however, turn out to have a more positive impact on the bound value. We therefore present the following strengthening of the relaxation (10):

$$\begin{aligned} &\min \langle Q, Y \rangle \\ &\text{s.t. (10b)–(10d)} \\ &\sum_{e \in \delta(i)} y_{fe} \geq y_f \qquad \forall f \in E, \forall i \in V. \end{aligned} \tag{11}$$

The RLT-type cuts (9) that are added in (11) belong to a class of exponentially many RLT-type of subtour elimination constraints that may be used for solving the QMSTP, see e.g., Guimarães et al. (2020), Pereira et al. (2015b), Rostami and Malucelli (2015), Sotirov and Verchére (2024). Several relaxations including the RLT-type of subtour elimination constraints have been proposed in the literature. In particular, Pereira et al. (2015b) propose an incomplete first-level RLT relaxation for the QMSTP that contains a large number of RLT-type of subtour elimination constraints. We emphasize that, although written differently, the cuts (9) are all included in the relaxation proposed by Pereira et al. (2015b). Guimarães et al. (2020) propose an SDP relaxation for the QMSTP that consists of the incomplete first-level RLT relaxation of Pereira et al. (2015b) accompanied by an additional PSD constraint as in (10d). Hence, both the relaxations (10) and (11) are theoretically dominated by the SDP relaxation of Guimarães et al. (2020). Sotirov and Verchére (2024) compute bounds for the complete first-level RTL relaxation for the QMSTP and show that those bounds dominate the bounds from

Pereira et al. (2015b). Finally, Rostami and Malucelli (2015) derive an incomplete second-level RLT relaxation, that dominates previously mentioned first-level RLT relaxations, but not the SDP relaxations.

From the previous discussion it becomes clear that (10) and (11) would benefit from adding the remaining (exponentially many) RLT-type subtour elimination constraints, which would result in the stronger relaxation proposed by Guimarães et al. (2020). We decide not to do so, due to the type of algorithm we will use to solve our relaxations in Section 5. The algorithm of Guimarães et al. (2020) is based on a Lagrangian relaxation scheme where the PSD constraint is dualized, after which the subproblems are solved as minimum spanning tree problems. Thus, the exponentially many subtour elimination constraints are implicitly dealt with by a combinatorial algorithm, e.g., the classical MST algorithm of Prim (1957). In contrast, we solve our relaxations with the Peaceman-Rachford splitting method (Peaceman & Rachford, 1955) originating from convex optimization. This algorithm is suitable for efficiently solving large-scale SDPs. As a drawback, however, the addition of many polyhedral cutting planes complicates the approach, especially when these cuts have overlapping support, see Section 5.2 for details. For that reason, we restrict ourselves to the constraints (9), resulting in an efficient algorithm that computes bounds that improve on the basic SDP relaxation (10). Although the resulting bound is theoretically weaker than the bound from Guimarães et al. (2020), it is interesting to note that this is not what we observe in practice. Since our approach is able to solve the relaxations up to high precision, the resulting bounds are in practice stronger than the ones reported by Guimarães et al. (2020). We refer to Section 6 for details.

In the remaining part of this section, we consider the strict feasibility of the DNN relaxations. Let $\widetilde{Y} = \begin{pmatrix} Y & y \\ y^\top & 1 \end{pmatrix}$ and $\widetilde{Q} = \begin{pmatrix} Q & \mathbf{0}_m \\ \mathbf{0}_m^\top & 0 \end{pmatrix}$. A DNN relaxation is called strictly or Slater feasible if there exists a feasible matrix $\widetilde{Y} \succ \mathbf{0}$ that satisfies all linear inequalities with strict inequality. Such a matrix is called a Slater feasible point. It is not difficult to verify that

$$T = \begin{pmatrix} \mathbf{1}_m \\ -(n-1) \end{pmatrix} \tag{12}$$

is an eigenvector corresponding to the zero eigenvalue of any matrix $\widetilde{Y}$ feasible for (10). Since there is no feasible matrix $\widetilde{Y}$ which is positive definite, the DNN relaxation (10) has no Slater feasible point.

As the intersection of the feasible set of (10) and the interior of $\mathcal{S}_+^{m+1}$ turns out to be empty, the feasible set of (10) is entirely contained in one of the faces of $\mathcal{S}_+^{m+1}$. The facial structure of the PSD cone can be exploited to project the feasible set of (10) onto the minimal face of $\mathcal{S}_+^{m+1}$ that contains it (Drusvyatskiy & Wolkowicz, 2017). This facial reduction technique can be used to obtain equivalent DNN relaxations that are strictly feasible, see (Hu et al., 2023).

To provide a facially reduced DNN relaxation of (10), let $W \in \mathbb{R}^{(m+1)\times m}$ be a matrix whose columns form a basis for $\mathcal{W} = \text{null}(T^\top)$, see (12). As we will show in Theorem 3 later on in this section, the relaxation (10) may be equivalently written as the following facially reduced relaxation:

$$\begin{aligned} \min\ & \langle \widetilde{Q}, WRW^\top \rangle \\ \text{s.t. } & \text{diag}(WRW^\top) = (WRW^\top)\mathbf{u}_{m+1} \\ & (WRW^\top)_{m+1,m+1} = 1 \\ & WRW^\top \geq \mathbf{0}, \quad R \succeq \mathbf{0}. \end{aligned} \tag{13}$$

We obtained this relaxation from (10) by replacing $\widetilde{Y}$ with $WRW^\top$ and removing redundant constraints. Note that the feasible set of (10) is contained in $W\mathcal{S}_+^m W^\top$, which is a face of $\mathcal{S}_+^{m+1}$. To show that (13) has an interior point, we use Theorem 3.15 from (Hu et al., 2023). That theorem additionally takes into account a zero pattern in the feasible matrix, which is not present in our problem.

Theorem 2 (Theorem 3.15 in Hu et al. (2023)). *Let* $\mathcal{Q} = \left\{ y \in \mathbb{R}^m : \mathcal{A}\left(\begin{pmatrix} yy^\top & y \\ y^\top & 1 \end{pmatrix} \right) = \mathbf{0},\ y \geq \mathbf{0} \right\}$, *where* $\mathcal{A}$ *is a linear transformation, be the feasible set of a quadratically constrained program. Suppose* $\text{aff}(\text{conv}(\mathcal{Q})) = \mathcal{L}$ *with* $\dim(\mathcal{L}) = p$. *Then, there exist a matrix* C *with full row rank and* d *such that* $\mathcal{L} = \{y \in \mathbb{R}^m : Cy = d\}$. *Let* $M = (C \ \ -d)$ *and* W *be a matrix such that its columns form a basis of* $\text{null}(M)$. *Let* $\mathcal{J} = \{(i,j) : y_i y_j = 0\ \forall y \in \mathcal{Q}\}$ *and* $\mathcal{J}^c$ *be its complement. Then, there exists a Slater point* $\hat{R}$ *for the facially reduced, DNN feasible set:*

$$\hat{\mathcal{Q}}_R = \Big\{ R \in \mathcal{S}^{p+1} : R \succeq \mathbf{0},\ \left(WRW^\top\right)_{\mathcal{J}} = 0,\ \left(WRW^\top\right)_{\mathcal{J}^c} \geq \mathbf{0},\ \mathcal{A}(WRW^\top) = \mathbf{0} \Big\}.$$

We are now ready to state the following result on our facially reduced problem.

Theorem 3. *For* $n \geq 3$, *the DNN relaxation* (13) *is a strictly feasible equivalent reformulation of* (10).

Proof. Let

$$\begin{aligned} \mathcal{Q}_n(m) &= \{y \in \{0,1\}^m : \mathbf{1}_m^\top y = n-1\} \\ &= \{y \in \mathbb{R}^m : \mathbf{1}_m^\top y = n-1,\ y_i y_i = y_i\ \forall i \in [m]\} \\ &= \left\{ y \in \mathbb{R}^m : \mathcal{A}\left(\begin{pmatrix} yy^\top & y \\ y^\top & 1 \end{pmatrix} \right) = \mathbf{0},\ y \geq \mathbf{0} \right\}, \end{aligned}$$

where $\mathcal{A}(X) = \left(\mathcal{A}_1(X), \cdots, \mathcal{A}_{2m+1}(X)\right)^\top$ with

$$\mathcal{A}_i(X) = \left\langle \begin{pmatrix} \mathbf{u}_i\mathbf{u}_i^\top & -\frac{1}{2}\mathbf{u}_i \\ -\frac{1}{2}\mathbf{u}_i^\top & 0 \end{pmatrix}, X \right\rangle \text{for all } i \in [m], \text{ and}$$

$$\mathcal{A}_{m+i}(X) = \left\langle \frac{1}{2}\left(\mathbf{u}_i \begin{pmatrix} \mathbf{1}_m^\top & -(n-1) \end{pmatrix} + \begin{pmatrix} \mathbf{1}_m \\ -(n-1) \end{pmatrix} u_i^\top \right), X \right\rangle \quad \text{for all } i \in [m+1].$$

Note that in the definition of $\mathcal{Q}_n(m)$, the equality $\mathcal{A}_i(X) = 0$ models the constraint $y_i^2 = y_i$ for all $i \in [m]$. The constraint $\mathcal{A}_{2m+1}(X) = 0$ models the constraint $\mathbf{1}_m^\top y = n-1$. For the indices $i \in [m]$, the constraint $\mathcal{A}_{m+i}(X) = 0$ models the redundant constraint $y_i(\mathbf{1}_m^\top y) = (n-1)y_i$ for all $1 \leq i \leq m$.

The convex hull equals $\text{conv}(\mathcal{Q}_n(m)) = \{y \in [0,1]^m : \mathbf{1}_m^\top y = n-1\}$. For each index $i \in [m]$ there exist vectors $y^1, y^2 \in \mathcal{Q}_n(m)$ such that $y_i^1 > 0$ and $y_i^2 < 1$, hence, we get that the affine hull is $\text{aff}(\text{conv}(\mathcal{Q}_n(m))) = \{y \in \mathbb{R}^m : \mathbf{1}_m^\top y = n-1\}$, and has dimension $m - \text{rank}(\mathbf{1}_m^\top) = m-1$. Hence, $M = T^\top$ where M is from Theorem 2 and T given in (12). Let $W \in \mathbb{R}^{(m+1)\times m}$ be a matrix whose columns form a basis of the nullspace of M. Then, a face of $\mathcal{S}_+^{m+1}$ containing the feasible set of (10) is of the form $W\mathcal{S}_+^m W^\top$. Therefore, one can replace $\widetilde{Y}$ with $WRW^\top$ in (10).

Moreover, it holds that for each pair of indices $(i,j) \in [m] \times [m]$, there exists a vector $y \in \mathcal{Q}_n(m)$ such that $y_i = y_j = 1$, and hence the index set $\mathcal{J} = \{(i,j) : y_i y_j = 0\ \forall y \in \mathcal{Q}\}$ is empty. Thus, by Theorem 2, there exists a Slater feasible point for the facially reduced DNN relaxation (13). □

On top of imposing strict feasibility, facial reduction reduces both the number of variables and constraints. Therefore, the relaxation (13) is preferred over (10). In a similar fashion, relaxation (11) can be rewritten by replacing $\widetilde{Y}$ in (11) by $WRW^\top$.

5. Peaceman-Rachford splitting method for the QMSTP

Interior point solvers have difficulties computing our DNN relaxations for medium-sized problems in a reasonable time due to the large number of (inequality) constraints. Therefore, we use the Peaceman-Rachford splitting method (PRSM) for computing the bounds. The PRSM was first proposed in (Lions & Mercier, 1979; Peaceman & Rachford, 1955) and is a symmetric variant of the alternating direction method of multipliers (ADMM). For more details and convergence results, we refer to (He et al., 2016).

5.1. PRSM For solving the DNN relaxation

In this section, we outline the main steps of the Peaceman-Rachford splitting method for solving the DNN relaxation for the QMSTP (13).

Recall that the matrix W should be such that its columns provide a basis for $\mathcal{W} = \text{null}(T^\top)$. For reasons explained later, we additionally require the columns of W to be orthonormal. Therefore, we take W as the matrix obtained from applying a QR decomposition to $\begin{pmatrix}(n-1)\mathbf{I}_m & \mathbf{1}_m\end{pmatrix}^\top$.

Now, we define the following sets

$$\mathcal{R} := \{R \in S^m : R \succeq \mathbf{0},\ \text{tr}(R) = n\}, \tag{14}$$

$$\mathcal{Y} := \left\{\widetilde{Y} \in S^{m+1} : \widetilde{Y} = \begin{pmatrix} Y & y \\ y^\top & 1\end{pmatrix},\ \text{diag}(Y) = y,\ \mathbf{0} \le \widetilde{Y} \le \mathbf{J},\ \text{tr}(\widetilde{Y}) = n\right\}, \tag{15}$$

and rewrite (13) as

$$\min\left\{\left\langle \widetilde{Q}, \widetilde{Y}\right\rangle : \widetilde{Y} = WRW^\top,\ R \in \mathcal{R},\ \widetilde{Y} \in \mathcal{Y}\right\}. \tag{16}$$

Note that we added redundant constraints to $\mathcal{Y}$ and $\mathcal{R}$, where the constraint $\text{tr}(R) = n$ holds, since the columns in W are orthonormalized. Those redundant constraints help for the efficiency of the algorithm, see e.g., (Li et al., 2021; De Meijer et al., 2023; Oliveira et al., 2018).

The class of alternating direction augmented Lagrangian methods, to which the PRSM belongs, aims to minimize the augmented Lagrangian function, see e.g., (Boyd et al., 2011; Wen et al., 2010). For a fixed penalty parameter $\tau > 0$, the augmented Lagrangian function of (16) w.r.t. the constraint $\widetilde{Y} = WRW^\top$ is

$$\mathcal{L}_\tau(R, \widetilde{Y}, S) = \left\langle \widetilde{Q}, \widetilde{Y}\right\rangle + \left\langle S, \widetilde{Y} - WRW^\top\right\rangle + \frac{\tau}{2}\left\|\widetilde{Y} - WRW^\top\right\|_F^2.$$

The basic idea of the PRSM is to iteratively alternate between optimizing $\mathcal{L}_\tau$ with respect to R and $\widetilde{Y}$ and updating the dual variable S. The $(k+1)$-th iteration of the PRSM to minimize the augmented Lagrangian function is

$$\begin{aligned}
R^{k+1} &= \arg\min_{R\in\mathcal{R}} \mathcal{L}_\tau(R, \widetilde{Y}^k, S^k)\\
S^{\frac{k+1}{2}} &= S^k + \gamma_1\tau(\widetilde{Y}^k - WR^{k+1}W^\top)\\
\widetilde{Y}^{k+1} &= \arg\min_{\widetilde{Y}\in\mathcal{Y}} \mathcal{L}_\tau(R^{k+1}, \widetilde{Y}, S^{\frac{k+1}{2}})\\
S^{k+1} &= S^{\frac{k+1}{2}} + \gamma_2\tau(\widetilde{Y}^{k+1} - WR^{k+1}W^\top),
\end{aligned}$$

with step lengths $\gamma_1 \in (-1, 1)$ and $\gamma_2 \in \left(0, \frac{1+\sqrt{5}}{2}\right)$ satisfying $\gamma_1 + \gamma_2 > 0$ and $|\gamma_1| < 1 + \gamma_2 - \gamma_2^2$, see (He et al., 2016). The optimization problems occurring in this PRSM scheme can be simplified to projection problems. Namely, optimizing the augmented Lagrangian over $\mathcal{R}$ can be simplified to

$$\begin{aligned}
R^{k+1} &= \arg\min_{R\in\mathcal{R}} \langle S^k, -WRW^\top\rangle + \frac{\tau}{2}\left\|\widetilde{Y}^k - WRW^\top\right\|_F\\
&= \mathcal{P}_{\mathcal{R}}\left(W^\top\left(\widetilde{Y}^k + \frac{1}{\tau}S^k\right)W\right),
\end{aligned}$$

where we exploited the fact that the columns of W are orthonormal. The projection $\mathcal{P}_{\mathcal{R}}(M)$ of a matrix $M \in S^m$ onto the set $\mathcal{R}$ can be computed by projecting the eigenvalues of M in the spectral decomposition onto the n-simplex Δ_n, see e.g., (Li et al., 2021). In more detail, let $M = U\text{Diag}(\lambda)U^\top$ be the eigenvalue decomposition of M with λ denoting the vector of eigenvalues of M, then $\mathcal{P}_{\mathcal{R}}(M) = U\text{Diag}(\mathcal{P}_{\Delta_n}(\lambda))U^\top$. The projection onto the simplex can be performed efficiently. We refer to (Condat, 2016) for an overview of algorithms for projecting onto the simplex and their complexities.

Similarly, the optimization problem over the polyhedral set $\mathcal{Y}$ can be reformulated as

$$\begin{aligned}
\widetilde{Y}^{k+1} &= \arg\min_{\widetilde{Y}\in\mathcal{Y}} \left\langle \widetilde{Q}, \widetilde{Y}\right\rangle + \left\langle S^{\frac{k+1}{2}}, \widetilde{Y}\right\rangle + \frac{\tau}{2}\left\|\widetilde{Y} - WR^{k+1}W^\top\right\|_F\\
&= \mathcal{P}_{\mathcal{Y}}\left(WR^{k+1}W^\top - \frac{1}{\tau}\left(\widetilde{Q} + S^{\frac{k+1}{2}}\right)\right).
\end{aligned}$$

The projection onto $\mathcal{Y}$ can then be done in the following way

$$\mathcal{P}_{\mathcal{Y}}\left(\begin{pmatrix} Z & z\\ z^\top & \omega\end{pmatrix}\right) = \mathcal{P}_{[0,1]}\left(\begin{pmatrix} Z - \text{Diag}(\text{diag}(Z)) + v & v\\ v^\top & 1\end{pmatrix}\right),$$

where $v = \mathcal{P}_{\bar{\Delta}(n-1)}\left(\frac{1}{3}\text{diag}(Z) + \frac{2}{3}z\right)$ and $\mathcal{P}_{[0,1]}$ denotes the elementwise projection onto the interval $[0, 1]$.

5.2. PRSM For solving the strengthened DNN relaxation

In this subsection, we modify the previously described PRSM algorithm that solves the relaxation (13), so that it can handle additional RLT-type constraints.

Let us extend the set $\mathcal{Y}$, see (15), by adding the RLT-type constraints, yielding

$$\mathcal{Y}_{RLT} = \left\{\widetilde{Y} \in S^{m+1} : \widetilde{Y} = \begin{pmatrix} Y & y\\ y^\top & 1\end{pmatrix},\ \text{diag}(Y) = y,\ \text{tr}(\widetilde{Y}) = n,\ \mathbf{0} \le \widetilde{Y} \le \mathbf{J},\ \sum_{e\in\delta(i)} y_{fe} \ge y_f \quad \forall f \in E, \forall i \in V\right\}.$$

Thus, the strengthened DNN relaxation (16) is as follows

$$\min\left\{\left\langle \widetilde{Q}, \widetilde{Y}\right\rangle : \widetilde{Y} = WRW^\top,\ R \in \mathcal{R},\ \widetilde{Y} \in \mathcal{Y}_{RLT}\right\}. \tag{17}$$

The RLT-type constraints make the projection onto $\mathcal{Y}_{RLT}$ significantly harder. To the best of our knowledge, there is no closed-form expression for the projection onto $\mathcal{Y}_{RLT}$. However, one may write $\mathcal{Y}_{RLT}$ as an intersection of sets that are easier to project on and then use an algorithm to project onto the intersection of convex sets. The cyclic Dykstra's projection algorithm (Boyle & Dykstra, 1986) is a suitable algorithm. An overview and analysis of algorithms to project onto the intersection of convex sets can be found in (Bauschke & Koch, 2015).

To apply Dykstra's cyclic projection algorithm, let $\mathcal{K}$ denote a coloring of the graph G, i.e., $\mathcal{K} = \{K_1, \dots, K_N\}$ is a partitioning of V into independent sets of G. We then define the polyhedral sets $\mathcal{Y}^k$ as

$$\mathcal{Y}^k := \left\{\widetilde{Y} \in \mathbb{R}^{(m+1)\times(m+1)} : \widetilde{Y} = \begin{pmatrix} Y & y\\ y^\top & 1\end{pmatrix},\ \text{diag}(Y) = y,\ \sum_{e\in\delta(i)} y_{fe} \ge y_f \quad \forall f \in E, \forall i \in K_k\right\},$$

for $k = 1, \dots, N$. With this we can now rewrite $\mathcal{Y}_{RLT}$ as $\mathcal{Y}_{RLT} = \mathcal{Y} \cap \left(\bigcap_{k=1}^N \mathcal{Y}^k\right)$.

The projection onto the sets $\mathcal{Y}^k$ can be performed independently over each row $f \in E$ of Y and the corresponding entries of y_f in $\widetilde{Y}$. This allows us to restrict ourselves to projections onto the following type of sets

$$T_f^k := \left\{z \in \mathbb{R}^{m+2} : z_f = z_{m+1} = z_{m+2},\ \sum_{e\in\delta(i)} z_e \ge z_f \quad \forall i \in K_k\right\}, \tag{18}$$

where the first $m+1$ entries correspond to the f-th row of $\widetilde{Y}$ and the last entry z_{m+2} corresponds to $\widetilde{Y}_{m+1,f}$. The projection onto T_f^k can then be computed as presented in the following proposition.

Proposition 5. *Let $a \in \mathbb{R}^{m+2}$, $f \in E$ and let $\mathcal{K} = \{K_1, \dots, K_N\}$ denote a coloring of G. For each $i \in K_k$, we define $g_i := \frac{a_f + a_{m+1} + a_{m+2}}{3} - \sum_{e\in\delta(i)} a_e$ and sort these values in non-increasing order, i.e., $g_{\sigma(1)} \ge g_{\sigma(2)} \ge \cdots \ge g_{\sigma(n_k)}$, where $n_k = |K_k|$ and $\sigma : [n_k] \to K_k$ is an appropriate sorting permutation. For each $p \in [n_k]$, let*

$$\omega(p) := \frac{\sum_{j=1}^p \frac{g_{\sigma(j)}}{d(\sigma(j))}}{3 + \sum_{j=1}^p \frac{1}{d(\sigma(j))}},$$

where $d(\sigma(j))$ denotes the degree of vertex $\sigma(j)$ in G. If $g_i \le 0$ for all $i \in K_k$, then $\mathcal{P}_{T_f^k}(a) = z$, where $z_e = a_e$ for all $e \in E \setminus \{f\}$ and $z_f = z_{m+1} = z_{m+2} = \frac{a_f + a_{m+1} + a_{m+2}}{3}$. Otherwise, let p^* denote the largest index p for which $g_{\sigma(p)} > \omega(p)$. Then, $\mathcal{P}_{T_f^k}(a) = z$, where

$$z_e = \begin{cases} \frac{a_f + a_{m+1} + a_{m+2}}{3} - \omega(p^*) & \text{if } e \in \{f, m+1, m+2\},\\ a_e + \frac{1}{d(i)}(g_i - \omega(p^*)) & \text{if } e \in \delta(i) \setminus \{f\} \text{ for } i \in K_k \setminus V(f) \text{ with } \sigma(i) \le p^*,\\ a_e - \frac{1}{d(i)-1}\sum_{e\in\delta(i)\setminus\{f\}} a_e & \text{if } e \in \delta(i) \setminus \{f\} \text{ for } i \in K_k \cap V(f) \text{ with } \sum_{e\in\delta(i)\setminus\{f\}} a_e < 0,\\ a_e & \text{otherwise.}\end{cases}$$

Proof. First, observe that if $g_{\sigma(1)} \leq 0$, then $g_i \leq 0$ for all $i \in K_k$. Consequently, the projection of a onto T_f^k is given by z, where z is such that $z_e = a_e$ for all $e \in E \setminus \{f\}$ and $z_f = z_{m+1} = z_{m+2} = \frac{a_f + a_{m+1} + a_{m+2}}{3}$.

If $g_{\sigma(1)} > 0$, then $\omega(1) = \frac{\frac{g_{\sigma(1)}}{d(\sigma(1))}}{3 + \frac{1}{d(\sigma(1))}} < \frac{g_{\sigma(1)}}{d(\sigma(1))} \leq g_{\sigma(1)}$. Hence, the largest index p for which $g_{\sigma(p)} > \omega(p)$, i.e., the index p^*, exists. Next, we prove that the projection $z = P_{T_f^k}(a)$ is of the described form.

Using the fact that $z_f = z_{m+1} = z_{m+2}$, the vector z can be obtained as the solution of the following optimization problem, where we restrict to the support of the constraints in T_f^k.

$$\begin{aligned} \min_z \quad & \sum_{i \in K_k} \sum_{e \in \delta(i) \setminus \{f\}} ||a_e - z_e||_2^2 + ||a_f - z_f||_2^2 + ||a_{m+1} \\ & - z_f||_2^2 + ||a_{m+2} - z_f||_2^2 \\ \text{s.t.} \quad & \sum_{e \in \delta(i)} z_e \geq z_f \qquad \forall i \in K_k. \end{aligned} \tag{19}$$

Let λ_i, $i \in K_k$, denote the dual variables corresponding to the constraints of (19). We further denote by $V(f)$ the two vertices in G adjacent to $f \in E$. Then, the KKT optimality conditions for (19) are as follows

$$2(z_e - a_e) - \lambda_i = 0 \qquad \forall e \in \delta(i) \setminus \{f\}, \forall i \in K_k \tag{20}$$

$$6z_f - 2(a_f + a_{m+1} + a_{m+2}) + \sum_{i \in K_k \setminus V(f)} \lambda_i = 0 \tag{21}$$

$$\sum_{e \in \delta(i)} z_e \geq z_f \qquad \forall i \in K_k \tag{22}$$

$$\lambda_i (z_f - \sum_{e \in \delta(i)} z_e) = 0 \qquad \forall i \in K_k \tag{23}$$

$$\lambda_i \geq 0 \qquad \forall i \in K_k. \tag{24}$$

It follows from (20) and (21) that we have

$$\begin{aligned} z_f &= \frac{a_f + a_{m+1} + a_{m+2}}{3} - \frac{1}{6} \sum_{i \in K_k \setminus V(f)} \lambda_i, \quad \text{and} \\ z_e &= a_e + \frac{1}{2}\lambda_i \qquad \forall e \in \delta(i) \setminus \{f\}, \ \forall i \in K_k. \end{aligned} \tag{25}$$

Suppose $K^* \subseteq K_k$ is the set of vertices for which $\lambda_i > 0$ at an optimal solution of (19). The complementary slackness constraints (23) then imply that $z_f = \sum_{e \in \delta(i)} z_e$ for all $i \in K^* \setminus V(f)$ and $\sum_{e \in \delta(i) \setminus \{f\}} z_e = 0$ for $i \in K^* \cap V(f)$. Note that $|K^* \cap V(f)| \leq 1$ since K_k is an independent set in G. By exploiting (25) and $\sum_{j \in K_k \setminus V(f)} \lambda_j = \sum_{j \in K^* \setminus V(f)} \lambda_j$, these equations can be rewritten to

$$\begin{aligned} & \frac{a_f + a_{m+1} + a_{m+2}}{3} - \frac{1}{6} \sum_{j \in K^* \setminus V(f)} \lambda_j = \sum_{e \in \delta(i)} \left(a_e + \frac{1}{2}\lambda_i\right) \\ \iff \quad & \lambda_i = \frac{2}{d(i)} \left(\frac{a_f + a_{m+1} + a_{m+2}}{3} - \sum_{e \in \delta(i)} a_e - \frac{1}{6} \sum_{j \in K^* \setminus V(f)} \lambda_j \right) \\ \iff \quad & \lambda_i = \frac{2}{d(i)} \left(g_i - \frac{1}{6} \sum_{j \in K^* \setminus V(f)} \lambda_j \right) \end{aligned} \tag{26}$$

for all $i \in K^* \setminus V(f)$. Summing the latter equations over all $i \in K^* \setminus V(f)$ yields

$$\sum_{i \in K^* \setminus V(f)} \lambda_i = 2 \sum_{i \in K^* \setminus V(f)} \frac{g_i}{d(i)} - \frac{1}{3} \sum_{i \in K^* \setminus V(f)} \frac{1}{d(i)} \sum_{j \in K^* \setminus V(f)} \lambda_j,$$

or equivalently, $\sum_{i \in K^* \setminus V(f)} \lambda_i = \frac{2 \sum_{i \in K^* \setminus V(f)} \frac{g_i}{d(i)}}{1 + \frac{1}{3} \sum_{i \in K^* \setminus V(f)} \frac{1}{d(i)}}$. After substitution into (26), we obtain

$$\lambda_i = \frac{2}{d(i)} \left(g_i - \frac{\sum_{i \in K^* \setminus V(f)} \frac{g_i}{d(i)}}{3 + \sum_{i \in K^* \setminus V(f)} \frac{1}{d(i)}} \right) > 0 \tag{27}$$

for all $i \in K^* \setminus V(f)$. For each $i \in (K_k \setminus K^*) \setminus V(F)$, we have $\lambda_i = 0$. The inequalities (22) for these i then read

$$\sum_{e \in \delta(i)} a_e \geq \frac{a_f + a_{m+1} + a_{m+2}}{3} - \frac{\sum_{i \in K^* \setminus V(f)} \frac{g_i}{d(i)}}{3 + \sum_{i \in K^* \setminus V(f)} \frac{1}{d(i)}},$$

or equivalently,

$$g_i - \frac{\sum_{i \in K^* \setminus V(f)} \frac{g_i}{d(i)}}{3 + \sum_{i \in K^* \setminus V(f)} \frac{1}{d(i)}} \leq 0. \tag{28}$$

By combining (27) and (28) we obtain the following optimality conditions on the dual variables λ concerning the indices in $K_k \setminus V(F)$

$$\begin{cases} \frac{2}{d(i)} \left(g_i - \frac{\sum_{i \in K^* \setminus V(f)} \frac{g_i}{d(i)}}{3 + \sum_{i \in K^* \setminus V(f)} \frac{1}{d(i)}} \right) > 0 & \text{for all } i \in K^* \setminus V(f), \\ g_i - \frac{\sum_{i \in K^* \setminus V(f)} \frac{g_i}{d(i)}}{3 + \sum_{i \in K^* \setminus V(f)} \frac{1}{d(i)}} \leq 0 & \text{for all } i \in (K_k \setminus K^*) \setminus V(F). \end{cases} \tag{29}$$

We conclude from the conditions (29) that the support of λ restricted to $K_k \setminus V(f)$ always consists of the vertices for which g_i lies above a certain threshold value. To find this threshold value, we sort the g_i's in non-increasing order and check all possible candidate sets for $K^* \setminus V(f)$ corresponding to the first r entries in this sorted list. Let $\sigma : [n_k] \to K_k$ denote an according sorting permutation, i.e., σ is bijective and fulfills $g_{\sigma(1)} \geq g_{\sigma(2)} \geq \cdots \geq g_{\sigma(n_k)}$. For each candidate set $\{\sigma(1), \ldots, \sigma(p)\} \subseteq K_k \setminus V(f)$, it suffices to check whether $g_{\sigma(p)}$ is strictly larger than the candidate threshold value

$$\omega(p) := \frac{\sum_{j=1}^{p} \frac{g_{\sigma(j)}}{d(\sigma(j))}}{3 + \sum_{j=1}^{p} \frac{1}{d(\sigma(j))}}.$$

If p^* is the largest index for which this holds, then this candidate set equals $K^* \setminus V(f)$. The existence of such a p^* is guaranteed by the existence of a solution to the projection problem (19).

Finally, we need to address the optimality conditions for all $i \in K_k \cap V(f)$. In case $i \in K^* \cap V(f)$, we have $\lambda_i > 0$, and due to complementary slackness (23) it holds that

$$0 = \sum_{e \in \delta(i) \setminus \{f\}} z_e = \sum_{e \in \delta(i) \setminus \{f\}} \left(a_e + \frac{1}{2}\lambda_i\right), \text{or equivalently,}$$

$$\lambda_i = -\frac{2}{d(i) - 1} \sum_{e \in \delta(i) \setminus \{f\}} a_e > 0.$$

We note here that we may w.l.o.g. assume that $d(i) > 1$. Namely, if $d(i) = 1$, then the set $\delta(i) \setminus \{f\}$ is empty, hence λ_i will not appear anywhere in (25), making this dual variable redundant.

For the case $i \in (K_k \setminus K^*) \cap V(f)$, and hence $\lambda_i = 0$, condition (23) with (22) reads as $\sum_{e \in \delta(i) \setminus \{f\}} a_e \geq 0$. Combining both cases, we obtain the following optimality conditions for $i \in K_k \cap V(F)$:

$$\begin{cases} \sum_{e \in \delta(i) \setminus \{f\}} a_e < 0 & \text{for } i \in K^* \cap V(f), \\ \sum_{e \in \delta(i) \setminus \{f\}} a_e \geq 0 & \text{for } i \in (K_k \setminus K^*) \cap V(f). \end{cases}$$

Altogether, the equations (25) then imply

$$z_e = \begin{cases} \frac{a_f + a_{m+1} + a_{m+2}}{3} - \omega(p^*) & \text{if } f \in \{e, m+1, m+2\}, \\ a_e + \frac{1}{d(i)}(g_i - \omega(p^*)) & \text{if } e \in \delta(i) \setminus \{f\} \text{ and } i \in K^* \setminus V(f), \\ a_e - \frac{1}{d(i)-1} \sum_{e \in \delta(i) \setminus \{f\}} a_e & \text{if } e \in \delta(i) \setminus \{f\} \text{ and } i \in K_k \cap V(f) \text{ with } \sum_{e \in \delta(i) \setminus \{f\}} a_e < 0, \\ a_e & \text{otherwise.} \end{cases}$$

□

It follows from Lemma 5 that the projection onto T_f^k involves both a sorting and an enumeration of a list of n_k elements. Hence, the worst-case time complexity is $O(n_k \log n_k)$.

In fact, for computational purposes, we are not going to project on $\mathcal{Y}_{RLT}$ but iteratively add violated cuts only. For that, we denote by $C \subseteq V \times E$ the set of violated cuts that we to add to $\mathcal{Y}$, where an element (i, f) represents the cut $\sum_{e \in \delta(i)} y_{ef} \geq y_f$. We further define analogously to $\mathcal{Y}_{RLT}$ the polyhedral set

$$\mathcal{Y}_C := \left\{ \widetilde{Y} \in \mathbb{R}^{(m+1) \times (m+1)} \ : \ \widetilde{Y} = \begin{pmatrix} Y & y \\ y^\top & 1 \end{pmatrix}, \right.$$

$$\operatorname{diag}(Y) = y,\ \sum_{e\in\delta(i)} y_{fe} \geq y_f \quad \forall (i,f)\in\mathcal{C}\Big\}.$$

The projection follows the same idea as explained above for the projection onto $\mathcal{Y}_{RLT}$, but in this case, instead of partitioning the vertex set V into independent sets, we can partition the constraints in $\mathcal{C}$ for each edge f separately. For a fixed f, we partition the vertices occurring together with f in $\mathcal{C}$ into independent sets $K_1^f, \dots, K_{N_f}^f$. Note that the number of independent sets N_f for an edge will probably be way smaller than the number of colors needed to color the whole graph, which can, in the worst case of a complete graph, be the number of vertices. Furthermore, as mentioned above, it is possible to project independently over each row $f \in E$, which allows us to parallelize this step. Hence, we cluster the cut constraints in $\mathcal{C}_k = \{(i,f)\in\mathcal{C} : f \in E,\ i \in K_k^f\}$ for $1 \leq k \leq N_{\max}$ with $N_{\max} := \max\{N_f : f \in E\}$ and obtain $\mathcal{Y}_{\mathcal{C}} = \mathcal{Y} \cap \left(\bigcap_{k=1}^{N_{\max}} \mathcal{Y}_{\mathcal{C}_k}\right)$, where we can easily project onto $\mathcal{Y}_{\mathcal{C}_k}$ using Lemma 5. A pseudocode for the Cyclic Dykstra projection algorithm to project onto $\mathcal{Y}_{\mathcal{C}}$ can be found in Algorithm 1. To compute the lower

Algorithm 1 Dykstra's cyclic projection algorithm to project onto $\mathcal{Y}_{\mathcal{C}}$.

Input: matrix M, cuts $\mathcal{C}$, ε_{proj}
Output: the projection $\mathcal{P}_{\mathcal{Y}_{\mathcal{C}}}(M)$ of M onto $\mathcal{Y}_{\mathcal{C}}$

cluster $\mathcal{C}$ into $\{\mathcal{C}_1, \dots, \mathcal{C}_{N_{\max}}\}$
initialize $X = M$, $P = \mathbf{0}$, $Q_1 = \cdots = Q_{N_{\max}} = \mathbf{0}$
repeat
  $X_{old} = X$
  $X_{tmp} = X + P$
  $X = \mathcal{P}_{\mathcal{Y}}(X_{tmp})$
  $P = X_{tmp} - X$
  for $k = 1, \dots, N_{\max}$ **do**
    $X_{tmp} = X + Q_k$
    $X = \mathcal{P}_{\mathcal{Y}_{\mathcal{C}_k}}(X_{tmp})$
    $Q_k = X_{tmp} - X$
  end for
until $\|X_{old} - X\| < \varepsilon_{proj}$
return X

bound (17) with a PRSM algorithm, we first compute the DNN bound (16) with the PRSM, as explained in the previous subsection. Then, we separate violated cuts from the current solution and add the `ncutsmax` most violated ones to $\mathcal{C}$. We then proceed to compute (16) with the additional new cuts in $\mathcal{C}$ with the PRSM and use the solution from before for a warm-start. This process of separating and adding new cuts to $\mathcal{C}$ in an outer loop is iterated until one of the stopping criteria is met. Algorithm 2 provides a pseudocode for the described algorithm.

5.3. Stopping criteria and post-processing of the PRSM algorithm

In this subsection, we briefly discuss the stopping criteria and the post-processing phase of our PRSM algorithm.

5.3.0.1. Stopping criteria. We use several criteria to decide when to stop the inner and outer iterations of Algorithm 2. The main stopping criteria for the inner while loop is when the primal and dual errors satisfy

$$\max\left\{\frac{\left\|\widetilde{Y}^{k+1} - W R^{k+1} W^\top\right\|_F}{1+\left\|\widetilde{Y}^{k+1}\right\|_F},\ \tau\frac{\left\|W^\top(\widetilde{Y}^k - \widetilde{Y}^{k+1})W\right\|_F}{1+\left\|S^{k+1}\right\|_F}\right\} \leq \varepsilon_{PRSM},$$

cf. (Boyd et al., 2011). We further stop the inner iterations when the maximum number of total PRSM iterations or a time limit is reached. In that case, we compute a valid dual bound as described below, and stop the algorithm.

Algorithm 2 PRSM algorithm to compute lower bounds on the QMST.

Input: graph $G = (V, E)$, cost matrix $\widetilde{Q}$
Output: (valid) lower bound LB

initialize $\widetilde{Y}^0$, S^0, τ, γ_1, γ_2, set $\mathcal{C} = \emptyset$ ▷ cf. Section 6
compute W, e.g., apply QR decomposition to $\begin{pmatrix}(n-1)\mathbf{I}_m & \mathbf{1}_m\end{pmatrix}^\top$
$k = 0$
while no stopping criteria met **do**
  while no stopping criteria met **do**
    $R^{k+1} = \mathcal{P}_{\mathcal{R}}(W^\top(\widetilde{Y}^k + \frac{1}{\tau}S^k)W)$
    $S^{\frac{k+1}{2}} = S^k + \gamma_1\tau(\widetilde{Y}^k - W R^{k+1} W^\top)$
    $\widetilde{Y}^{k+1} = \mathcal{P}_{\mathcal{Y}_{\mathcal{C}}}\left(W R^{k+1} W^\top - \frac{1}{\tau}\left(\widetilde{Q} + S^{\frac{k+1}{2}}\right)\right)$
    $S^{k+1} = S^{\frac{k+1}{2}} + \gamma_2\tau(\widetilde{Y}^{k+1} - W R^{k+1} W^\top)$
    $k = k + 1$
  end while
  compute a valid lower bound LB from S^k ▷ cf. Section 5.3
  separate violated cuts and add the `ncutsmax` most violated ones to $\mathcal{C}$
  cluster the cuts in $\mathcal{C}$
end while
return LB

For the outer loop, we have the following possible stopping criteria. If an upper bound is known, the algorithm stops as soon as the obtained valid lower bound closes the gap. We further stop the algorithm if the number of new violated cuts found is below a certain threshold `ncutsmin`. If the improvement of the valid lower bound compared to the valid lower bound of the previous outer iteration is smaller than `epslbimprov`, we stop the algorithm as well. And finally, we stop after a maximum of `noutermax` outer iterations.

5.3.0.2. Valid lower bound. The value obtained as an output of Algorithm 2 does not necessarily provide a lower bound for the problem, as the convergence of the PRSM is typically not monotonic, and one stops the algorithm earlier. Therefore, it is necessary to perform a post-processing procedure to obtain a valid lower bound. We apply the approach presented in (Li et al., 2021). The safe lower bound derived by this method is then given by

$$\operatorname{lb}(S^{\text{out}}) = \min_{\widetilde{Y}\in\mathcal{Y}_{\mathcal{C}}} \langle \widetilde{Q} + S^{\text{out}}, \widetilde{Y}\rangle - n\lambda_{\max}(W^\top S^{\text{out}} W),$$

where S^{out} denotes the dual matrix variable resulting from (an early stop of) the PRSM. The computation of this lower bound boils down to computing the largest eigenvalue and solving a linear program. Similarly, one can obtain a valid lower bound from the PRSM algorithm that solves (16), by replacing $\mathcal{Y}_{\mathcal{C}}$ with $\mathcal{Y}$, see (15), in the above expression.

6. Computational results

We implemented[1] our algorithm in Julia (Bezanson et al., 2017) version 1.10.0. For solving the linear program to compute a valid lower bound, we are using the solver HiGHS (Huangfu & Hall, 2018) with the modeling language JuMP (Lubin et al., 2023). The projection onto $\mathcal{C}_k$ is multithreaded, as described in Section 5.2. For computing the eigendecomposition to perform the projection onto $\mathcal{R}$, see (14), we use the Julia function `eigen`. As we use the Intel Math Kernel Library, this function is executed multithreaded. All computations were carried out on an AMD EPYC 7343 with 16 cores with 4.00GHz and 1024GB RAM, operated under Debian GNU/Linux 11.

[1] The code can be found on https://github.com/melaniesi/QMST.jl.

Table 1
Comparison to averaged results on lower bounds for CP instances. Note that the experiments were carried out on different machines, see Section 6 for the details.

	This study				Guimarães et al. (2020)			
	DNN		DNN + CUTS		LAGN		LAGP	
	gap (%)	time (s)	gap (%)	time (s)	gap (%)	time (s)	gap (%)	time (s)
n = 10	4.64	0.08	2.03	9.31	3.78	0.41	3.43	1.03
n = 15	5.54	0.41	4.24	18.77	5.25	4.81	8.54	6.59
n = 20	5.91	1.02	5.32	27.30	6.43	30.64	13.32	32.62
n = 25	7.52	2.32	6.83	35.80	8.72	122.87	17.56	239.23
n = 30	8.66	5.85	8.37	54.10	11.86	358.20	21.91	892.74
n = 35	9.87	7.62	9.68	58.70	15.96	897.69	24.03	3582.20
n = 40	11.33	14.34	11.19	68.10	23.53	1597.73	28.72	6050.58
n = 45	11.88	27.13	11.78	84.40	27.34	3195.97	29.51	16297.23
n = 50	13.08	45.58	13.03	103.22	31.45	6030.00	31.99	31953.35
d = 33 %	4.93	1.97	3.97	23.31	4.84	145.89	9.80	191.42
d = 67 %	8.99	7.02	8.17	44.53	14.28	1124.67	20.96	3968.84
d = 100 %	12.22	25.79	12.01	85.39	25.65	2808.88	28.92	15524.93
class = 1	9.08	8.14	8.51	21.87	17.93	551.84	20.66	5270.2
class = 2	10.96	15.69	10.23	45.99	17.27	1629.16	23.73	8583.7
class = 3	4.35	13.49	3.63	106.83	7.63	1625.35	12.14	4462.49
class = 4	10.47	9.05	9.84	29.61	16.87	1632.91	23.04	7930.53

6.1. Parameter setting

We initialize the matrices, penalty parameters, and step lengths as follows. As starting values for the matrices, we choose $S^0 = \mathbf{0}$ and

$$\widetilde{Y}^0 = \begin{pmatrix} \frac{(n-1)}{m}\mathbf{I} + \frac{(n-1)(n-2)}{m(m-1)}(\mathbf{J}-\mathbf{I}) & \frac{(n-1)}{m}\mathbf{1} \\ \frac{(n-1)}{m}\mathbf{1}^\top & 1 \end{pmatrix}$$

Based on the results of computational tests, we have determined the values for the penalty parameter τ and step lengths. We set the step length parameters to $\gamma_1 = 0.9$, $\gamma_2 = 1$. For the penalty parameter, let $q_{\max} := \max\{\mathrm{tr}(Q), \|Q\|_F\}$ and $q_{\min} := \min\{\mathrm{tr}(Q), \|Q\|_F\}$, we then set

$$\tau = \begin{cases} \sqrt{\frac{q_{\min}}{m+1}}\|Q\|_F & \text{if } \frac{q_{\max}}{q_{\min}} < 1.2, \\ \sqrt{\frac{q_{\max}}{q_{\min}}}\|Q\|_F & \text{else.} \end{cases}$$

We run our algorithm for all instances with $\varepsilon_{PRSM} = 10^{-4}$ and the parameter ε_{proj} is set to 10^{-5}. Violated cuts are considered if the violation is greater than 10^{-3} and after each outer iteration, the `ncutsmax` $= m$ most violated cuts are added. No further cuts are added if the improvement of the lower bound is smaller than `epslbimprov` $= 10^{-3}$ or the number of new violated cuts found is less than `ncutsmin` $= 10$. The maximum wall-clock time for running our algorithm is set to 3 hours per instance, and the maximum number of total iterations is set to 10 000. We set the number of maximum outer iterations to `noutermax` $= 10$.

6.2. Benchmark instances

We test our algorithm on the following three benchmark sets. The first benchmark set OP was introduced by Öncan and Punnen (2010). The benchmark set consists of 3 different classes, each consisting of 160 instances on complete graphs: the OPsym, OPvsym and OPesym instances. The OPsym instances have diagonal entries chosen uniformly from [100], and the off-diagonal values are uniformly distributed at random in [20]. For instances in the class OPvsym, the diagonal values are uniformly distributed in [10 000], and the off-diagonal values $Q_{\{i,j\},\{k,l\}}$ are computed as $w(i)w(j)w(k)w(l)$, where $w : V \to [10]$ assigns to each vertex in the graph a uniformly distributed weight at random in [10]. The cost matrix for instances of the type OPesym is constructed in the following way. First, the vertex coordinates are randomly chosen in the box $[0, 100] \times [0, 100]$, and the edges are represented as straight lines connecting vertices. The edge cost Q_{ee} is then set as the length of the edge e, and the interaction cost between two edges e and f is computed as the Euclidean distance between the midpoints of e and f. For each of those test sets, they randomly generated 10 instances each for $n \in \{6, 7, \ldots, 17, 18\} \cup \{20, 30, 50\}$. We do not include the benchmark instances of type OPesym and $n = 20$ in our study, as we were unable to locate the correct instances[2]

The second family of benchmark instances CP was introduced by Cordone and Passeri (2008). The benchmark set consists of 108 instances divided into 4 classes, specifying the sets from which the diagonal and off-diagonal values of the cost matrix are chosen uniformly at random. For each pair of the number of vertices $n \in \{10, 15, 20, 25, 30, 35, 40, 45, 50\}$, density $d \in \{33\,\%, 67\,\%, 100\,\%\}$ and class, one random graph was generated. The values of the cost matrix Q are uniformly distributed on the sets as listed below.

class	CP1	CP2	CP3	CP4
diagonal values	[10]	[10]	[100]	[100]
off-diagonal values	[10]	[100]	[10]	[100]

The last benchmark set SV was introduced by Sotirov and Verchére in their recent paper (Sotirov & Verchére, 2024). It consists of 24 instances, with one random graph for each pair of $n \in \{10, 12, 14, 16, 18, 20, 25, 30\}$ and $d \in \{33\,\%, 67\,\%, 100\,\%\}$. They constructed the cost matrices in such a way that for a given maximum cost for the diagonal entries, and a maximum cost for the off-diagonal entries, 10 % of the edges have high interaction costs with each other (between 90 and 100 % of the maximum off-diagonal cost) and low interaction costs with the rest (between 20 and 40 % of the maximum off-diagonal cost). The other 90 % of edges have an interaction cost of between 50 and 70 % of the maximum off-diagonal cost with each other. The diagonal entries are chosen to be between 0 and 20 % of the maximum diagonal cost.

6.3. Bounds from the literature

We compare our computational results to lower bounds from (Guimarães et al., 2020; Pereira et al., 2015b; Sotirov & Verchére, 2024). The upper bounds on the SV instances are taken from Sotirov and Ver-

[2] In the benchmark set https://data.mendeley.com/datasets/cmnh9xc6wb/1, the instances indicated as type OPesym for $n = 20$ are the OPvsym for $n = 6$.

Table 2
Results for CP1.

Instance				DNN			DNN + CUTS					
n	*d* (%)	*m*	UB	LB	gap (%)	time (s)	LB	gap (%)	time (s)	iterations	cuts	closed (%)
10	33	14	350	341.08	2.55	0.02	349.97	0.01	0.74	502	28	99.63
10	67	30	255	242.83	4.77	0.14	251.43	1.40	10.08	816	48	70.69
10	100	45	239	226.33	5.30	0.19	228.96	4.20	13.87	820	60	20.75
15	33	34	745	706.16	5.21	0.13	719.04	3.48	8.73	810	91	33.17
15	67	70	659	619.40	6.01	0.36	628.71	4.60	7.16	796	140	23.53
15	100	105	620	585.24	5.61	0.67	587.03	5.32	4.70	286	105	5.12
20	33	62	1379	1318.35	4.40	0.18	1328.45	3.67	4.99	530	143	16.66
20	67	127	1252	1171.25	6.45	0.99	1179.66	5.78	10.62	511	221	10.41
20	100	190	1174	1072.25	8.67	0.92	1074.01	8.52	7.81	166	168	1.73
25	33	99	2185	2101.49	3.82	0.75	2134.53	2.31	12.94	787	289	39.56
25	67	201	2023	1863.59	7.88	0.81	1866.65	7.73	8.23	163	228	1.92
25	100	300	1943	1705.87	12.20	2.44	1706.67	12.16	10.60	155	138	0.34
30	33	143	3205	3073.14	4.11	1.58	3089.28	3.61	11.21	584	334	12.24
30	67	291	2998	2709.48	9.62	1.94	2712.29	9.53	10.25	151	284	0.97
30	100	435	2874	2475.81	13.86	4.31	2476.25	13.84	16.19	165	102	0.11
35	33	196	4474	4235.79	5.32	1.49	4251.43	4.97	12.83	381	410	6.56
35	67	398	4147	3715.44	10.41	2.66	3717.16	10.37	12.92	133	253	0.40
35	100	595	4000	3399.70	15.01	11.80	3399.99	15.00	30.73	177	90	0.05
40	33	257	5945	5573.59	6.25	1.24	5587.07	6.02	12.56	188	456	3.63
40	67	522	5567	4878.53	12.37	6.82	4879.76	12.34	24.27	142	227	0.18
40	100	780	5368	4457.25	16.97	21.13	4457.41	16.96	51.71	189	78	0.02
45	33	326	7521	7065.52	6.06	2.97	7080.87	5.85	15.74	255	584	3.37
45	67	663	7161	6210.43	13.27	11.60	6210.88	13.27	30.85	142	170	0.05
45	100	990	6944	5668.59	18.37	43.86	5668.64	18.37	77.93	193	33	0.00
50	33	404	9393	8737.94	6.97	2.32	8749.60	6.85	14.86	149	564	1.78
50	67	820	8958	7668.14	14.40	19.19	7668.62	14.39	48.21	156	152	0.04
50	100	1225	8713	7029.42	19.32	79.40	7029.45	19.32	119.82	206	32	0.00
Average					9.08			8.51				13.07

Table 3
Results for CP2.

Instance				DNN			DNN + CUTS					
n	*d* (%)	*m*	UB	LB	gap (%)	time (s)	LB	gap (%)	time (s)	iterations	cuts	closed (%)
10	33	14	3122	2958.98	5.22	0.01	3115.06	0.22	2.88	579	28	95.75
10	67	30	2042	1929.22	5.52	0.08	2027.39	0.72	25.75	923	55	87.05
10	100	45	1815	1681.60	7.35	0.14	1697.01	6.50	24.15	949	50	11.55
15	33	34	6539	6108.39	6.59	0.05	6236.79	4.62	9.54	666	83	29.82
15	67	70	5573	5182.26	7.01	0.25	5273.86	5.37	10.19	651	120	23.44
15	100	105	5184	4802.29	7.36	0.51	4819.98	7.02	19.41	461	126	4.63
20	33	62	12,425	11770.26	5.27	0.12	11860.87	4.54	9.11	549	140	13.84
20	67	127	10,893	10251.15	5.89	0.66	10316.10	5.30	13.62	389	182	10.12
20	100	190	10,215	9138.20	10.54	1.53	9156.76	10.36	14.36	346	153	1.72
25	33	99	19,976	19156.48	4.10	0.56	19499.99	2.38	35.03	1054	282	41.92
25	67	201	18,251	16563.59	9.25	1.36	16595.43	9.07	19.68	323	221	1.89
25	100	300	17,411	14772.45	15.15	3.40	14782.12	15.10	28.02	346	140	0.37
30	33	143	29,731	28301.88	4.81	0.78	28469.60	4.24	22.82	496	326	11.74
30	67	291	27,581	24303.73	11.88	2.86	24330.85	11.78	22.24	318	259	0.83
30	100	435	26,146	21699.39	17.01	9.23	21703.81	16.99	35.44	361	90	0.10
35	33	196	42,305	39405.07	6.85	0.98	39551.95	6.51	19.02	337	400	5.06
35	67	398	38,490	33520.06	12.91	5.37	33537.58	12.87	35.03	318	258	0.35
35	100	595	36,723	30027.26	18.23	20.46	30030.25	18.22	63.99	388	83	0.04
40	33	257	56,237	51995.88	7.54	1.83	52136.64	7.29	19.07	284	437	3.32
40	67	522	51,851	44247.99	14.66	11.95	44260.68	14.64	53.48	313	238	0.17
40	100	780	49,817	39565.62	20.58	40.64	39567.19	20.57	113.05	413	63	0.02
45	33	326	70,603	65914.02	6.64	2.38	66041.56	6.46	32.93	310	589	2.72
45	67	663	66,889	56511.11	15.52	22.95	56515.55	15.51	63.23	313	151	0.04
45	100	990	64,840	50526.76	22.07	88.58	50527.18	22.07	152.35	419	42	0.00
50	33	404	88,942	81835.33	7.99	3.48	81933.27	7.88	27.64	262	557	1.38
50	67	820	84,020	69935.17	16.76	37.26	69939.94	16.76	101.24	342	145	0.03
50	100	1225	81,858	62875.12	23.19	166.17	62875.44	23.19	268.61	453	30	0.00
Average					10.96			10.23				12.89

chére (2024), for all other instances, the upper bounds are taken from a collection of best known results in the literature[3].

The bounds from (Guimarães et al., 2020), called LAGN and LAGP, are used in the to-date best exact algorithm for the QMSTP. Those bounds are obtained from two different ways of dualizing an SDP relaxation of QMSTP. For LAGN, the semidefiniteness constraint is dualized, and a subgradient method is used to compute the optimum. Whereas for computing LAGP, there is no semidefiniteness constraint present, but a

[3] https://homes.di.unimi.it/cordone/research/qmst.html

Table 4
Results for CP3.

Instance				DNN			DNN + CUTS					
n	d (%)	m	UB	LB	gap (%)	time (s)	LB	gap (%)	time (s)	iterations	cuts	closed (%)
10	33	14	646	625.91	3.11	0.01	625.91	3.11	0.01	130	0	0.00
10	67	30	488	460.23	5.69	0.04	487.03	0.20	3.68	538	26	96.52
10	100	45	426	422.02	0.93	0.09	424.27	0.41	6.67	556	18	56.51
15	33	34	1236	1203.32	2.64	0.15	1225.21	0.87	11.52	1311	79	66.99
15	67	70	966	931.32	3.59	0.25	946.11	2.06	39.28	902	89	42.66
15	100	105	975	953.87	2.17	1.59	961.97	1.34	89.76	1860	145	38.32
20	33	62	1972	1906.69	3.31	0.29	1924.45	2.41	21.52	1088	178	27.19
20	67	127	1792	1742.91	2.74	1.52	1762.12	1.67	101.72	1303	311	39.13
20	100	190	1544	1514.39	1.92	4.25	1520.24	1.54	122.27	1212	248	19.78
25	33	99	2976	2861.20	3.86	1.01	2904.43	2.40	25.98	1353	281	37.66
25	67	201	2546	2468.53	3.04	5.70	2487.99	2.28	100.34	1415	487	25.13
25	100	300	2471	2371.04	4.05	7.83	2382.68	3.57	152.13	811	438	11.65
30	33	143	4070	4001.92	1.67	2.55	4027.98	1.03	39.17	1278	460	38.28
30	67	291	3649	3522.91	3.46	6.93	3544.28	2.87	131.82	921	635	16.95
30	100	435	3483	3293.54	5.44	12.41	3305.31	5.10	189.36	684	665	6.21
35	33	196	5423	5261.95	2.97	3.76	5302.59	2.22	59.07	1361	622	25.24
35	67	398	4981	4757.51	4.49	7.89	4767.61	4.28	113.69	593	627	4.52
35	100	595	4770	4451.46	6.68	21.04	4454.47	6.61	298.89	503	593	0.94
40	33	257	6925	6708.96	3.12	5.77	6753.53	2.48	73.30	1099	849	20.63
40	67	522	6456	6110.47	5.35	12.16	6122.99	5.16	136.20	482	776	3.62
40	100	780	6208	5716.76	7.91	37.61	5718.10	7.89	183.36	446	422	0.27
45	33	326	8720	8484.37	2.70	9.77	8517.33	2.32	81.91	1051	1035	13.99
45	67	663	8225	7692.48	6.47	18.71	7700.56	6.38	147.23	373	884	1.52
45	100	990	7827	7160.41	8.52	67.09	7161.31	8.50	288.73	384	382	0.14
50	33	404	10,717	10341.87	3.50	12.76	10367.75	3.26	89.62	851	1101	6.90
50	67	820	10,100	9366.40	7.26	28.69	9368.84	7.24	157.12	286	660	0.33
50	100	1225	9836	8767.89	10.86	94.33	8768.14	10.86	220.08	292	166	0.02
Average					4.35			3.63				22.26

Table 5
Results for CP4.

Instance				DNN			DNN + CUTS					
n	d (%)	m	UB	LB	gap (%)	time (s)	LB	gap (%)	time (s)	iterations	cuts	closed (%)
10	33	14	3486	3391.98	2.70	0.02	3485.78	0.01	2.19	562	28	99.77
10	67	30	2404	2261.59	5.92	0.09	2348.16	2.32	10.89	676	50	60.79
10	100	45	2197	2051.05	6.64	0.16	2082.04	5.23	10.77	713	56	21.23
15	33	34	7245	6790.59	6.27	0.16	6958.02	3.96	10.37	816	81	36.84
15	67	70	6188	5769.57	6.76	0.29	5862.17	5.27	9.26	595	153	22.13
15	100	105	5879	5449.98	7.30	0.47	5470.56	6.95	5.27	310	118	4.80
20	33	62	13,288	12666.64	4.68	0.18	12766.55	3.92	4.47	523	138	16.08
20	67	127	11,893	11096.91	6.69	0.76	11178.80	6.01	7.61	387	187	10.29
20	100	190	11,101	9951.75	10.35	0.88	9971.71	10.17	9.54	204	171	1.74
25	33	99	21,176	20227.51	4.48	0.77	20589.60	2.77	14.67	994	288	38.18
25	67	201	19,207	17574.38	8.50	0.89	17610.67	8.31	9.87	180	228	2.22
25	100	300	18,370	15807.56	13.95	2.29	15817.10	13.90	12.11	170	139	0.37
30	33	143	31,077	29621.42	4.68	1.31	29783.85	4.16	14.20	539	325	11.16
30	67	291	28,777	25518.14	11.32	1.87	25546.39	11.23	11.24	160	271	0.87
30	100	435	27,314	22912.94	16.11	24.38	22917.52	16.10	145.29	1151	93	0.10
35	33	196	43,629	40846.14	6.38	1.11	41011.82	6.00	12.59	306	392	5.95
35	67	398	39,660	34968.45	11.83	3.48	34986.17	11.78	14.38	147	255	0.38
35	100	595	38,049	31466.73	17.30	11.39	31469.65	17.29	31.31	192	90	0.04
40	33	257	57,874	53626.37	7.34	4.24	53763.25	7.10	72.15	882	441	3.22
40	67	522	53,592	45901.66	14.35	6.33	45914.47	14.33	24.59	157	235	0.17
40	100	780	51,229	41229.31	19.52	22.34	41231.11	19.52	53.43	206	73	0.02
45	33	326	72,676	67799.91	6.71	3.21	67966.40	6.48	17.31	265	582	3.41
45	67	663	68,737	58407.29	15.03	10.40	58412.38	15.02	29.33	154	165	0.05
45	100	990	66,508	52425.30	21.17	44.09	52425.78	21.17	75.26	207	40	0.00
50	33	404	91,009	83901.38	7.81	2.98	84027.09	7.67	16.53	162	564	1.77
50	67	820	86,231	72050.92	16.44	19.43	72055.97	16.44	47.69	168	153	0.04
50	100	1225	83,838	65003.10	22.47	80.94	65003.46	22.47	127.26	222	33	0.00
Average					10.47			9.84				12.65

Table 6
Results for SV instances.

Instance				DNN			DNN + CUTS					
n	*d* (%)	*m*	UB	LB	gap (%)	time (s)	LB	gap (%)	time (s)	iterations	cuts	closed (%)
10	33	11	4217	4207.86	0.22	0.01	4207.86	0.21	0.01	105	0	0.00
10	67	25	3981	3960.66	0.51	0.45	3974.47	0.15	6.20	1155	47	67.88
10	100	45	3930	3906.64	0.59	0.24	3914.25	0.38	14.53	945	78	32.56
12	33	28	6141	6115.80	0.41	0.08	6124.29	0.26	2.15	619	42	33.70
12	67	51	6050	6018.40	0.52	0.19	6030.75	0.31	17.54	803	71	39.09
12	100	66	6051	6003.50	0.79	0.32	6012.51	0.63	22.41	1003	144	18.98
14	33	29	8736	8691.84	0.51	0.12	8733.38	0.02	4.85	1111	81	94.07
14	67	53	8606	8580.96	0.29	0.25	8593.99	0.14	11.48	956	133	52.03
14	100	91	8513	8458.85	0.64	1.28	8482.15	0.35	32.85	1206	221	43.03
16	33	36	11,735	11685.13	0.43	0.14	11706.79	0.24	3.51	929	102	43.43
16	67	87	11,610	11536.13	0.64	1.07	11551.71	0.50	12.39	949	217	21.09
16	100	120	11,516	11468.40	0.41	2.51	11480.52	0.30	39.79	1675	280	25.46
18	33	51	15,125	15059.14	0.44	0.26	15093.78	0.20	11.70	1253	167	52.59
18	67	108	15,019	14947.73	0.47	1.59	14958.75	0.40	14.80	1101	229	15.46
18	100	153	14,943	14883.24	0.40	3.25	14889.90	0.35	17.47	850	247	11.14
20	33	60	19,057	18975.25	0.43	0.28	19001.60	0.29	9.09	1191	177	32.23
20	67	127	18,830	18757.29	0.39	2.40	18766.64	0.33	9.66	1008	245	12.85
20	100	190	18,812	18699.67	0.60	2.31	18704.25	0.57	14.02	398	261	4.08
25	33	104	30,747	30660.41	0.28	1.01	30685.44	0.20	14.76	1127	333	28.91
25	67	195	30,554	30416.45	0.45	3.09	30435.81	0.39	19.97	672	553	14.07
25	100	300	30,405	30202.33	0.67	4.64	30207.32	0.65	13.46	271	263	2.46
30	33	152	45,184	45048.50	0.30	2.87	45077.39	0.23	17.08	1056	499	21.32
30	67	273	44,989	44756.82	0.52	3.52	44765.07	0.50	12.29	324	362	3.55
30	100	435	44,847	44428.13	0.93	4.66	44432.08	0.92	13.72	138	227	0.94
Average					0.49			0.36				27.96

Table 7
Results for OPsym instances.

Instance			DNN			DNN + CUTS					
n	*m*	UB	LB	gap (%)	time (s)	LB	gap (%)	time (s)	iterations	cuts	closed (%)
6	15	258.40	249.24	3.55	0.01	251.42	2.70	0.10	112.3	1.0	23.84
7	21	326.80	307.63	5.87	0.06	315.81	3.36	11.19	454.4	12.8	42.68
8	28	438.50	420.37	4.14	0.04	428.77	2.22	7.46	388.7	19.1	46.35
9	36	534.90	505.99	5.40	0.06	525.22	1.81	18.09	523.0	33.6	66.50
10	45	653.90	627.00	4.11	0.09	645.03	1.36	35.16	756.7	60.5	67.03
11	55	785.90	747.23	4.92	0.13	764.22	2.76	45.54	784.2	71.8	43.92
12	66	918.50	875.83	4.65	0.20	892.54	2.83	49.01	809.7	93.9	39.15
13	78	1067.10	1018.91	4.52	0.28	1036.33	2.88	68.12	913.6	121.5	36.16
14	91	1249.80	1200.89	3.91	0.58	1220.08	2.38	84.08	984.5	157.0	39.24
15	105	1390.20	1327.62	4.50	0.77	1348.64	2.99	79.95	998.6	186.3	33.59
16	120	1629.30	1542.28	5.34	0.92	1567.60	3.79	65.70	986.8	227.7	29.09
17	136	1823.80	1723.14	5.52	1.41	1746.23	4.25	103.45	1046.7	253.9	22.93
18	153	2981.00	1963.13	34.15	1.80	1985.06	33.41	104.87	950.1	279.8	2.15
20	190	2572.70	2415.88	6.10	2.61	2428.77	5.59	99.94	846.0	299.3	8.22
30	435	6015.90	5324.15	11.50	10.38	5326.24	11.46	102.24	426.9	228.9	0.30
50	1225	17616.90	14104.59	19.94	168.73	14104.71	19.94	440.47	387.5	44.0	0.00
Average				8.01			6.48				31.32

Table 8
Results for OPesym instances.

Instance			DNN			DNN + CUTS					
n	*m*	UB	LB	gap (%)	time (s)	LB	gap (%)	time (s)	iterations	cuts	closed (%)
6	15	541.20	421.57	22.10	0.01	499.51	7.70	3.80	279.8	9.5	65.15
7	21	783.70	568.79	27.42	0.03	743.62	5.11	27.81	526.2	24.3	81.35
8	28	1020.10	706.73	30.72	0.04	956.62	6.22	55.29	548.3	45.6	79.74
9	36	1356.00	1113.97	17.85	0.05	1313.50	3.13	74.30	695.3	57.9	82.44
10	45	1427.10	1044.20	26.83	0.07	1362.01	4.56	118.12	643.2	84.2	83.00
11	55	1545.10	1122.77	27.33	0.12	1431.81	7.33	223.74	873.6	100.0	73.18
12	66	1901.60	1420.73	25.29	0.13	1815.21	4.54	432.63	993.1	160.9	82.03
13	78	2175.30	1684.29	22.57	0.18	2051.16	5.71	430.93	943.8	174.7	74.72
14	91	2527.90	1911.59	24.38	0.46	2367.42	6.35	720.47	1107.6	256.4	73.96
15	105	2588.80	2173.45	16.04	0.52	2426.66	6.26	468.08	810.8	200.6	60.96
16	120	2980.10	2360.16	20.80	1.07	2765.42	7.20	834.69	1214.3	291.8	65.37
17	136	3372.20	2327.66	30.98	1.00	3165.01	6.14	1933.56	1678.0	445.3	80.16
18	153	3569.00	2645.32	25.88	1.06	3281.36	8.06	1947.79	1403.7	421.7	68.86
30	435	8056.70	6114.86	24.10	17.30	7174.26	10.95	10045.11	1244.4	1056.8	54.56
50	1225	15788.80	13030.28	17.47	406.93	13333.52	15.55	10923.68	800.5	1225.0	10.99
Average				23.98			6.99				69.10

Table 9
Results for OPvsym instances.

Instance			DNN			DNN + CUTS					
n	m	UB	LB	gap (%)	time (s)	LB	gap (%)	time (s)	iterations	cuts	closed (%)
6	15	16273.90	14868.34	8.64	0.01	14961.49	8.06	0.14	56.8	1.0	6.63
7	21	19625.70	18483.49	5.82	0.02	18483.49	5.82	0.02	49.2	0.0	0.00
8	28	27039.40	23571.44	12.83	0.02	25283.58	6.49	22.61	757.0	9.1	49.37
9	36	22769.90	19600.00	13.92	0.04	21514.48	5.51	5.74	267.2	7.5	60.40
10	45	25743.80	22937.70	10.90	0.06	24522.61	4.74	19.27	494.3	17.4	56.48
11	55	29325.60	24918.41	15.03	0.07	28930.89	1.35	30.67	350.4	38.2	91.04
12	66	32577.80	29185.79	10.41	0.13	32481.73	0.29	115.64	674.7	43.4	97.17
13	78	40488.50	34246.69	15.42	0.13	39782.08	1.74	73.07	385.7	64.8	88.68
14	91	44240.40	38383.15	13.24	0.52	44071.71	0.38	156.78	797.9	64.4	97.12
15	105	50821.60	42399.17	16.57	0.46	50113.95	1.39	155.86	534.6	100.7	91.60
16	120	41940.20	35936.50	14.31	1.26	41356.18	1.39	290.43	1023.2	85.7	90.27
17	136	41819.00	32332.91	22.68	0.79	41374.89	1.06	592.39	1183.9	143.8	95.32
18	153	46130.20	36741.44	20.35	1.27	45077.44	2.28	645.68	1142.1	143.0	88.79
20	190	55326.20	41076.20	25.76	2.08	55048.51	0.50	1530.01	1362.4	184.8	98.05
30	435	78999.90	55898.20	29.24	21.51	76507.06	3.16	10781.08	1944.7	426.6	89.21
50	1225	165419.60	107324.67	35.12	633.63	154950.21	6.33	10835.89	1787.8	1064.8	81.98
Average				16.89			3.16				73.88

semi-infinite reformulation together with polyhedral cutting planes is solved using a bundle method.

The lower bounds VS1 and VS2 were introduced by Sotirov and Verchére (2024). These lower bounds are based on an extended formulation of the minimum quadratic spanning tree problem and are strengthened by facet defining inequalities of the Boolean Quadric polytope. The lower bound VS2 is stronger than VS1.

Pereira et al. (2015b) solved several benchmark problems of sizes up to 50 vertices using a RLT based relaxation RLT1. RLT1 is an incomplete first-level RLT relaxation and is computed by dualizing the symmetry constraint, applying the GL procedure, and using a subgradient algorithm. Another RLT based bound among the strongest relaxations in the literature is RLT2, presented in (Rostami & Malucelli, 2015). Rostami and Malucelli (2015) use a dual-ascent procedure for computing their relaxation based on the second-level of RLT.

6.4. Computational results

We first present a comparison to the results from (Guimarães et al., 2020), where the authors also compute SDP bounds. Their computations were carried out on a machine with 32 GB RAM and two E5645 Intel Xeon processors, with six 2.40GHz cores each.

The structure of Table 1 is analogous to Table 3 in (Guimarães et al., 2020) and reads as follows. The rows are grouped into 3 blocks, each reporting the results averaged over all CP instances with the same property as specified in the first column of the table. The first block of rows averages over instances of the same size, the second averages the results over the densities of the graphs, and the last block averages over the different classes of the CP instances. In the second column of Table 1, we report the average gap obtained by the valid lower bound obtained with our PRSM algorithm when stopping after the first outer iteration, cf. (16). We compute the relative gap between that lower bound (LB_{DNN}) and the best known upper bound (UB) from the literature using $100(UB - \mathrm{LB}_{DNN})/UB$. We remark here that the same gap was calculated by Guimarães et al. (2020).[4] In the third column, we report the average wall clock time in seconds needed to compute this lower bound. In column 4, we report the average gap obtained by the bound returned by Algorithm 2, cf. (17), and in column 5, the average time needed to compute this bound. In the sixth and seventh column, we list the average gaps and computation times for the bound LAGN of (Guimarães et al., 2020), which is used in the best up-to-date exact algorithm for the QMSTP. The average gaps and computation times of LAGP, the second lower bound introduced in (Guimarães et al., 2020), are given in the last two columns of Table 1.

The results in Table 1 show that for the CP instances, our lower bounds are, on average, significantly stronger than the SDP bounds LAGN and LAGP. Except for the instances with $n \in \{10, 15\}$, the average computation times for solving our relaxations are smaller than those reported for computing SDP bounds LAGN and LAGP. The average time to compute the DNN + CUTS bound, that is (17), over all CP instances is 51 seconds, compared to 1 360 and 5 652 seconds for LAGN and LAGP, respectively. More significant difference in the computation times and relative gaps can be seen for larger instances. Even if the machine used for this study is stronger than the one used for computing LAGN and LAGP, it is evident that with an increasing size n, the time for computing LAGN and LAGP grows rapidly. One can also observe that the less dense the instances are, the smaller the average relative gap. Furthermore, the effect of adding cuts is more significant for sparse graphs than for dense graphs. Guimarães et al. (2020, Table 4) compare their bounds to RLT1 (Pereira et al., 2015b), which can be computed approximately three times faster than LAGN but yields much weaker bounds. The average gap of bound RLT2 (Rostami & Malucelli, 2015) over all instances of size $n \leq 35$ for each of the four CP classes is at least three times larger than our reported average gaps for (16). Overall, Table 1 shows that, especially for larger CP instances, our bounds are significantly stronger and faster to compute than any other bounds.

The latter conclusion at first might seem counterintuitive, as our relaxations (10) and (11) are theoretically weaker than the bound from Guimarães et al. (2020), where the algorithms LAGN and LAGP are based on. However, it follows from Table 1 that this is not the case in practice. We believe that the difference might be caused by the fact that the algorithms LAGN and LAGP are stopped before reaching their theoretical optimal solution. This hypothesis is strengthened by the fact that the reported gaps of LAGN and LAGP are different, although they result from the same theoretical SDP bound. Since our approach is able to solve the relaxation up to high precision, our bounds turn out to be stronger than the ones reported by Guimarães et al. (2020). Finally, due to our postprocessing step explained in Section 5.3, we have full certainty that our approach provides a valid lower bound even if the algorithm is stopped at a low precision.

In the Tables 2 to 6 we report the computational results for all benchmark instances of the test sets CP and SV. The first four columns give details about the instance as the number of vertices, the edge density, the number of edges and an upper bound on the QMST. The next three columns report the valid lower bound (16) obtained after the first

[4] There was a typo in that paper that claims differently, but our statement can be easily verified.

outer loop of our PRSM algorithm, the relative gap to the upper bound $100(UB - \mathrm{LB}_{DNN})/UB$, and the wall clock time in seconds needed to compute that bound. The last six columns outline the computational results of our algorithm to compute (17). In columns 8 to 10, we provide the valid lower bound returned by our algorithm, the relative gap, and the wall clock time needed to compute the lower bound. The next two columns list the total number of iterations and the total number of cuts added. In the last column, we report the relative gap closed by adding the RLT-type cuts to the DNN relaxation (16). This performance measurement is computed as $100(\mathrm{LB}_{DNN+CUTS} - \mathrm{LB}_{DNN})/(\mathrm{UB} - \mathrm{LB}_{DNN})$, where $\mathrm{LB}_{\mathrm{DNN}}$ refers to the lower bound (16) reported in column 5 and $\mathrm{LB}_{\mathrm{DNN+CUTS}}$ is the lower bound (17) reported in column 8 in each table. This metric gives information on how much the gap to the upper bound was improved. For a quick overview, we print the average gaps at the end of each table.

Tables 2 to 5 show that especially for CP instances with $n \geq 30$ vertices and edge density 100 % there were only a few violated cuts found. Hence, the relative improvement of the DNN relaxation by adding those cuts was only marginal. One can further observe that the improvement of the relative gap and the relative gap closed, is better for smaller instances. For larger instances, adding cuts such as the RLT-type of the cut-set constraints for subsets S of size 2 and larger, might further improve the DNN bounds.

Table 6 presents the results of our algorithm for the benchmark set SV introduced by Sotirov and Verchére (2024). To the best of our knowledge, there are no results on LAGN, LAGP, and RLT2 for this benchmark set. The by far best lower bound up to date for the SV instance set was VS2. Our DNN relaxation bound without cuts outperforms VS2 for all instances, with the number of edges $m \geq 45$, except for the instance with $n = 12$ and $d = 67\,\%$. Both our relaxations yield a relative gap of less than 1 %. The relative gap of VS2 ranges between 0 and 16.4 %. The maximum runtime to compute the DNN bound for these instances is less than 5 seconds, whereas computation time for bound VS2 of $n = 30$ and $d = 100\,\%$ was reported to be 45 minutes. Computing the DNN bound with cuts is faster than the reported time to compute VS2 for all instances with more than 80 edges.

Tables 7 to 9 read similarly to the tables for the CP and SV benchmark sets but the results are averaged over all instances of the same size. Again, to the best of our knowledge, we are not aware of any detailed and complete results for LAGN and LAGP on the OP benchmark set.

Table 7 reports the results obtained for the benchmark set OPsym. The lower bound (17) with cuts outperforms VS2 for $n \geq 10$, and RLT2 for $n \geq 8$ with the exception of $n = 18$, where the average relative gap for RLT2 is reported to be 33 % and is 33.41 % for the DNN bound with cuts. For $n = 50$, no bounds were reported. One can observe that the absolute improvement by adding RLT cuts to (16) for $n \geq 9$ is approximately 20.

Table 8 shows that for the benchmark set OPesym adding the RLT-type cuts to (16) yields a substantial improvement of the relative gap. The DNN lower bound with cuts yields better bounds compared to VS2 but is clearly dominated by RLT1, giving an average relative gap between 0.2 % and 1.7 % for instances with $n \leq 30$.

Sotirov and Verchére (2024) report that the relative gap of the VS1 lower bound is less than or equal 0.2 % for all instances of the class OPvsym. Although, on average, not many violated cuts to be added were found, the averaged relative bound closed is above 49 % for all instances except that with $n \in \{6, 7\}$, where on average only 0.5 violated cuts were found. Considering the instances with $n \geq 11$, the average relative bound closed is even above 80 %.

The time limit of 3 hours was reached by all instances from OPesym and OPvsym of size $n = 50$ and almost all of those instances of size $n = 30$. The higher computational costs for those two classes of benchmark instances can be explained, among other things, by the high number of clusters $N_{\max}$, cf. Section 5.2. The number of clusters has a direct effect on the computation time of Dykstra's algorithm, which accounts for a substantial part of the overall computation time. The average number of clusters needed for the OPvsym and OPesym instances are 6.43 and 6.38, whereas the average over all other benchmark instances is 3.26. Note that for those two classes of instances, added RTL-type constraints significantly improve lower bounds. Additionally, as for the CP3 instances, one can observe the higher number of iterations until convergence of the algorithm compared to other classes in our benchmark sets.

7. Conclusion

This paper provides a mixed-integer semidefinite programming formulation for the quadratic minimum spanning tree problem. This formulation includes only one connectivity constraint, which is a linear matrix inequality based on the algebraic connectivity of trees. By exploiting the MISDP formulation, we derive a DNN relaxation for the QMSTP. We also derive the cut-set and RLT-type constraints as Chvátal-Gomory cuts of the MISDP by applying a CG procedure for mixed integer conic programming. The RLT-type constraints are added to the DNN relaxation, resulting in a strengthened DNN relaxation. An iterative cutting plane Peaceman-Rachford splitting method is designed to compute the DNN relaxation with the RLT-type constraints of the QMSTP efficiently.

The computational experiments on the benchmark instances from the literature demonstrate that our bounds significantly outperform existing bounds in quality and computation time. While other approaches struggled to compute bounds for larger instances, we compute strong bounds in short time.

Given these results, incorporating our new bounds in a branch-and-bound algorithm would be the obvious next step for further research. Another topic for future research would be to incorporate additional RLT-type cut-set constraints to further strengthen the DNN relaxation.

CRediT authorship contribution statement

Frank de Meijer: Writing – original draft, Validation, Software, Methodology, Formal analysis, Conceptualization; **Melanie Siebenhofer:** Writing – original draft, Validation, Software, Methodology, Formal analysis, Conceptualization; **Renata Sotirov:** Writing – original draft, Validation, Software, Methodology, Formal analysis, Conceptualization; **Angelika Wiegele:** Writing – original draft, Validation, Software, Methodology, Formal analysis, Conceptualization.

Acknowledgments

This research was funded in part by the Austrian Science Fund (FWF) [10.55776/DOC78]. For open access purposes, the authors have applied a CC BY public copyright license to any author-accepted manuscript version arising from this submission.

Supplementary material

Supplementary material associated with this article can be found in the online version at 10.1016/j.ejor.2025.10.051.

References

Assad, A., & Xu, W. (1992). The quadratic minimum spanning tree problem. *Naval Research Logistics*, *39*(3), 399–417.

Bauschke, H. H., & Koch, V. R. (2015). Projection methods: Swiss army knives for solving feasibility and best approximation problems with halfspaces. In *Infinite products of operators and their applications* (pp. 1–40). AMS, Providence, RI (*vol. 636*). Contemp. Math. https://doi.org/10.1090/conm/636/12726

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and TrendsÂ® in Machine Learning*, *3*(1), 1–122. https://doi.org/10.1561/2200000016

Boyle, J. P., & Dykstra, R. L. (1986). A method for finding projections onto the intersection of convex sets in hilbert spaces. In R. Dykstra, T. Robertson, & F. T. Wright (Eds.), *Advances in order restricted statistical inference* (pp. 28–47). New York, NY.

Çezik, M. T., & Iyengar, G. N. (2005). Cuts for mixed 0–1 conic programming. *Mathematical Programming*, *104*, 179–202.

Chiang, T.-C., Liu, C.-H., & Huang, Y.-M. (2007). A near-optimal multicast scheme for mobile ad hoc networks using a hybrid genetic algorithm. *Expert Systems with Applications*, *33*(3), 734–742.

Chou, W., & Kershenbaum, A. (1974). A unified algorithm for designing multidrop teleprocessing networks. *IEEE Transactions on Communications*, *22*, 1762–1772.

Chvátal, V. (1973). Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, *4*, 305–337.

Condat, L. (2016). Fast projection onto the simplex and the ℓ_1 ball. *Mathematical Programming*, *158*(1–2), 575–585. https://doi.org/10.1007/s10107-015-0946-6

Cordone, R., & Passeri, G. (2008). Heuristic and exact approaches to the quadratic minimum spanning tree problem. In *Seventh cologne twente workshop on graphs and comb. opt., gargano, italy, 13–15 may, 2008* (pp. 52–55). University of Milan.

Cordone, R., & Passeri, G. (2012). Solving the quadratic minimum spanning tree problem. *Applied Mathematics and Computation*, *218*, 11597–11612.

Ćustić, A., Zhang, R., & Punnen, A. P. (2018). The quadratic minimum spanning tree problem and its variations. *Discrete Optimization*, *27*, 73–87.

Cvetković, D., Čangalović, M., & Kovačević-Vujčić, V. (1999). Semidefinite programming methods for the symmetric traveling salesman problem. In G. Cornuj́ols, R. E. Burkard, & G. J. Woeginger (Eds.), *Integer programming and combinatorial optimization (IPCO 1999)*. Springer, Berlin, Heidelberg (*vol. 1610*). Lecture Notes in Comput. Sci.

Drusvyatskiy, D., & Wolkowicz, H. (2017). The many faces of degeneracy in conic optimization. *Foundations and Trends in Optimization*, *3*(2), 77–170.

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, *23*(2), 298–305. http://eudml.org/doc/12723.

Gilmore, P. C. (1962). Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the Society for Industrial and Applied Mathematics*, *10*(2), 305–313.

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, *64*(5), 275–278.

Grone, R., & Merris, R. (1990). Ordering trees by algebraic connectivity. *Graphs and Combinatorics*, *6*, 229–237.

Guimarães, D. A., Cunha, A. S. d., & Pereira, D. L. (2020). Semidefinite programming lower bounds and branch-and-bound algorithms for the quadratic minimum spanning tree problem. *The European Journal of Operational Research*, *280*(1), 46–58.

He, B., Ma, F., & Yuan, X. (2016). Convergence study on the symmetric version of ADMM with larger step sizes. *SIAM Journal on Imaging Sciences*, *9*(3), 1467–1501. https://doi.org/10.1137/15M1044448

Hu, H., Sotirov, R., & Wolkowicz, H. (2023). Facial reduction for symmetry reduced semidefinite and doubly nonnegative programs. *Mathematical Programming*, *200*(1), 475–529. https://doi.org/10.1007/s10107-022-01890-9

Huangfu, Q., & Hall, J. A. J. (2018). Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, *10*(1), 119–142. https://doi.org/10.1007/s12532-017-0130-5

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. In *Proc. amer. math. soc., 7*.

Lawler, E. L. (1963). The quadratic assignment problem. *Management Science*, *9*(4), 586–599.

Li, X., Pong, T. K., Sun, H., & Wolkowicz, H. (2021). A strictly contractive peaceman-rachford splitting method for the doubly nonnegative relaxation of the minimum cut problem. *Computational Optimization and Applications*, *78*(3), 853–891. https://doi.org/10.1007/s10589-020-00261-4

Lions, P.-L., & Mercier, B. (1979). Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, *16*(6), 964–979. http://www.jstor.org/stable/2156649.

Lubin, M., Dowson, O., Dias Garcia, J., Huchette, J., Legat, B., & Vielma, J. P. (2023). JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. *Mathematical Programming Computation*, *15*, 581–589. https://doi.org/10.1007/s12532-023-00239-3

De Meijer, F., & Sotirov, R. (2024a). The chvátal-gomory procedure for integer SDPs with applications in combinatorial optimization. *Mathematical Programming*, *209*, 323–395. https://doi.org/10.1007/s10107-024-02069-0

De Meijer, F., & Sotirov, R. (2024b). On integrality in semidefinite programming for discrete optimization. *SIAM Journal on Optimization*, *34*(1).

De Meijer, F., Sotirov, R., Wiegele, A., & Zhao, S. (2023). Partitioning through projections: Strong SDP bounds for large graph partition problems. *Computers & Operations Research*, *151*, 106088. https://www.sciencedirect.com/science/article/pii/S0305054822003185. https://doi.org/https://doi.org/10.1016/j.cor.2022.106088

Oliveira, D. E., Wolkowicz, H., & Xu, Y. (2018). ADMM For the SDP relaxation of the QAP. *Mathematical Programming Computation*, *10*, 631–658. https://doi.org/10.1007/s12532-018-0148-3

Öncan, T., & Punnen, A. P. (2010). The quadratic minimum spanning tree problem: A lower bounding procedure and an efficient search algorithm. *Computers & Operations Research*, *37*(10), 176–1773.

Palubeckis, G., Rubliauskas, D., & Targamadz, A. (2010). Metaheuristic approaches for the quadratic minimum spanning tree problem. *Information Technology and Control*, *29*, 257–268.

Peaceman, D. W., & Rachford, H. H. (1955). The numerical solution of parabolic and elliptic differential equations. *SIAM Journal on Applied Mathematics*, *3*(1), 28–41. http://www.jstor.org/stable/2098834.

Pereira, D. L., Gendreau, M., & Cunha, A. S. d. (2015a). Branch-and-cut and branch-and-cut-and-price algorithms for the adjacent only quadratic minimum spanning tree problem. *Networks*, *65*, 367–379.

Pereira, D. L., Gendreau, M., & Cunha, A. S. d. (2015b). Lower bounds and exact algorithms for the quadratic minimum spanning tree problem. *Computers & Operations Research*, *63*, 149–160.

Prim, R. C. (1957). Shortest connection networks and some generalizations. *The Bell Systems Technical Journal*, *36*(6), 1389–1401.

Punnen, A. P. (2001). Combinatorial optimization with multiplicative objective function. *International Journal of Operations and Quantitative Management*, *7*, 205–209.

Rostami, B., & Malucelli, F. (2015). Lower bounds for the quadratic minimum spanning tree problem based on reduced cost computation. *Computers & Operations Research*, *64*, 178–188.

Sherali, H. D., & Adams, W. P. (1998). A reformulation-linearization technique for solving discrete and continuous nonconvex problems (vol. 31). SSBM.

Sotirov, R., & Verchére, Z. (2024). The quadratic minimum spanning tree problem: Lower bounds via extended formulations. *Vietnam Journal of Mathematics*, . https://doi.org/10.1007/s10013-024-00694-y.

Wen, Z., Goldfarb, D., & Yin, W. (2010). Alternating direction augmented lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, *2*(3), 203–230.