Facial Reduction for Symmetry Reduced Semidefinite and Doubly Nonnegative Programs

Hao Hu∗ Renata Sotirov† Henry Wolkowicz‡

February 4, 2022

Abstract

We consider both facial reduction, FR, and symmetry reduction, SR, techniques for semidefinite programming, SDP. We show that the two together fit surprisingly well in an alternating direction method of multipliers, ADMM, approach. In fact, this approach allows for simply adding on nonnegativity constraints, and solving the doubly nonnegative, DNN, relaxation of many classes of hard combinatorial problems. We also show that the singularity degree remains the same after SR, and that the DNN relaxations considered here have singularity degree one, that is reduced to zero after FR. The combination of FR and SR leads to a significant improvement in both numerical stability and running time for both the ADMM and interior point approaches.

We test our method on various DNN relaxations of hard combinatorial problems including quadratic assignment problems with sizes of more than \( n = 500 \). This translates to a semidefinite constraint of order \( 250,000 \) and \( 625 \times 10^8 \) nonnegative constrained variables, before applying the reduction techniques.

Keywords: Semidefinite programming, group symmetry, facial reduction, quadratic assignment problem, vertex separator problem.

AMS subject classifications: 90C22, 90C25

Contents

1 Introduction 2
1.1 Outline .................................................. 3

2 Background 4
2.1 Semidefinite programming .................................. 4
2.1.1 Strict feasibility and facial reduction ................. 4
2.2 Group invariance and symmetry reduction, SR ........... 5
2.2.1 First symmetry reduction using \( X = B^*(x) \) .............. 7
2.2.2 Second symmetry reduction to block diagonal form using \( X = Q\tilde{B}^*(x)Q^T \) 8

∗Department of Combinatorics and Optimization Faculty of Mathematics, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1; h92hu@uwaterloo.ca.
†Department of Econometrics and Operations Research, Tilburg University, The Netherlands; r.sotirov@uvt.nl
‡Department of Combinatorics and Optimization Faculty of Mathematics, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1; Research supported by The Natural Sciences and Engineering Research Council of Canada; www.math.uwaterloo.ca/~hwolkowi.
1 Introduction

We consider two reduction techniques, facial and symmetry reduction, for semidefinite programming, SDP. We see that the exposing vector approach for facial reduction, FR, moves naturally onto the symmetry reduction, SR. We show that the combination of the two reductions fits surprisingly well in an alternating direction method of multipliers, ADMM, approach. In fact, the combination of FR and SR makes possible solving the doubly nonnegative, DNN, relaxations of many classes of hard combinatorial problems by using ADMM. The combination of facial and symmetry reduction also leads to a significant improvement in both numerical stability and running time for both the ADMM and interior point approaches. We test our method on various DNN relaxations of hard combinatorial problems including quadratic assignment problems (QAP) with sizes of more than \( n = 500 \), see Table 5.1. Note that the order of the symmetric matrix variable in the SDP relaxation of the QAP with \( n = 500 \), before applying the reduction
techniques, is 250,000. This yields approximately $625 \times 10^8$ nonnegatively constrained variables in the semidefinite constrained matrix of the original, not reduced, problem formulation.

Semidefinite programming can be viewed as an extension of linear programming where the nonnegative orthant is replaced by the cone of positive semidefinite matrices. Although there are many algorithms for solving semidefinite programs, they currently do not scale well and often do not provide high accuracy solutions. An early method for exploiting sparsity and reducing problem size was based on recognizing a chordal pattern in the matrices forming the SDP, see e.g., [21, 32], and the survey [64]. Another technique is that of symmetry reduction, a methodology, pioneered by Schrijver [52], that exploits symmetries in the data matrices that allows for the problem size to be reduced, often significantly. More details and surveys for SR are available in [2,13].

Without loss of generality, we consider the case where the primal problem has a finite optimal value. Then for linear programming, strong duality holds for both the primal and the dual problems. But, this is not the case for SDP, where the primal and/or the dual can be unattained, and one can even have a positive duality gap between the primal and dual optimal values. The usual constraint qualification to guarantee strong duality is the Slater condition, strict feasibility. Failure of the Slater condition may lead to theoretical and numerical problems when solving the SDP. Facial reduction, FR, introduced by Borwein and Wolkowicz [5–7], addresses this issue by projecting the minimal face of the SDP into a lower dimensional space. The literature for the theory and applications for FR is large. For a recent survey and theses see [18,46,55].

An earlier work [39] combines partial FR and SR for solving sum of square (SOS) programs. In particular, Löfberg [39] applies a partial FR via monomial basis selection and shows how to perform a partial SR via identification of sign-symmetries to obtain block-diagonal SOS programs. Examples in [39] verify the efficiency of the combined approach for SOS programs. For the connection between FR and monomial basis selection see [18,65].

In our paper, we assume that we know how to do FR and SR separately for the input SDP instance. Under this assumption, we show that it is possible to implement FR to the symmetry reduced SDP. The obtained reduced SDP is both facially reduced and symmetry reduced. And, it can be solved in a numerically stable manner by interior point methods. Moreover, the nonnegativity constraints can be added to the original SDP, and the resulting DNN relaxation can be solved efficiently, as the nonnegativities follow through, and are in fact simplified, to the reduced SDP program. Thus, in fact we solve the facially and symmetry reduced DNN relaxation using an alternating direction method of multipliers approach, ADMM. As a consequence, we are able to solve some huge DNN relaxations for highly symmetric instances of certain hard combinatorial problems, and we do so in a reasonable amount of time.

We include theoretical results on facial reduction, as well as on the singularity degree of both SDP and DNN relaxations. We present a view of FR for DNN from the ground set of the original hard combinatorial problem. The singularity degree indicates the importance of FR for splitting type methods. In particular we show that the singularity degree remains the same after SR, and that our applications all have singularity degree one, that get reduced to zero after FR.

1.1 Outline

In Section 2 we provide the background on using substitutions to first obtain FR and then symmetry and block diagonal SR. In Section 3 we show how to apply FR to the symmetry reduced SDP, and we also provide conditions such that the obtained SDP is strictly feasible. In fact, we show that the nonnegativity constraints are essentially unchanged and that we have strict feasibility for the reduced DNN relaxation. The results that singularity degree does not
change after SR are included as well, see Section 3.3. In Section 4 we show that the reduced DNN relaxation can be solved efficiently using an ADMM approach. In Section 5 we apply our result to two classes of problems: the quadratic assignment and graph partitioning problems. Concluding comments are in Section 6.

2 Background

2.1 Semidefinite programming

The semidefinite program, SDP, in standard form is

$$\min \{ \langle C, X \rangle \mid A(X) = b, \ X \succeq 0 \} ,$$  \hspace{1cm} (2.1)

where the linear transformation $A : S^n \rightarrow \mathbb{R}^m$ maps real $n \times n$ symmetric matrices to $\mathbb{R}^m$, and $X \succeq 0$ denotes positive semidefiniteness, i.e., $X \in S^n_+$. The set $S^n_+$ is the positive semidefinite, PSD, cone. In the case of a doubly nonnegative, DNN, relaxation, nonnegativity constraints, $X \geq 0$, are added to (2.1), i.e., we use the DNN cone denoted $\text{DNN} \equiv \text{DNN}^n = S^n_+ \cap \mathbb{R}^{n \times n}_+$. Without loss of generality, we assume that $A$ is onto. We let

$$(P_F) \hspace{1cm} F_X := \{ X \succeq 0 \mid A(X) = b \}$$  \hspace{1cm} (2.2)

denote the feasibility problem for this formulation with data $A, b, S^n_+$ from (2.1). Note that the linear equality constraint is equivalent to

$$A(X) = (\langle A_i, X \rangle) = (b_i) \in \mathbb{R}^m,$$

for some $A_i \in S^n, i = 1, \ldots, m$. The adjoint transformation $A^* : \mathbb{R}^m \rightarrow S^n$ is $A^*(y) = \sum_{i=1}^m y_i A_i$.

2.1.1 Strict feasibility and facial reduction

The standard constraint qualification to guarantee strong duality\(^1\) for the primal SDP is the Slater constraint qualification (strict feasibility)

$$\exists \hat{X} : \ A(\hat{X}) = b, \ \hat{X} \succ 0,$$

where $\hat{X} \succ 0$ denotes positive definiteness, i.e., $\hat{X} \in S^n_{++}$. For many problems where strict feasibility fails, one can exploit structure and facially reduce the problem to obtain strict feasibility, see e.g., [5,6] for the theory and [7] for the facial reduction algorithm. A survey with various views of FR is given in [18]. Facial reduction means that there exists a full column rank matrix $V \in \mathbb{R}^{n \times r}, r < n$, and the corresponding adjoint of the linear transformation $V : S^n \rightarrow S^r$ given in

$$V^*(R) = V R V^T, \ R \in S^r,$$

such that the substitution $X = V^*(R)$ results in the equivalent, regularized, smaller dimensional, problem

$$p^*_{\text{SDP}} = \min \{ \langle V^T CV, R \rangle \mid \langle V^T A_i V, R \rangle = b_i, \ i \in I \subseteq \{1, \ldots, m\}, \ R \in S^r_+ \}.$$  \hspace{1cm} (2.3)

\(^1\)Strong duality for the primal means a zero duality gap, $p^*_{\text{SDP}} = d^*_{\text{SDP}}$, and dual attainment.

\(^2\)FR generally results in the constraints becoming linearly dependent. Therefore, a linearly independent subset need only be used [55].
Strict feasibility holds for (2.3). The cone $V S^+_n V^T$ is the minimal face of the SDP, i.e., the smallest face of $S^n_+$ that contains the feasible set, $F_X$. And

$$\text{range}(V) = \text{range}(X), \forall X \in \text{relint}(F_X).$$

If $U \in \mathbb{R}^{n \times n - r}$ with $\text{range}(U) = \text{null}(V^T)$, then $W := UU^T$ is an exposing vector for the minimal face, i.e.,

$$X \text{ feasible} \implies WX = 0.$$

Let $F_R$ denote the feasible set for (2.3). We emphasize the following constant rank result for the FR substitution:

$$R \in F_R, \text{ rank}(R) = r \iff X = V^*(R) \in F_X, \text{ rank}(X) = r.$$

**Remark 2.1.** For a typical FR algorithm for finding the minimal face, at each iteration the dimension is strictly reduced, and at least one redundant linear constraint can be discarded, i.e., we need at most $\min\{m, n - 1\}$ iterations, e.g., [18, 56], [55, Theorem 3.5.4].

Note that FR can also be considered in the original space using rotations. Each step of FR involves finding an exposing vector $W = UU^T$ to the minimal face. Without loss of generality, we can assume that the matrix $Q = [V \ U]$ is orthogonal. Then the FR that reduces the size of the problem $X = V^*(R) = VRV^T$ can equivalently be considered as a rotation (orthogonal congruence):

$$X = [V \ U] \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} [V \ U]^T, \quad \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} V & U \end{bmatrix}^T X \begin{bmatrix} V & U \end{bmatrix},$$

i.e., after this rotation, we can discard zero blocks and reduce the size of the problem. We note that this can then be compared to the Constrained Set Invariance Conditions approach in [46], where a special projection is used to obtain the reduced problem. In addition, the approach in [46] performs the projections on the primal-dual problem thus maintaining the original optimal values of both. In contrast, we emphasize the importance of the primal problem as being the problem of interest. After FR we have a regularized primal problem (2.3) with optimal value the same as that of the original primal problem. In addition, the reduced program has the important property that the dual of the dual is the primal.

### 2.2 Group invariance and symmetry reduction, SR

We now find a substitution using the adjoint linear transformation $\tilde{B}^*(x)$ from (2.11) below, that provides the SR in block diagonal form. We first look at the procedure for simplifying an SDP that is invariant under the action of a symmetry group. This approach was introduced by Schrijver [52]; see also the survey [2]. The appropriate algebra isomorphism follows from the Artin-Wedderburn theory [66]. A more general framework is given in the thesis [46]. More details can be found in e.g., [12, 22, 24, 62].

Let $G$ be a nontrivial group of permutation matrices of size $n$. The commutant, $A_G$, (or centralizer ring) of $G$ is defined as the subspace

$$A_G := \{X \in \mathbb{R}^{n \times n} \mid PX = XP, \forall P \in G\}. \quad (2.4)$$

Thus, $A_G$ is the set of matrices that are self-permutation-congruent for all $P \in G$. An equivalent definition of the commutant is

$$A_G := \{X \in \mathbb{R}^{n \times n} \mid \mathcal{R}_G(X) = X\},$$
where
\[ \mathcal{R}_G(X) := \frac{1}{|G|} \sum_{P \in G} PXPT, \quad X \in \mathbb{R}^{n \times n}, \]  

(2.5)
is called the Reynolds operator (or group average) of \( G \). The operator \( \mathcal{R}_G \) is the orthogonal projection onto the commutant. The commutant \( A_G \) is a matrix \(*\)-algebra, i.e., it is a set of matrices that is closed under addition, scalar multiplication, matrix multiplication, and taking transposes. One may obtain a basis for \( A_G \) from the orbits of the action of \( G \) on ordered pairs of vertices, where the orbit of \( (u_i, u_j) \in \{0, 1\}^n \times \{0, 1\}^n \) under the action of \( G \) is the set \( \{(Pu_i, Pu_j) \mid P \in G\} \), and \( u_i \in \mathbb{R}^n \) is the \( i \)-th unit vector. In what follows, we denote

basis for \( A_G : \{B_1, \ldots, B_d\}, \quad B_i \in \{0, 1\}^{n \times n}, \quad \forall i. \) (2.6)

Let \( J \cong J_n \) (resp. \( I \cong I_n \)) denote the matrix of all ones (resp. the identity matrix) of appropriate size. The basis (2.6) forms a so-called coherent configuration.

**Definition 2.2 (coherent configuration).** A set of zero-one \( n \times n \) matrices \( \{B_1, \ldots, B_d\} \) is called a coherent configuration of rank \( d \) if it satisfies the following properties:

1. \( \sum_{i \in \mathcal{I}} B_i = I \) for some \( \mathcal{I} \subset \{1, \ldots, d\} \), and \( \sum_{i=1}^d B_i = J \);
2. \( B_i^T \in \{B_1, \ldots, B_d\} \) for \( i = 1, \ldots, d \);
3. \( B_iB_j \in \text{span}\{B_1, \ldots, B_d\}, \quad \forall i, j \in \{1, \ldots, d\} \).

In what follows we obtain that the Reynolds operator maps the feasible set \( \mathcal{F}_X \) of (2.1) into itself and keeps the objective value the same, i.e.,

\[ X \in \mathcal{F}_X \implies \mathcal{R}_G(X) \in \mathcal{F}_X \text{ and } \langle C, \mathcal{R}_G(X) \rangle = \langle C, X \rangle. \]

One can restrict optimization of an SDP problem to feasible points in a matrix \(*\)-algebra that contains the data matrices of that problem, see e.g., [14, 23]. In particular, the following result is known.

**Theorem 2.3 ([14], Theorem 4).** Let \( A_G \) denote a matrix \(*\)-algebra that contains the data matrices of an SDP problem as well as the identity matrix. If the SDP problem has an optimal solution, then it has an optimal solution in \( A_G \).

**Remark 2.4.** In [14], the authors consider complex matrix \(*\)-algebras. However in most applications, including applications in this paper, the data matrices are real symmetric matrices and \( A_G \) has a real basis, see Definition 2.2. Thus, we consider here the special real case. The authors in [14] also prove that if \( A_G \) has a real basis, and the SDP has an optimal solution, then it has a real optimal solution in \( A_G \). Real matrix \(*\)-algebras are also considered in [15, 16, 43].

In addition, Theorem 2.3 holds for DNN, i.e., we can move any nonnegativity constraints on \( X \) added to the SDP in Theorem 2.3 to simple nonnegativity constraints on \( x \) in (2.7), see e.g., [12, Pg 5].

Therefore, we may restrict the feasible set of the optimization problem to its intersection with \( A_G \). In particular, we can use the basis matrices and assume that

\[ X \in \mathcal{F}_X \cap A_G \iff X = \sum_{i=1}^d x_i B_i =: \mathcal{B}^*(x) \in \mathcal{F}_X, \text{ for some } x \in \mathbb{R}^d. \] (2.7)

From now on we assume that \( G \) is such that \( A_G \) contains the data matrices of (2.1).
Example 2.5 (Hamming Graphs). We now present an example of an algebra that we use later in our numerics.

The Hamming graph \( H(d,q) \) is the Cartesian product of \( d \) copies of the complete graph \( K_q \), with vertices represented by \( d \)-tuples of letters from an alphabet of size \( q \). The Hamming distance between vertices \( u \) and \( v \), denoted by \(|(u,v)|\), is the number of positions in which \( d \)-tuples \( u \) and \( v \) differ.

The matrices
\[
(B_i)_{u,v} := \begin{cases} 
1 & \text{if } |(u,v)| = i, \\
0 & \text{otherwise}
\end{cases}, \quad i = 0,\ldots,d
\]
form a basis of the Bose-Mesner algebra of the Hamming scheme, see [17]. In particular, \( B_0 = I \) is the identity matrix and \( B_1 \) is the adjacency matrix of the Hamming graph \( H(d,q) \) of size \( q^d \times q^d \). In cases, like for the Bose-Mesner algebra, when one of the basis elements equals the identity matrix, it is common to set the index of the corresponding basis element to zero. The basis matrices \( B_i \) can be simultaneously diagonalized by the real, orthogonal matrix \( Q \) given by
\[
Q_{u,v} = 2^{-\frac{d}{2}} (-1)^{u^T v}.
\]
The distinct elements of the matrix \( Q^T B_i Q \) equal \( K_i(j) \) \( (j = 0,\ldots,d) \) where
\[
K_i(j) := \sum_{h=0}^i (-1)^h (q-1)^{i-h} \binom{j}{h} \binom{d-j}{i-h}, \quad j = 0,\ldots,d,
\]
are Krawtchouk polynomials. We denote by \( \mu_j := \binom{d}{j}(q-1)^j \) the multiplicity of the \( j \)-th eigenvalue \( K_i(j) \). The elements of the character table \( P \in \mathbb{R}^{(d+1)\times(d+1)} \) of the Hamming scheme \( H(d,q) \), given in terms of the Krawtchouk polynomials, are
\[
p_{i,j} := K_i(j), \quad i,j = 0,\ldots,d.
\]
In the later sections, we use the following well-known orthogonality relations on the Krawtchouk polynomial, see e.g., [17],
\[
\sum_{j=0}^d K_r(j) K_s(j) \binom{d}{j} (q-1)^j = q^d \binom{d}{s} (q-1)^{s-r} \delta_{r,s}, \quad r,s = 0,\ldots,d, \tag{2.8}
\]
where \( \delta_{r,s} \) is the Kronecker delta function.

2.2.1 First symmetry reduction using \( X = B^*(x) \)

We now obtain our first reduced program using the substitution \( X = B^*(x) \). Note that the program is reduced in the sense that the feasible set can be smaller though the optimal value remains the same.

\[
p_{\text{SDP}} = \min \{ \langle B(C), x \rangle \mid (A \circ B^*)(x) = b, \quad B^*(x) \succeq 0 \}, \quad \text{(substitution } X = B^*(x)) \tag{2.9}
\]
Here, \( B \) is the adjoint of \( B^* \). In the case of a DNN relaxation, the structure of the basis in (2.6) allows us to equate \( X = B^*(x) \succeq 0 \) with the simpler \( x \succeq 0 \). This changes the standard doubly nonnegative cone into a splitting, the Cartesian product of the cones \( x \succeq 0, B^*(x) \succeq 0 \), see Remarks 2.4 and 5.1.

A matrix \( \ast \)-algebra \( \mathcal{M} \) is called basic if \( \mathcal{M} = \{ \bigoplus_{i=1}^d M \mid M \in \mathbb{C}^{m \times m} \} \), where \( \bigoplus \) denotes the direct sum of matrices. A very important decomposition result for matrix \( \ast \)-algebras is the following result due to Wedderburn.
Theorem 2.6 ([66]). Let \( \mathcal{M} \) be a matrix \(*\)-algebra containing the identity matrix. Then there exists a unitary matrix \( Q \) such that \( Q^* \mathcal{M} Q \) is a direct sum of basic matrix \(*\)-algebras.

The above result is derived for a complex matrix \(*\)-algebras. In [43], the authors study numerical algorithms for block-diagonalization of matrix \(*\)-algebras over \( \mathbb{R} \). Unfortunately, the Wedderburn decomposition described in the above theorem does not directly apply for \(*\)-algebras over reals. To demonstrate our approach in the section on numerical results we use known orthogonal matrices or a simple heuristics to obtain them.

To simplify our presentation, the matrix \( Q \) in Theorem 2.6 is assumed to be real orthogonal. (The case when \( Q \) is complex can be derived analogously.) Then, the matrices in the basis \( B_j, j = 1, \ldots, d \), can be mutually block-diagonalized by some orthogonal matrix \( Q \). More precisely, there exists an orthogonal matrix \( Q \) such that we get the following block-diagonal transformation on \( B_j \):

\[
\tilde{B}_j := Q^T B_j Q =: \text{Blkdiag}(\tilde{B}_j^k)_{k=1}^t, \quad \forall j = 1, \ldots, d. \tag{2.10}
\]

For \( Q^T X Q = \sum_{j=1}^d x_j \tilde{B}_j \), we now define the linear transformation for obtaining the block matrix diagonal form:

\[
\tilde{B}^*(x) := \sum_{j=1}^d x_j \tilde{B}_j = \begin{bmatrix} \tilde{B}_1^*(x) \\ \vdots \\ \tilde{B}_t^*(x) \end{bmatrix} =: \text{Blkdiag}(\tilde{B}_k^*(x))_{k=1}^t, \tag{2.11}
\]

where

\[
\tilde{B}_k^*(x) := \sum_{j=1}^d x_j \tilde{B}_j^k \in S_{n_k}^+
\]

is the \( k \)-th diagonal block of \( \tilde{B}^*(x) \), and the sum of the \( t \) block sizes \( n_1 + \ldots + n_t = n \). Thus, for any feasible \( X \) we get

\[
X = B^*(x) = Q \tilde{B}^*(x) Q^T \in \mathcal{F}_X.
\]

2.2.2 Second symmetry reduction to block diagonal form using \( X = Q \tilde{B}^*(x) Q^T \)

We now derive the second reduced program using the substitution \( X = Q \tilde{B}^*(x) Q^T \). The program is further reduced since we obtain the block diagonal problem

\[
p_{\text{SDP}}^* = \min \{ \langle \tilde{B}(\tilde{C}), x \rangle \mid (\tilde{A} \circ \tilde{B}^*)(x) = b, \quad \tilde{B}^*(x) \succeq 0 \}, \tag{2.12}
\]

where \( \tilde{C} = Q^T C Q \) and \( \tilde{A} \) is the linear transformation obtained from \( A \) as follows: \( \tilde{A}_j = Q^T A_j Q, \forall j \). We denote the corresponding blocks as \( \tilde{A}_j^k, \forall j = 1, \ldots, d, \forall k = 1, \ldots, t \).

We see that the objective in (2.12) satisfies

\[
\tilde{c} := \tilde{B}(\tilde{C}) = (\langle \tilde{B}_j, \tilde{C} \rangle) = (\langle B_j, C \rangle) \in \mathbb{R}^d.
\]

While the \( i \)-th row of the linear equality constraint in (2.12), \( Ax = b \), is

\[
b_i = (Ax)_i = (\langle \tilde{A} \circ \tilde{B}^* \rangle(x))_i = (\tilde{A}_i \circ \tilde{B}^*(x)) = (\tilde{B}(\tilde{A}_i), x).
\]

Therefore

\[
\tilde{A}_{ij} = (\tilde{B}(\tilde{A}_i))_j = (\tilde{B}_j, \tilde{A}_i) = \langle B_j, A_i \rangle, \quad i = 1, \ldots, m, \quad j = 1, \ldots, d. \tag{2.13}
\]
Without loss of generality, we can now define
\[ c := \tilde{c}, \quad A := \tilde{A}. \]
Moreover, just as for FR, the SR step can result in \( A \) not being full row rank (onto). We then have to choose a nice (well conditioned) submatrix that is full row rank and use the resulting subsystem of \( Ax = b \). We see below how to do this and simultaneously obtain strict feasibility.

We can now rewrite the SDP (2.1) as
\[
\text{p}^{*}\text{SDP} = \min \left\{ c^T x \mid Ax = b, \quad \tilde{B}^{*}(x) \succeq 0, \quad k = 1, \ldots, t \right\}. \tag{2.14}
\]
For many applications, there are repeated blocks. We then take advantage of this to reduce the size of the problem and maintain stability.

The program (2.14) is a symmetry reduced formulation of (2.1). We denote its feasible set and feasible slacks as
\[
\mathcal{F}_x := \{ x \mid \tilde{B}^*(x) \succeq 0, \quad Ax = b, \quad x \in \mathbb{R}^d \}, \quad \mathcal{S}_x := \{ \tilde{B}^*(x) \succeq 0 \mid Ax = b, \quad x \in \mathbb{R}^d \}. \tag{2.15}
\]
We denote the feasibility problem for this formulation with data \( \tilde{B}^*, A, b, S^n_+ \) of the feasible set \( \mathcal{F}_x = \mathcal{P}_{F_x} \). We bear in mind that \( \tilde{B}^*(x) \) is a block-diagonal matrix. But it is written as a single matrix for convenience in order to describe FR for the symmetry reduced program below.

Since \( \tilde{B}_1, \ldots, \tilde{B}_d \) are block diagonal, symmetric matrices, the symmetry reduced formulation is typically much smaller than the original problem, i.e.,
\[
x \in \mathbb{R}^d, \quad d \ll \sum_{i=1}^{d} t(n_i) \ll t(n),
\]
where \( t(k) = k(k+1)/2 \) is the triangular number.

### 3 Facial reduction for the symmetry reduced program

In this section, we show how to apply FR to the symmetry reduced SDP (2.14). The key is using the exposing vector view of facial reduction, [18]. Formally speaking, if an exposing vector of the minimal face of the SDP (2.1) is given, then we are able to construct a corresponding exposing vector of the minimal face of the symmetry reduced program (2.14). In fact, we show that all the exposing vectors of the symmetry reduced program can be obtained from the exposing vectors of the original program. In general, one can find exposing vectors from the original program by exploiting the structure. However, this is lost after the SR and results in a more difficult task in finding an exposing vector.

In addition, we follow the above theme on simply adding on the nonnegativities and extend many of the results to the DNN program. We include results on the singularity degree to emphasize the importance of FR for stability and that SR does not increase the singularity degree.

#### 3.1 Rank preserving

We begin with showing the maximum rank preserving properties of SR. Note that
\[
\max \{ \text{rank}(X) \mid X \in \mathcal{F}_X \} = \text{rank}(X), \quad \forall X \in \text{relint}(\mathcal{F}_X)
\]
\[
= \text{rank}(X), \quad \forall X \in \text{relint}(\text{face}(\mathcal{F}_X)),
\]
where face(\( \mathcal{F}_X \)) is the minimal face of \( S^n_+ \) containing the feasible set. We let \( F \leq K \) denote that \( F \) is a face of the cone \( K \).
**Theorem 3.1.** Let \( r = \max \{ \text{rank}(X) \mid X \in \mathcal{F}_X \} \). Then

\[
\begin{align*}
    r &= \max \left\{ \text{rank} \left( \frac{1}{|\mathcal{G}|} \sum_{P \in \mathcal{G}} P^T X P \right) \mid X \in \mathcal{F}_X \right\} \\
    &= \max \{ \text{rank}(X) \mid X \in \mathcal{F}_X \cap A_G \} \\
    &= \max \{ \text{rank}(\hat{B}^*(x)) \mid \hat{B}^*(x) \in S_x \}.
\end{align*}
\]

**Proof.** Let \( X \in \mathcal{F}_X \) be the matrix with maximum rank \( r \). Then \( X \) is in the relative interior of the minimal face \( f \subseteq S^n_\rho \) containing \( \mathcal{F}_X \), i.e.,

\[
    X \in \text{relint}(f) = [V \ U] \begin{bmatrix} S^n_+ & 0 \\ 0 & 0 \end{bmatrix} [V \ U]^T, \quad \text{for some orthogonal} \ [V \ U].
\]

The nonsingular congruence \( P^T X P \) is feasible for each \( P \in \mathcal{G} \), and also has rank \( r \). Note that

\[
    A, B \in S^n_\rho \implies \text{rank}(A + B) \geq \max\{\text{rank}(A), \text{rank}(B)\}.
\]

Therefore, applying the Reynolds operator, we have

\[
    X_0 = \frac{1}{|\mathcal{G}|} \sum_{P \in \mathcal{G}} P^T X P \in \text{relint}(f).
\]

Since \( X_0 \in A_G \), we have \( Q^T X_0 Q \in S_x (= Q^T (\mathcal{F}_X \cap A_G) Q) \) and it has rank \( r \), where \( Q \) is the orthogonal matrix given above in (2.10).

Conversely, if \( \hat{B}^*(x) \in S_x \) with rank \( r \), then \( X := Q \hat{B}^*(x) Q^T \) is in \( \mathcal{F}_X \) with rank \( r \). \( \square \)

Note that in the proof of Theorem 3.1 we exploit the following known properties of the Reynolds operator: \( \text{rank}(\mathcal{R}_G(X)) \geq \text{rank}(X) \), which is valid for all \( X \) that are positive semidefinite, and \( \mathcal{R}_G(F_X) = F_X \cap A_G \).

**Corollary 3.2.** The program (2.1) is strictly feasible if, and only if, its symmetry reduced program (2.14) is strictly feasible.

**Remark 3.3.** From the proof of Theorem 3.1, if there is a linear transformation \( X = \mathcal{L}(x) \) with a full rank feasible \( \hat{X} \in \text{range}(\mathcal{L}), \hat{X} = \mathcal{L}(\hat{x}) \), then in general we can conclude that the substitution \( X = \mathcal{L}(x) \) results in a smaller SDP with strict feasibility holding at \( \hat{x} \), i.e.,

\[
    \hat{X} > 0, \mathcal{A}(\hat{X}) = b, \hat{X} = \mathcal{L}(\hat{x}) \implies \mathcal{L}(\hat{x}) > 0, (\mathcal{A} \circ \mathcal{L})(\hat{x}) = b.
\]

### 3.2 Exposing vectors

For many given combinatorial problems, the semidefinite relaxation is not strictly feasible, i.e., it is degenerate, ill-posed, and we can apply FR [18, 46, 55]. From Section 3.1 above, we see that this implies that the symmetry reduced problem is degenerate as well. Although both SR and FR can be performed separately to obtain two independent problems, there has not been any study that implements these techniques simultaneously and efficiently for SDPs, i.e., to obtain a symmetry reduced problem that also guarantees strict feasibility. Recall that [39] combines partial FR and SR for solving SOS programs.

In what follows, we show that the exposing vectors of the symmetry reduced program (2.14) can be obtained from the exposing vectors of the original program (2.1). This enables us to facially reduce the symmetry reduced program (2.14) using the structure from the original problem.
Let $W = UU^T$, with $U \in \mathbb{R}^{n \times (n-r)}$ full column rank; and let $W$ be a nonzero exposing vector of a face of $S^d_+$ containing the feasible region $F_X$ of (2.1). Let $V \in \mathbb{R}^{n \times r}$ be such that
\[
\text{range}(V) = \text{null}(U^T).
\]
Then $FR$ means that we can use the substitution $X = V^*(R) = VRV^T$ and obtain the following equivalent, smaller, formulation of (2.1):
\[
p_{\text{SDP}}^\ast = \min \{ \langle V^TCV, R \rangle \mid \langle V^TA, V, R \rangle = b_i, \ i \in I \subseteq \{1, \ldots, m\}, \ R \in S^d_+ \}. \tag{3.1}
\]
If $V$ exposes the minimal face containing $F_X$, then strict feasibility holds. In fact, $\tilde{R}$ strictly feasible corresponds to $\tilde{X} = V^*(\tilde{R}) \in \text{relint}(F_X)$.

The following results show how to find an exposing vector that is in the commutant $A_G$.

**Lemma 3.4.** Let $W$ be an exposing vector of rank $d$ of a face $F \subseteq S^d_+, F_X \subseteq F$. Then there exists an exposing vector $W_G \in A_G$ of $F$ with $\text{rank}(W_G) \geq d$.

**Proof.** Let $W$ be the exposing vector of rank $d$, i.e., $W \succeq 0$ and
\[
X \in F_X \implies \langle W, X \rangle = 0.
\]
Since (2.1) is $G$-invariant, $PX^TP^T \in F_X$ for every $P \in G$, we conclude that
\[
\langle W, PX^TP^T \rangle = \langle P^T WP, X \rangle = 0.
\]
Therefore, $P^T WP \succeq 0$ is an exposing vector of rank $d$. Thus $W_G = \frac{1}{|G|} \sum_{P \in G} P^T WP$ is an exposing vector of $F$.

That the rank is at least $d$ follows from taking the sum of nonsingular congruences of $W \succeq 0$. \hfill \boxdot

**Lemma 3.5.** Let $W \in A_G$ be an exposing vector of a face $F \subseteq S^d_+, F_X \subseteq F$. Let $Q$ be the orthogonal matrix given above in (2.10). Then $\tilde{W} = Q^T WQ$ exposes a face of $S^d_+$ containing $S_x$.

**Proof.** Let
\[
Z = \sum_{i=1}^d x_i B_i = Q^T \left( \sum_{i=1}^d x_i B_i \right) Q \in S_x.
\]
Then, by construction $Z$ is a block-diagonal matrix, say $Z = \text{Blkdiag}(Z_1, \ldots, Z_t)$. Now, since $W$ is an exposing vector of the face of $S^d_+$ containing $F_X$ we have
\[
WX = 0, \ \forall X \in F_X \implies WX = 0, \ \forall X = \sum_i x_i B_i \succeq 0, \ \text{for some } x \text{ with } Ax = b
\]
\[
\implies \tilde{W} Z = 0, \ \forall Z \in S_x,
\]
where $\tilde{W} = Q^T WQ \succeq 0$. Thus, $\tilde{W}$ is an exposing vector of a proper face of $S^d_+$ containing $S_x$.

Since $Z = \text{Blkdiag}(Z_1, \ldots, Z_t)$ is a block-diagonal matrix and $W \in A_G$, we have that $\tilde{W} = \text{Blkdiag}(\tilde{W}_1, \ldots, \tilde{W}_t)$ with $\tilde{W}_i$ the corresponding $i$-th diagonal block of $Q^T WQ$. \hfill \boxdot
Since we may assume $W \in A_G$, the exposing vector $Q^T W Q$ is a block-diagonal matrix. Now, let us show that $Q^T W Q$ exposes the minimal face of $\mathbb{S}^n_+$ containing $S_x$, face$(S_x)$. It suffices to show that the rank of $Q^T W Q$ is $n - r$, see Theorem 3.1.

**Theorem 3.6.** Let $W \in A_G$ be an exposing vector of face$(F_X)$, the minimal face of $\mathbb{S}^n_+$ containing $F_X$. Then the block-diagonal matrix $\tilde{W} = Q^T W Q$ exposes face$(S_x)$, the minimal face of $\mathbb{S}^n_+$ containing $S_x$.

**Proof.** The minimality follows from Theorem 3.1, as rank$(\tilde{W}) = \text{rank}(W) = n - r$. \hfill \square

Now let $\tilde{W} = Q^T W Q$ expose the minimal face of $\mathbb{S}^n_+$ containing $S_x$, and let

$$\tilde{W} = \text{Blkdiag}(\tilde{W}_1, \ldots, \tilde{W}_t), \quad \tilde{W}_i = \tilde{U}_i \tilde{U}_i^T, \quad \tilde{U}_i \text{ full rank, } i = 1, \ldots, t.$$ 

Let $\tilde{V}_i$ be a full rank matrix whose columns form a basis for the orthogonal complement to the columns of $\tilde{U}_i, i = 1, \ldots, t$. Take $\tilde{V} = \text{Blkdiag}(\tilde{V}_1, \ldots, \tilde{V}_t)$. Then, the facially reduced formulation of (2.14) is

$$p_{FR}^* = \min \{ c^T x \mid Ax = b, \quad \tilde{B}^*(x) = \tilde{V} \tilde{R} \tilde{V}^T, \quad \tilde{R} \succeq 0 \}$$

$$= \min \{ c^T x \mid Ax = b, \quad \tilde{B}_k^*(x) = \tilde{V}_k \tilde{R}_k \tilde{V}_k^T, \quad \tilde{R}_k \succeq 0, \forall k = 1, \ldots, t \},$$

(3.2)

where $\tilde{V}_k \tilde{R}_k \tilde{V}_k^T$ is the corresponding $k$-th block of $\tilde{B}^*(x)$, and $\tilde{R} = \text{Blkdiag}(\tilde{R}_1, \ldots, \tilde{R}_t)$. Note that some of the blocks $\tilde{B}_k^*(x)$, and corresponding $\tilde{R}_k$, might be the same and thus can be removed in the computation, see Theorem 2.6.

**Remark 3.7.** We have assumed that an exposing vector of the minimum face of the original $SDP(2.1)$ is available. If this is not the case, then we can find a strictly feasible formulation of (2.1), and an exposing vector of the minimum face for the original problem, by using a finite number (at most $\min\{m, n - 1\}$) facial reduction steps, e.g., [18, 55, 56].

We note here that reduction techniques based on the Constrained Set Invariance Conditions, such as $*$-algebra techniques, can obtain strict feasibility by removing zero blocks after the appropriate projection, see [46].

### 3.2.1 Order of reductions

To obtain the combined symmetry and facially reduced semidefinite program (3.2), we first apply $\text{SR}$ to (2.1), and then follow this with $\text{FR}$ to the form in (2.14). A natural question is whether the order of reduction matters.

Note that the objective $\langle V^T CV, R \rangle$ and the constraints $\langle V^T A_i V, R \rangle = b_i, i \in \mathcal{I} \subseteq \{1, \ldots, m\}$, of the facially reduced program (2.3), see also (3.1), depend on the choice of $V$. We now show that the choice of this $V$ is crucial when reversing the order of reductions $\text{FR}$ and $\text{SR}$. For a naive choice of $V$, we can lose all symmetries structure for $\text{SR}$ in Section 2.2. For example, assume the data matrices $C, A_1, \ldots, A_m$ of the original problem are invariant under a non-trivial permutation group $\mathcal{G}$, i.e., they are in the commutant $A_G$, see (2.4). However the data matrices $V^T CV, V^T A_1 V, \ldots, V^T A_m V$ of the facially reduced problem may not be invariant under any non-trivial group of permutation matrices for the given $V$. Note that we can always replace $V \leftarrow VS$ using any invertible $S$. Then an arbitrary invertible congruence $S^T V^T A_i V S$ will destroy the symmetry structure in the constraint matrices.

**Lemma 3.8.** Let $V, \tilde{V}, Q$ be given as in the above paragraph, and in Theorem 3.6 and (3.2). Then

$$\text{range}(V) = \text{range}(Q \tilde{V}).$$
Proof.

\[
\text{range}(\tilde{V}) = \text{null}(Q^TWQ) \implies Q(\text{range}(\tilde{V})) = Q(\text{null}(Q^TWQ)) \\
\implies \text{range}(Q\tilde{V}) = \text{null}(Q^TWQQ^T) \\
\implies \text{range}(Q\tilde{V}) = \text{null}(W).
\]

From Lemma 3.8, we can set \( V = Q\tilde{V} \) for FR. The objective and constraints become

\[
\langle V^T CV, R \rangle = \langle \tilde{V}^T \tilde{C}\tilde{V}, R \rangle, \quad \langle V^T A_i V, R \rangle = \langle \tilde{V}^T \tilde{A}_i \tilde{V}, R \rangle = b_i, \forall i.
\]

As \( \tilde{C}, \tilde{A}_i \) and \( \tilde{V} \) are block-diagonal matrices with appropriate sizes, the data matrices \( \tilde{V}^T \tilde{C}\tilde{V} \) and \( \tilde{V}^T \tilde{A}_i \tilde{V} \) are block-diagonal as well. Since \( R \) is also a block-diagonal matrix, this choice of \( V \) implicitly exploits symmetry of the original problem data. The reduction in this case is a special case of a general reduction technique known as a projection-based method, see [46] and Remark 2.1 above.

We conclude that if FR is implemented first, then for SR to follow it is crucial to find a suitable matrix \( V \) to retain the symmetry structure in the facially reduced problem. Therefore, it is more convenient for our approach to apply symmetry reduction before facial reduction and exploit the simple relation with the exposing vectors. However, for some other symmetry reduction methods it might be more appropriate to do first FR and then SR, assuming that some symmetry is preserved after FR and that the SR and FR procedures have comparable costs, see e.g., [39].

3.3 Doubly nonnegative, DNN, program

In this section, the theory above for SDP is extended to doubly nonnegative program. We show that if a maximal exposing vector \( W \) (see Definition 3.11) of the DNN program (3.3) is given, then we can construct an exposing vector for the minimal face of the symmetry reduced DNN program (3.4). This results in a strictly feasible symmetry reduced DNN program (3.5).

Note that in addition to positive definiteness, we need \( X > 0 \), elementwise positivity, for strict feasibility to hold for the DNN relaxation. We denote the cone of nonnegative symmetric matrices of order \( n \) by \( \mathbb{N}^n \). The following extends [27, Proposition 2.3] for the intersection of faces to include exposing vectors.

**Theorem 3.9.** Let \( F_S \subseteq S^n_+ \), and let \( F_N \subseteq \mathbb{N}^n \). Let \( W_S \in S^n_+, W_N \in \mathbb{N}^n \) be exposing vectors for \( F_S, F_N \), respectively. Then

\[
W_S + W_N \text{ is an exposing vector for } F_S \cap F_N \text{ for } \text{DNN}^n.
\]

**Proof.** Note that since \( \mathbb{N}^n \) is a polyhedral cone, and both \( S^n_+, \mathbb{N}^n \) are self-dual, we get that the dual cone (nonnegative polar)

\[
(\text{DNN}^n)^* = S^n_+ + \mathbb{N}^n.
\]

Note that, by abuse of notation,

\[
\langle W_S, F_S \rangle = 0, \quad \langle W_N, F_N \rangle = 0.
\]

We can now take the sum of the exposing vectors and it is clearly an exposing vector on the intersection of the faces. \( \square \)

**Remark 3.10.** 1. *Theorem 3.9 holds for arbitrary closed convex cones.*
2. For our application, we note that the intersection $F_S \cap F_N$ is characterized by the facial representation $X \in V \mathbb{S}_+^n V^T$ and $X_{ij} = 0$ for appropriate indices $i, j$. FR on the PSD cone allows one to obtain a new SDP problem of lower dimension, since any face of the PSD cone is isomorphic to a smaller PSD cone. However, the DNN cone does not possess this property. Namely, a face of a DNN cone is not necessarily isomorphic to a smaller DNN cone. However, our splitting still allows for a simplification based on the Cartesian product of a SDP cone and a $\mathbb{N}^n$ cone, see (2.9), the paragraph after (2.9), and Remark 5.1, below.

The DNN program is defined as

$$\mathcal{P}_{\text{DNN}} = \min \{ \langle C, X \rangle | \mathcal{A}(X) = b, X \in \text{DNN}^n \}. \quad (3.3)$$

The symmetry reduced formulation of the DNN program (3.3) is

$$p_{\text{DNN}}^* = \min \{ c^T x | Ax = b, x \geq 0, \tilde{B}_k^*(x) \succeq 0, k = 1, \ldots, t \}, \quad (3.4)$$

see (2.11) for the definition of $\tilde{B}_k^*(x)$. Recall that the symmetry reduced formulation of the SDP program (2.1) is (2.14). The ambient cone of the symmetry reduced program (3.4) is the Cartesian product of cones $(\mathbb{R}_+^d, \mathbb{S}_+^n, \ldots, \mathbb{S}_+^n)$.

Let $W \in \text{DNN}^*$ be an exposing vector of (3.3). Then $W = W_S + W_N$ for some $W_S \in \mathbb{S}_+^n$ and $W_N \in \mathbb{N}^n$. The exposing vector $W \in \text{DNN}^*$ satisfies $\langle W, X \rangle = 0$ for every feasible $X$ of (3.3). Since it also holds that $\langle W_S, X \rangle \geq 0$ and $\langle W_N, X \rangle \geq 0$, we have

$$\langle W_S, X \rangle = \langle W_N, X \rangle = 0,$$

for every feasible $X$ of (3.3).

We are going to construct an exposing vector for the symmetry reduced program (3.4) by using $W$. Here the exposing vectors $(\overline{W}_n, \ldots, \overline{W}_n)$ for the semidefinite cones $(\mathbb{S}_+^n, \ldots, \mathbb{S}_+^n)$ can be derived in the same way as before. Therefore we only have to find an exposing vector for the nonnegative cone $\mathbb{R}_+^d$. Let $x$ be feasible for (3.4). Then $X = B^*(x)$ is feasible for (3.3). We have

$$\langle W_N, X \rangle = \langle W_N, B^*(x) \rangle = \langle B(W_N), x \rangle = 0.$$

Define $w := B(W_N)$. Since $W_N$ is nonnegative and $(B(W_N))_i = (B_i, W_N)$ for some zero-one matrix $B_i$, the vector $w$ is nonnegative. Then $\langle w, x \rangle = 0$ implies that $w$ is an exposing vector for the cone $\mathbb{R}_+^d$ of (3.4).

Thus facial reduction for the nonnegative cone $\mathbb{R}_+^d$ simply removes the entries $x_i$ associated to positive entries $w_i > 0$ from the program. Let $\tilde{x}$ be the vector obtained by removing these entries from $x$. Define the new data matrices $\tilde{c}, \tilde{A}$ correspondingly. The facial reduction for the semidefinite cones are handled in the same way as before. This yields the following facially reduced formulation of (3.4)

$$p_{\text{DNN}}^* = \min \{ c^T \tilde{x} | \tilde{A} \tilde{x} = b, \tilde{x} \geq 0, \tilde{B}_k^*(\tilde{x}) = \tilde{V}_k \tilde{R}_k \tilde{V}_k^T, \tilde{R}_k \succeq 0, \forall k = 1, \ldots, t \}. \quad (3.5)$$

We show below that if $W$ is a maximal exposing vector of (3.3), see Definition 3.11, then the facially and symmetry reduced program (3.5) is strictly feasible. In the sequel, we denote by $\text{supp}(M)$ the support of the matrix $M$.

**Definition 3.11.** An exposing vector $W$ of the DNN program (3.3) is maximal if it has a decomposition $W = W_S + W_N$ for some $W_S \in \mathbb{S}_+^n$ and $W_N \in \mathbb{N}^n$ satisfying:

(i) $\text{rank}(W_S)$ is maximal;
(ii) the number of positive entries in \( W_N \) is maximal, i.e., \( \text{supp}(W'_N) \subseteq \text{supp}(W_N) \) for any other exposing vector \( W_N \).

Note that if \( W_N, W'_N \in \mathbb{N}^n \) are exposing vectors for a DNN program, then \( W_N + W'_N \in \mathbb{N}^n \) is also an exposing vector. Therefore the support of \( W_N \) in the decomposition of a maximal exposing vector \( W \) is unique in Definition 3.11.

**Theorem 3.12.** Assume that \( W = W_S + W_N \), where \( W_S \in \mathbb{S}_+^n \), \( W_N \in \mathbb{N}^n \), is a maximal exposing vector of the DNN program (3.3). Then \( W_S \) and \( W_N \) are exposing vectors for the minimal face of the symmetry reduced program (3.4), or equivalently, the facially and symmetry reduced program (3.5) is strictly feasible.

**Proof.** Assume, for the sake of contradiction, that (3.5) is not strictly feasible. The existence of a feasible solution for (3.5) such that \( \tilde{R}_k \succ 0 \) for \( k = 1, \ldots, t \) can be derived in the same way as before, see Theorem 3.6. Therefore, we consider here only the case that there does not exist feasible \( \tilde{x} \) for (3.5) that is strictly positive. Then there exists an exposing vector \( w' \in \mathbb{R}^d_+ \) for (3.4) such that \( \text{supp}(w) \subsetneq \text{supp}(w') \). Let \( W'_N := B^*(w') \in \mathbb{N}^n \). Then \( \text{supp}(W_N) \subsetneq \text{supp}(W'_N) \).

Let \( X \in \text{DNN} \) be feasible for (3.3). Then \( R_G(X) = B^*(x) \in \text{DNN} \) for some \( x \) feasible for (3.4), and thus

\[
\langle W'_N, R_G(X) \rangle = \langle w', x \rangle = 0.
\]

But \( \text{supp}(X) \subseteq \text{supp}(R_G(X)) \), this means that \( \langle W_N, X \rangle = 0 \). Thus \( W_N \) is an exposing vector for (3.3) such that \( \text{supp}(W_N) \subsetneq \text{supp}(W'_N) \). This contradicts the maximality of \( W \). Thus the program (3.5) is strictly feasible. Note that the fact that we could move the nonnegativity to the reduced variable \( x \) was essential for obtaining the Slater condition.

**Remark 3.13.** One can also prove Theorem 3.12 by exploiting the following properties of \( R_G(X) \), the Reynolds operator of \( G \), see (2.5). For all feasible \( X \), it holds that

\[
\text{rank}(R_G(X)) \geq \text{rank}(X), \quad \text{supp}(R_G(X)) \supseteq \text{supp}(X).
\]

### 3.3.1 Facial reduction from the ground set for DNN

Our applications involve quadratic models of hard combinatorial problems. We now see that the view of strict feasibility and FR in [61, Theorems 3.1, 3.2] can be easily extended from SDP to DNN.

We follow the notation in [61] and define the feasible set or ground set of a quadratically constrained program as:

\[
Q := \left\{ x \in \mathbb{R}^n \mid A \left( \begin{bmatrix} 1 & x^T \\ x & xx^T \end{bmatrix} \right) = 0, \ x \geq 0 \right\},
\]

where \( A \) is a linear transformation. The relaxation, lifting, is then given by

\[
\hat{Q} := \left\{ \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \in \text{DNN}^{n+1} \mid A \left( \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \right) = 0 \right\}.
\]

Let the gangster set for \( Q \) be defined as

\[
\mathcal{G}_Q = \{(i, j) : x_i x_j = 0, \ \forall x \in Q \}
\]

with complement \( \mathcal{G}_Q^c \). Let the gangster set for \( \hat{Q} \) be defined as

\[
\mathcal{G}_{\hat{Q}} = \{(i, j) : X_{ij} = 0, \ \text{for all} \ \begin{bmatrix} 1 & x^T \\ x & X \end{bmatrix} \in \hat{Q} \}.
\]
Note that here the gangster sets are equal $\mathcal{G}_Q = \mathcal{G}_Q^\circ$, with appropriate indices. However, for a general DNN, we are not given the ground set and the gangster set is defined for the lifted problem only.

In what follows we use $e_k$ or $e$ when the meaning is clear, to denote the vector of all ones of order $k$.

**Theorem 3.14** (Slater). Suppose that $\text{conv}(Q)$ is full dimensional and that $\mathcal{G}_Q = \emptyset$. Then the Slater condition holds for $\hat{Q}$.

**Proof.** By the assumption, we can choose the finite set of vectors
\[
\left\{ v_{i,j} \in Q \mid v_{i,j} \geq 0, \text{ for each } (i,j) \in \{1,\ldots,n\} \times \{1,\ldots,n\} \right\}. \tag{3.6}
\]
As in [61], we choose an affine linear independent set $\{x_i\}_{i=1}^{n+1} \subseteq Q$, and form the matrix by adding ones and the $v_{i,j}$ defined in (3.6):
\[
V := \begin{bmatrix}
e_n & e_n^T \\
\end{bmatrix} \begin{bmatrix}
x_1,\ldots,x_{n+1} \\
v_{1,1},v_{1,2},\ldots,v_{n,n}
\end{bmatrix}
\]
We lift and get the Slater point $W := V V^T \in \hat{Q}, W > 0$.

We now extend this to obtain FR. We use our exposing vector viewpoint rather than the internal view in [61, Theorem 3.2]. We note that we can not move here the nonnegativity constraints onto $R$ as is done for our applications after SR. Moreover, though the Slater condition holds for the FR feasible set in (3.7), it is not necessarily true that the Mangasarian-Fromovitz constraint qualification holds, since some of the linear equality constraints typically become redundant after FR. We can however discard redundant equality constraints.

**Theorem 3.15** (facial reduction). Suppose that the affine hull, $\text{aff}(\text{conv}(Q)) = L$ and $\dim(L) = d$. Then there exist $A$ and $b$ with $A$ full row rank such that
\[
L = \{ x \in \mathbb{R}^n \mid Ax = b \}.
\]
Let $U = \begin{bmatrix} -b^T \\ A^T \end{bmatrix}$ and $V$ be full column rank with $\text{range}(V) = \text{null}(U)$. Then there exists a Slater point $\hat{R}$ for the FR, DNN feasible set
\[
\hat{Q}_R = \left\{ R \in S^{d+1} \mid R \succeq 0, (V R V^T) \mathcal{G}_Q^\circ \geq 0, (V R V^T)_{\mathcal{G}_Q^\circ} = 0, A(V R V^T) = 0 \right\}, \tag{3.7}
\]
where $(V R V^T)_S$ is the vector with indices chosen from the set $S$, and
\[
\hat{R} \succ 0, (V \hat{R} V^T) \mathcal{G}_Q^\circ \succ 0, (V \hat{R} V^T)_{\mathcal{G}_Q^\circ} = 0, A(V \hat{R} V^T) = 0. \tag{3.8}
\]
**Proof.** The proof is as for Theorem 3.14 after observing that $UU^T$ is an exposing vector. More precisely, from
\[
Ax - b = 0 \iff \begin{bmatrix} 1 \\ x \end{bmatrix}^T \begin{bmatrix} -b^T \\ A^T \end{bmatrix} = 0 \iff \begin{bmatrix} 1 \\ x \end{bmatrix}^T \begin{bmatrix} -b^T \\ A^T \end{bmatrix} \begin{bmatrix} -b^T \\ A^T \end{bmatrix}^T = 0, \tag{3.9}
\]
we see that $Y U U^T = 0$ for all lifted $Y$, and therefore also for the minimal face. Thus $U U^T$ is an exposing vector. The result follows after restricting the selection in (3.6) to the complement $\mathcal{G}_Q^\circ$. \qed
3.4 Singularity degree

The singularity degree defined for the semidefinite feasibility problem \( P_F \) (2.2), and denoted by \( \text{sd}(P_F) \), is the minimum number of steps with a nonzero exposing vector, for the FR algorithm to terminate with the minimal face. For \( P_{\text{SDP}} \) this means we terminate with a strictly feasible problem. Singularity degree was introduced for \( P_{\text{SDP}} \) in [58] to show that SDP feasibility problems always admit a Hölder error bound, more precisely, let \( d = \text{sd}(P_F) \), \( \mathcal{L} = \{ X | A(X) = b \} \), \( U \subset \mathbb{S}^n \) be compact. Let \( \text{dist} \) denote the norm-distance to a set. Then it is shown in [58] that there exists \( c > 0 \) such that

\[
\text{dist}(X, F_X) \leq c \left( \text{dist}^{2-d}(X, S^n_+) + \text{dist}^{2-d}(X, \mathcal{L}) \right), \forall X \in U.
\]

Remarkably, the exponent \( 2-d \) is independent of \( m, n \) and the rank of the matrices in \( F_X \). It strongly indicates the importance of FR for SDP, especially when obtaining approximate solutions with splitting type methods. This is illustrated by our numerical results, see Table 5.7 below, where lower bounds obtained by ADMM are dramatically better than those for IPM.

In this section, we show that the singularity degree of a symmetry reduced program is equal to the singularity degree of the original problem, see Theorem 3.17. Thus, we provide a heuristic indication that this error measure does not grow when applying SR. Of course, after completing FR, the singularity degree is optimal, 0.

The facial reduction algorithm applied to the semidefinite program (2.1) is described as follows. At the \( k \)-th step, FR finds an exposing vector of the feasible region of the reduced SDP of (2.1)

\[
\{ R \in S^r_k \mid A_V(R) = b_I \}, \text{ with } A_V(R) = (V^T A_i V, R)_{i \in I} \in \mathbb{R}^{|I|}, I \subseteq \{1, \ldots, m\}. \tag{3.10}
\]

Here \( V \) is a given matrix updated after each FR step. In the first step, \( V \) is the identity matrix and (3.10) is the feasible region \( F_X \) of the original problem (2.1). An exposing vector is then obtained by solving the following auxiliary system for \( y \):

\[
0 \neq A_V^*(y) \succeq 0 \text{ and } b^T y \leq 0. \tag{3.11}
\]

If \( y \) exists, then \( W = A_V^*(y) \in \mathbb{R}^{r_k \times r_k} \) is an exposing vector of the feasible region (3.10). We then do as follows:

\[
\begin{align*}
(i) & \text{ compute } V' \in \mathbb{R}^{r_k \times r_{k+1}}, \text{ full rank, range}(V') = \text{null}(W); \\
(ii) & \text{ set } V \leftarrow V'V' \in \mathbb{R}^{n \times r_{k+1}}; \\
(iii) & \text{ repeat from (3.11).} \tag{3.12}
\end{align*}
\]

At the \( k \)-th step, we have computed a vector \( y \) and a matrix \( V' \) that determines the facially reduced formulation at the next step. Choosing exposing vectors with maximum possible rank leads to the fewest iterations, see e.g., [40, 56]. For completeness, we now show that the number of iterations in the facial reduction algorithm only depends on the choice of \( y \) and not on the choice of \( V' \).

**Lemma 3.16.** The total number of facial reduction steps does not depend on the choice of \( V' \) and \( V \) in (3.12).

\(^3\)Our definition of singularity degree does not coincide with the definition from [58] when \( P_F = \{0\} \). In this case our definition gives \( \text{sd}(P_F) \geq 1 \), while Sturm defines \( \text{sd}(P_F) = 0 \). See also [40] for discussions on the definition; and see [56] for lower bound results.
Proof. Assume that $y$ satisfies the auxiliary system (3.11) for the feasible region (3.10). If we replace $V \in \mathbb{R}^{n \times r_k}$ in (3.10), with $VS \in \mathbb{R}^{n \times r_k}$, for some invertible matrix $S \in \mathbb{R}^{r_k \times r_k}$, then the same vector $y$ satisfies the new auxiliary system, as $b^T y \leq 0$ and

$$W_S := A_{V S}^*(y) = \sum_{i=1}^{m} (S^T V^T A_i V S) y_i = S^T A_{V}^*(y) S = S^T W S \succeq 0.$$ 

Since $S$ is invertible, it holds that $W_S \neq 0$ and $\text{rank}(W_S) = \text{rank}(W)$. Thus, we obtain the same reduction in the problem size at the $k$-th step.

As $\text{null}(W_S) = S^{-1} \text{null}(W)$, we have $S^{-1} V' \in \text{null}(W_S)$, where $V'$ satisfies $\text{range}(V') = \text{null}(V)$ as in (3.12). For any invertible matrix $T \in \mathbb{R}^{r(k+1) \times r(k+1)}$, we have that $V'_S = S^{-1} V'T \in \text{null}(W_S)$. Thus, in the second step of (3.12), we have

$$VS \leftarrow (VS)V'_S = VSS^{-1}V'T = (VV'T)T.$$ 

This means we can repeat our argument to show the reduction at each subsequent step is the same. $\square$

Now we describe the facial reduction algorithm applied to the symmetry reduced program (2.14). The facial reduction algorithm at the $k$-th step considers the feasible region in variables $(x, \tilde{R}_1, \ldots, \tilde{R}_t)$ determined by

$$Ax = b,$$

$$\text{blkdiag} \left( \tilde{B}^*(x) \right) = \left( \tilde{V}_1 \tilde{R}_1 \tilde{V}_1^T, \ldots, \tilde{V}_t \tilde{R}_t \tilde{V}_t^T \right),$$

for some $\tilde{V} = \text{Blkdiag}(\tilde{V}_1, \ldots, \tilde{V}_t)$ with $\tilde{V}_i \in \mathbb{R}^{n_i \times r_k}$, see also (3.2). Here $\text{blkdiag} = \text{Blkdiag}^*$. In the first step, $\tilde{V}$ is the identity matrix and we obtain the feasible region $F_x$ of the symmetry reduced program (2.14).

The auxiliary system for (3.13) is to find $(y, \tilde{W}_1, \ldots, \tilde{W}_t)$ such that

$$A^T y = \tilde{B}(\text{Blkdiag}(\tilde{W}_1, \ldots, \tilde{W}_t)),$$

$$(V_i^T \tilde{W}_1 \tilde{V}_1, \ldots, V_i^T \tilde{W}_t \tilde{V}_t) \in (S_i^{\text{ci}}, \ldots, S_i^{\text{ci}}) \text{ and } b^T y \leq 0.$$ (3.14)

Then $\text{Blkdiag}(V_i^T \tilde{W}_1 \tilde{V}_1, \ldots, V_i^T \tilde{W}_t \tilde{V}_t)$ is an exposing vector of the symmetry reduced problem. Let $\tilde{V}_i'$ be the matrix whose independent columns span $\text{null}(V_i^T \tilde{W}_i \tilde{V}_i)$. In the facial reduction algorithm, we replace the matrix $\tilde{V}_i$ by $\tilde{V}_i \tilde{V}_i'$. Then we repeat the algorithm until the auxiliary system (3.14) has no solution.

Our main result in this section is that the singularity degree of the symmetry reduced SDP (2.14) is equal to the singularity degree of the original SDP (2.1).

**Theorem 3.17.** $\text{sd}(\mathcal{P}_{F_x}) = \text{sd}(\mathcal{P}_F)$.

**Proof.** We show first the inequality $\text{sd}(\mathcal{P}_{F_x}) \leq \text{sd}(\mathcal{P}_F)$. In particular, we show that if we apply the facial reduction algorithm to the original SDP (2.1), then the solution of the auxiliary system (3.11) can be used to construct a solution to the auxiliary system (3.14) of the symmetry reduced problem (2.14).

Let $y$ be a solution to the auxiliary system (3.11) in the $k$-th facial reduction step. Let $W = A_{V}^*(y) \in A_G$ (see Lemma 3.4) and $\tilde{W} = Q^T W Q$, where $Q$ is as specified in Theorem 2.6. Further, let $\tilde{W}_j \in S_i^{\text{ci}}$ be the $j$-th block of $W (j = 1, \ldots, t)$.
If \( k = 1 \) in the \( \text{FR} \) algorithm, then the matrices \( V \) and \( \tilde{V} \) are identity matrices. As \( W \succeq 0 \), we have \( \tilde{W}_j \succeq 0 \) \((j = 1, \ldots, t)\). It also holds that \( b^T y \leq 0 \) and

\[
\mathcal{B}(\text{Blkdiag}(\tilde{W}_1, \ldots, \tilde{W}_t)) = \mathcal{B}(Q^T A^*(y)Q) = \mathcal{B}(A^*(y)) = A^T y,  
\]  
(3.15)

see (2.10) and (2.13). Thus \((y, \tilde{W}_1, \ldots, \tilde{W}_t)\) satisfies the auxiliary system (3.14). Also, we have that \( \text{rank}(A^*(y)) = \sum_{j=1}^t \text{rank}(\tilde{W}_j) \). Let \( V \) and \( \tilde{V} = \text{Blkdiag}(\tilde{V}_1, \ldots, \tilde{V}_t) \) be matrices whose independent columns span \( \text{null}(W) \) and \( \text{null}(\tilde{W}) \), respectively. It follows from Lemma 3.8 that \( \text{range}(V) = \text{range}(Q\tilde{V}) \). From Lemma 3.16 it follows that we can take \( V = Q\tilde{V} \) in the next step.

Let \( k > 1 \) and \( V = Q\tilde{V} \) where \( V \) and \( \tilde{V} \) are derived in the previous iterate of the \( \text{FR} \) algorithm. Then, we have that

\[
A^*_V(y) = V^T A^*(y) V = \tilde{V}^T (Q^T A^*(y)Q) \tilde{V} = \tilde{V}^T \tilde{W} \tilde{V}
\]

is block diagonal. As \( A^*_V(y) \succeq 0 \), we have that each block \( \tilde{V}^T \tilde{W}_j \tilde{V}_j \) \((j = 1, \ldots, t)\) is positive semidefinite. It also holds that \( b^T y \leq 0 \) and \( \mathcal{B}(\tilde{W}) = A^T y \). Thus \((y, \tilde{W}_1, \ldots, \tilde{W}_t)\) satisfies the auxiliary system (3.14). Further, we have that \( \text{rank}(A^*_V(y)) = \sum_{j=1}^t \text{rank}(\tilde{V}_j \tilde{W}_j \tilde{V}_j) \).

Let \( V' \) and \( \tilde{V}'_j \) \((j = 1, \ldots, t)\) be matrices whose independent columns span \( \text{null}(A^*_V(y)) \) and \( \text{null}(V_k^T \tilde{W}_k \tilde{V}_k) \) \((j = 1, \ldots, t)\), respectively. As \( A^*_V(y) = \tilde{V}^T \tilde{W} \tilde{V} \) is block diagonal we can simply take \( V' = V' \). Thus after updating \( V \leftarrow VV' \) and \( \tilde{V} \leftarrow \tilde{V}V' \), we have \( V = Q\tilde{V} \) in the next step. We can repeat the same argument until the facial reduction algorithm terminates.

Next, we show that \( \text{sd}(\mathcal{P}_F) \geq \text{sd}(\mathcal{P}_F) \). Let us assume that \((y, \tilde{W}_1, \ldots, \tilde{W}_t)\) satisfies the auxiliary system (3.14) in the first facial reduction step. Recall that in the first step, \( \tilde{V} \) is the identity matrix. For \( Q \) defined as in Theorem 2.6, we have that

\[
W := Q \text{Blkdiag}(\tilde{W}_1, \ldots, \tilde{W}_t)Q^T \in A_G.  
\]
(3.16)

To show that \( y \) satisfies the auxiliary system (3.11), we have to prove that \( A^*(y) \succeq 0 \). It holds that

\[
\mathcal{B}(A^*(y)) = A^T y = \mathcal{B}(\text{Blkdiag}(\tilde{W}_1, \ldots, \tilde{W}_t)) = \mathcal{B}(W),
\]
(3.17)

see also (3.15). The second equality above uses the feasibility of (3.14). Since we have that \( A^*(y) \in A_G \) and \( W \in A_G \), it follows from (3.17) that \( W = A^*(y) \), and from (3.16) and (3.14) that \( A^*(y) \succeq 0 \). Recall that we assumed that the data matrices of the \text{SDP} problem (2.1) are contained in the matrix \(*\)-algebra \( A_G \), see Section 2.2.

Let \( k > 1 \). Using \( V = Q\tilde{V} \) where \( V \) and \( \tilde{V} \) are derived in the previous iterate of the \( \text{FR} \) algorithm, (3.14) and (3.16), we have that

\[
0 \leq \tilde{V}^T \text{Blkdiag}(\tilde{W}_1, \ldots, \tilde{W}_t) \tilde{V} = V^T W V = V^T A^*(y) V = A^*_V(y).
\]

In addition, it follows from construction of \( W \) and Lemma 3.16 that we can take \( V \) and \( \tilde{V} \) such that \( V = Q\tilde{V} \) in the next \( \text{FR} \) step.

The following Corollary 3.18 follows from [61, Theorem 3.2] in that the linear manifold is represented by a concrete constraint and is applied to finding an exposing vector. More precisely, if we can find the affine span of our original feasible set in the ground space, then we can always find the representation using a linear mapping as in (3.9). This means we can always find the appropriate exposing vector and obtain singularity degree one, see also [18]. Note that this includes the hard combinatorial problems we work with below.
Corollary 3.18. Consider the quadratic model as given in Theorem 3.15, and suppose that the matrix $A$ is part of the given data of the problem, then the singularity degree is one.

Proof. The proof uses $A$ to construct the exposing vector. Therefore, one step of the FR algorithm is sufficient, see (3.9). More precisely, the linear constraint in the ground set is lifted into the SDP as in (3.9).

Remark 3.19. The definition of singularity degree can be extended to DNN, and to a general cone optimization problem, to be the minimum number of steps in the FR [7, Algor. B]. Here this means we continue to find the minimum number of steps with nonzero exposing vectors. An interesting question is to find the relationship between the singularity degree of the SDP and the DNN. It appears that the DNN is at most one more than for SDP. Moreover it is known that the singularity degree of the DNN is at most $n$, see [40, Corollary 20].

3.5 Simplifications

After FR, some of the constraints become redundant in the facially reduced program (3.1). We show here that the same constraints are also redundant in the facially and symmetry reduced program (3.2). Proof of Lemma 3.20 is clear.

Lemma 3.20. For any subset $I \subseteq [m] := \{1, \ldots, m\}$, we define the spectrahedron

$$F(I) := \{ X \in S^n \mid \langle A_i, X \rangle = b_i \ \forall i \in I, \ X = VRV^T, \ R \in S^r_+ \}. $$

If the constraints in $[m] \setminus I$ are redundant, e.g., $F([m]) = F(I)$, then $F([m]) \cap A_G = F(I) \cap A_G$.

Although a proof of Corollary 3.21 follows directly from Lemma 3.20, we provide it due to our intricate notation.

Corollary 3.21. Let $I \subset \{1, \ldots, m\}$. Suppose that the constraints $\langle A_k, VRV^T \rangle = b_k, k \not\in I$, are redundant in (3.1), i.e., the facially reduced formulation (3.1) is equivalent to

$$\min_{R \in S^r_+} \{ \langle VT CV, R \rangle \mid \langle V^T A_i V, R \rangle = b_i, \ \forall i \in I \}. \quad (3.18)$$

Then the constraints

$$\sum_{j=1}^d A_{k,j} x_j = b_k, k \not\in I,$$

are redundant in (3.2), i.e., the facially and symmetry reduced program (3.2) is equivalent to

$$\min_{x \in \mathbb{R}^d, R \in S^r_+} \{ c^T x \mid \sum_{j=1}^d A_{i,j} x_j = b_i, \forall i \in I, \ \tilde{B}^*(x) = \tilde{V} \tilde{R} \tilde{V}^T \}. \quad (3.19)$$

Proof. Let $Q$ be specified in Theorem 2.6. Since $\text{range}(V) = \text{range}(Q \tilde{V})$, see Lemma 3.8, we assume w.l.o.g. that $V$ in (3.18) satisfies $V = Q \tilde{V}$. Let $(x, \tilde{R})$ be feasible for (3.19). Define $X := \sum_{i=1}^d B_i x_i$. We show the equivalence in the following order:

$$(x, \tilde{R}) \text{ feasible for (3.19)} \implies \tilde{R} \text{ feasible for (3.18)} \implies \tilde{R} \text{ feasible for (3.1)} \implies (x, \tilde{R}) \text{ feasible for (3.2)}$$
Since \( Q^T(\sum_{i=1}^{d} B_i x_i)Q = \tilde{V} \tilde{R} \tilde{V}^T = Q^T \tilde{V} \tilde{R} \tilde{V}^T Q \), we have \( \sum_{i=1}^{d} B_i x_i = \tilde{V} \tilde{R} \tilde{V}^T \). Using the feasibility and (2.13), it holds that

\[
\langle A_i, \tilde{V} \tilde{R} \tilde{V}^T \rangle = \langle A_i, \sum_{j=1}^{d} B_j x_j \rangle = \sum_{j=1}^{d} A_{i,j} x_j = b_i, \forall i \in \mathcal{I}.
\]

Thus all the linear constraints in (3.18) are satisfied, and \( \tilde{R} \succeq 0 \) is feasible for (3.18). By assumption, \( \tilde{R} \) is also feasible for (3.1). Thus the constraints \( \langle A_i, \tilde{V} \tilde{R} \tilde{V}^T \rangle = b_i, \forall i \notin \mathcal{I} \) are satisfied as well. This shows that \( (x, \tilde{R}) \) is feasible for (3.2).

To obtain a formulation for the facially and symmetry reduced relaxation (3.2) in variable \( \tilde{R} \) only, we can replace \( x \) in terms of \( \tilde{R} \) using the constraint \( \tilde{B}^*(x) = \tilde{V} \tilde{R} \tilde{V}^T \). This substitution can be done easily by rewriting the constraints as

\[
b_i = \langle A_i, X \rangle = \langle Q^T A_i Q, Q^T X Q \rangle = \langle Q^T A_i Q, \tilde{B}^*(x) \rangle = \langle Q^T A_i Q, \tilde{V} \tilde{R} \tilde{V}^T \rangle.
\]

The objective can be similarly changed. This method, however, does not work for DNN relaxations. This difficulty can be resolved as follows.

**Theorem 3.22.** Consider the facially and symmetry reduced relaxation (3.2) with nonnegativity constraints,

\[
\min \{ c^T x \mid Ax = b, \quad \tilde{B}^*(x) = \tilde{V} \tilde{R} \tilde{V}^T, \quad \tilde{R} \succeq 0, \quad x \geq 0 \},
\]

(3.20)

where \( \tilde{R} = \text{Blkdiag}(\tilde{R}_1, \ldots, \tilde{R}_t) \). Equate \( x \) with

\[
x \leftarrow f(\tilde{R}) = \text{Diag}(w)^{-1} \mathcal{B}(\tilde{V} \tilde{R} \tilde{V}^T),
\]

where

\[
w = (\langle B_i, B_i \rangle)_1, \in \mathbb{R}^d, \quad V = Q \tilde{V},
\]

and \( Q \) is specified in Theorem 2.6. Then (3.20) is equivalent to

\[
\min \{ c^T f(\tilde{R}) \mid Af(\tilde{R}) = b, \quad \tilde{R} \succeq 0, \quad f(\tilde{R}) \geq 0 \},
\]

(3.21)

where diagonal blocks of the block diagonal matrix \( \tilde{R} \) are set to be equal for the corresponding repeating blocks in \( Q^T A_G Q \).

**Proof.** If \( (x, \tilde{R}) \) is feasible for (3.20), then \( \mathcal{B}^*(x) = \tilde{V} \tilde{R} \tilde{V}^T \). As \( w > 0 \) and \( \mathcal{B}\mathcal{B}^* = \text{Diag}(w) \), we have \( x = f(\tilde{R}) \) and thus \( \tilde{R} \) is feasible for (3.21).

Let \( \tilde{R} \) be feasible for (3.21). Since \( \tilde{V} \tilde{R} \tilde{V}^T \) is a block-diagonal matrix in the algebra \( Q^T A_G Q \), we have \( \tilde{V} \tilde{R} \tilde{V}^T = Q \tilde{V} \tilde{R} \tilde{V}^T Q^T \in A_G \). It follows from Theorem 2.6 that there exists a unique \( x \) such that \( \tilde{V} \tilde{R} \tilde{V}^T = \mathcal{B}^*(x) \). Then we must have \( x = f(\tilde{R}) \) and thus \( (x, \tilde{R}) \) is feasible for (3.20).

For the Hamming scheme, we have an explicit expression for the orthogonal matrix \( Q \) used in Theorem 3.22, see Example 2.5 and Section 5.1. In general, we do not know the corresponding orthogonal matrix explicitly. In Section 5.2, we use the heuristics from [15] to compute a block diagonalization of \( A_G \). In this case, the equivalence in Theorem 3.22 may not be true, and (3.21) may be weaker than (3.20). However our computational results indicate that all the bounds remain the same, see Table 5.7 below.
4 The alternating direction method of multipliers, ADMM

It is well known that interior-point methods do not scale well for SDP. Moreover, they have great difficulty with handling additional cutting planes such as nonnegativity constraints. In particular, solving the doubly nonnegative relaxation, DNN, using interior-point methods is extremely difficult. The alternating direction method of multipliers is a first-order method for convex problems developed in the 1970s, and rediscovered recently. This method decomposes an optimization problem into subproblems that may be easier to solve. In particular, it is extremely successful for splittings with two cones. This feature makes the ADMM well suited for our large-scaled DNN problems. For state of the art in theory and applications of the ADMM, we refer the interested readers to [8].

Oliveira, Wolkowicz and Xu [44] propose a version of the ADMM for solving an SDP relaxation for the Quadratic Assignment Problem (QAP). Their computational experiments show that the proposed variant of the ADMM exhibits remarkable robustness, efficiency, and even provides improved bounds.

4.1 Augmented Lagrangian

We modify the approach from [44] for solving our SR and FR reduced DNN relaxation (3.2). We have a greatly simplified structure as we applied SR to the SDP relaxation, and we were then able to move the nonnegativity constraints to a simple vector \(x \geq 0\) contraint. We in particular obtain a more efficient approach for solving the \(x\)-subproblem.

Let \(\tilde{V} = \text{Blkdiag}(\tilde{V}_1, \ldots, \tilde{V}_t)\) and \(\tilde{R} = \text{Blkdiag}(\tilde{R}_1, \ldots, \tilde{R}_t)\). The augmented Lagrangian of (3.2) corresponding to the linear constraints \(\tilde{B}^*(x) = \tilde{V} \tilde{R} \tilde{V}^T\) is given by:

\[
L(x, \tilde{R}, \tilde{Z}) = \langle \tilde{C}, \tilde{B}^*(x) \rangle + \langle \tilde{Z}, \tilde{B}^*(x) - \tilde{V} \tilde{R} \tilde{V}^T \rangle + \frac{\beta}{2} || \tilde{B}^*(x) - \tilde{V} \tilde{R} \tilde{V}^T ||^2,
\]

where, see (2.12), \(\tilde{C} = Q^T C Q\) is a block-diagonal matrix as \(C \in A_G\), \(\tilde{Z}\) is also in block-diagonal form, and \(\beta > 0\) is the penalty parameter.

The alternating direction method of multipliers, ADMM, uses the augmented Lagrangian, \(L(x, \tilde{R}, \tilde{Z})\), and essentially solves the max-min problem

\[
\max_{\tilde{Z}} \min_{x \in P, \tilde{R} \geq 0} L(x, \tilde{R}, \tilde{Z}),
\]

where \(P\) is a simple polyhedral set of constraints on \(x\), e.g., linear constraints \(Ax = b\) and nonnegativity constraints, see (4.3) below. The advantage of the method is the simplifications obtained for the constraints by taking advantage of the splitting in the variables. We then find the following updates \((x_+, \tilde{R}_+, \tilde{Z}_+)\):

\[
x_+ = \arg \min_{x \in P} L(x, \tilde{R}, \tilde{Z}), \\
\tilde{R}_+ = \arg \min_{\tilde{R} \geq 0} L(x_+, \tilde{R}, \tilde{Z}), \\
\tilde{Z}_+ = \tilde{Z} + \gamma \beta (\tilde{B}^*(x_+) - \tilde{V} \tilde{R}_+ \tilde{V}^T).
\]

Here, \(\gamma \in (0, \frac{1+\sqrt{5}}{2})\) is the step size for updating the dual variable \(\tilde{Z}\). In the following sections we explain in details how to solve each subproblem.
4.2 On solving the $\tilde{R}$-subproblem

The $\tilde{R}$-subproblem can be explicitly solved. We complete the square and get the equivalent problem

$$
\tilde{R}_+ = \min_{\tilde{R} \succeq 0} \| \tilde{C}^* - \tilde{V} \tilde{R} \tilde{V}^T + \frac{1}{\beta} \tilde{Z} \|^2
$$

$$
= \min_{\tilde{R} \succeq 0} \| \tilde{R} - \tilde{V}^T (\tilde{C}^* + \frac{1}{\beta} \tilde{Z}) \tilde{V} \|^2
$$

$$
= \sum_{k=1}^t \min_{\tilde{R}_k \succeq 0} \| \tilde{R}_k - (\tilde{V}^T (\tilde{C}^* + \frac{1}{\beta} \tilde{Z}) \tilde{V})_k \|^2. (4.1)
$$

Here, we normalize each block $\tilde{V}_k$ such that $\tilde{V}_k^T \tilde{V}_k = I$, and thus $(\tilde{V}^T (\tilde{C}^* + \frac{1}{\beta} \tilde{Z}) \tilde{V})_k$ is the $k$-th block of $\tilde{V}^T (\tilde{C}^* + \frac{1}{\beta} \tilde{Z}) \tilde{V}$ corresponding to $\tilde{R}_k$. So we only need to solve $k$ small problems whose optimal solutions are

$$
\tilde{R}_k = \mathcal{P}_{\mathbb{S}_+} \left( \tilde{V}^T (\tilde{C}^* + \frac{1}{\beta} \tilde{Z}) \tilde{V} \right)_k, \quad k = 1, \ldots, t,
$$

where $\mathcal{P}_{\mathbb{S}_+}(M)$ is the projection onto the cone of positive semidefinite matrices.

4.3 On solving the $x$-subproblem

For the $x$-subproblem, we have

$$
x_+ = \arg \min_{x \in P} \left\| \tilde{B}^*(x) - \tilde{V} \tilde{R} \tilde{V}^T + \frac{\tilde{C} + \tilde{Z}}{\beta} \right\|^2.
$$

(4.2)

For many combinatorial optimization problems, some of the constraints $Ax = b$ in (2.14) become redundant after FR of their semidefinite programming relaxations, see Corollary 3.21. Thus, the set $P$ often collapses to a simple set. This often leads to an analytic solution for the $x$-subproblem; e.g., this happens for the quadratic assignment, graph partitioning, vertex separator, and shortest path problems.

For some interesting applications, the $x$-subproblem is equivalent to the following special case of the weighted, relaxed, quadratic knapsack problem:

$$
\min_x \frac{1}{2} \| \mathcal{T}^*(x) - Y \|^2
$$

s.t. $x \in P := \{ x \mid w^T x = c, x \geq 0 \},
$$

(4.3)

where $Y$ is a given matrix and $\mathcal{T}^*(x) = \sum_{i=1}^q x_i T_i$ for some given symmetric matrices $T_i$. The problem (4.3) is a projection onto the weighted simplex. We consider the following assumption on a linear transformation $\mathcal{T} : \mathbb{S}^n \to \mathbb{R}^q$ and its adjoint.

**Assumption 4.1.** The linear transformation $\mathcal{T} : \mathbb{S}^n \to \mathbb{R}^q$ in (4.3) satisfies

$$
\mathcal{T}(\mathcal{T}^*(x)) = \text{Diag}(w)x, \forall x \in \mathbb{R}^q, \text{ for some } w > 0.
$$

**Lemma 4.2.** Suppose that the linear transformation $\mathcal{T}$ satisfies Assumption 4.1, and that (4.3) is feasible. Then the projection problem (4.3) can be solved efficiently (explicitly) using Algorithm 4.3.

**Proof.** The Lagrangian function of the problem is

$$
\frac{1}{2} \| \mathcal{T}^*(x) - Y \|^2 - \tau(w^T x - c) - \lambda^T x,
$$

$$
\lambda^T
$$
where $\tau \in \mathbb{R}$ and $\lambda \in \mathbb{R}^q_+$ are the Lagrangian multipliers. The KKT optimality conditions for the problem are given by

$$
\mathcal{T}(\mathcal{T}^* (x)) - \mathcal{T}(Y) - \tau w - \lambda = 0,
$$

$$
x \geq 0,
$$

$$
\lambda \geq 0,
$$

$$
\lambda^T x = 0,
$$

$$
w^T x = c.
$$

Note that $\text{Diag}(w)$ is the matrix representation of $\mathcal{T} \circ \mathcal{T}^*$. This means that $(w_i^{-1} y_i)_{i=1}^q$, and we can simplify the first condition. \(^4\) This yields

$$
x_i = w_i^{-1} (\mathcal{T}(Y))_i + \tau + w_i^{-1} \lambda_i.
$$

Define the data vector $y := \mathcal{T}(Y)$. The complementary slackness $\lambda^T x = 0$ implies that if $x_i > 0$, then $\lambda_i = 0$ and $x_i = w_i^{-1} y_i + \tau$. If $x_i = 0$, then $w_i^{-1} y_i + \tau = -w_i^{-1} \lambda_i \leq 0$. Thus the zero and positive entries of the optimal solution $x$ correspond to the smaller than $-\tau$ and the larger than $-\tau$ entries of $(w_i^{-1} y_i)_{i=1}^q$, respectively.

Let us assume, without loss of generality, that $(w_i^{-1} y_i)_{i=1}^q, x$ are sorted in non-increasing order:

$$
y_1/w_1 \geq \ldots \geq y_k/w_k \geq \ldots \geq y_q/w_q, \quad x_1 \geq \ldots \geq x_k > x_{k+1} = \ldots = x_q = 0.
$$

The condition $w^T x = c$ implies that

$$
w^T x = \sum_{i=1}^k w_i (y_i/w_i + \tau) = \sum_{i=1}^k y_i + \tau \sum_{i=1}^k w_i = c,
$$

and thus

$$
\tau = \frac{c - \sum_{i=1}^k y_i}{\sum_{i=1}^k w_i}.
$$

Therefore, one can solve the problem by simple inspection once $k$ is known. The following algorithm finds an optimal solution $x$ to the problem (4.3). The correctness of the algorithm is then similar to the projection onto the (unweighted) simplex problem, see \([10,11]\).  \(\square\)

---

**Algorithm 4.3** (Finding an optimal solution for (4.3)).

**Input:** $w \in \mathbb{R}^q, y \in \mathbb{R}^q$

Sort $\{y_i/w_i\}$ such that $y_1/w_1 \geq \ldots \geq y_q/w_q$

Set $k := \max_{1 \leq k \leq n} \{k \mid w^{-1} y_k + (\sum_{i=1}^k w_i)^{-1} (c - \sum_{i=1}^k y_i) > 0\}$

Set $\tau := (\sum_{i=1}^k w_i)^{-1} (c - \sum_{i=1}^k y_i)$

Set $x_i = \max\{w_i^{-1} y_i + \tau, 0\}$ for $i = 1, \ldots q$

**Output:** $x \in \mathbb{R}^q$

---

In our examples, see Sections 5.1 and 5.2, the $x$-subproblem (4.2) satisfies Assumption 4.1. Moreover, we have the following lemma. We remind the reader that $J$ denotes the matrix of all ones.

\(^4\)Note that this is always satisfied for basis matrices from a coherent configuration.
Lemma 4.4. The $x$-subproblem (4.2) satisfies Assumption 4.1, if

$$P = \{ x \in \mathbb{R}^q \mid \langle J, B^*(x) \rangle = c, x \geq 0 \}.$$  

Proof. It holds that

$$(\tilde{B}(\tilde{B}^*(x)))_i = \langle \tilde{B}_i, \sum_{j=1}^{q} \tilde{B}_j x_j \rangle = \langle \tilde{B}_i, \tilde{B}_i x_i \rangle = \text{trace}(Q^TB_i^TQ^TB_i) x_i = w_i x_i, \quad (4.4)$$

where $w_i = \text{trace}(B_i^TB_i)$. Furthermore, $\langle J, B^*(x) \rangle = w^T x$ with $w = (w_i) \in \mathbb{R}^q$. Thus we set $\mathcal{T} = B$ and note that $\mathcal{T}(\mathcal{T}^*(x)) = \text{Diag}(w^T)x$. \hfill $\square$

5 Numerical results

We now demonstrate the efficiency of our new approach on two classes of problems: the quadratic assignment problem, QAP, and several types of graph partitioning problem, GP.

Our tests were on: Dell PowerEdge M630 computer; two Intel Xeon E5-2637v3 4-core 3.5 GHz (Haswell) CPU; 64GB memory; linux. The interior point solver was Mosek, see [1]. We had to use a different computer to accommodate some of the larger problems when using an interior point approach, see description of Table 5.4.

We report only the ADMM solver time, since the preprocessing time is very small in comparison with the ADMM part. In particular, we find exposing vectors explicitly in all our examples. Further, for most of the test instances we know generators of automorphism groups as well as the orthogonal matrix $Q$. For instances for which we need to find generators e.g., the instances in Table 5.3 the preprocessing time is less than one second.

The stopping conditions and tolerances are outlined at the start of Section 5.1.3, in Definition 5.10. Our results include huge problems of sizes up to $n = 512$ for the QAP, yielding of the order $n^2$ SDP constraints and $n^4$ nonnegativity constraints.5

5.1 The quadratic assignment problem, QAP

5.1.1 Background for the QAP

The Quadratic Assignment Problem was introduced in 1957 by Koopmans and Beckmann as a model for location problems that take into account the linear cost of placing a new facility on a certain site, plus the quadratic cost arising from the product of the flow between facilities and distances between sites. The QAP contains the traveling salesman problem as a special case and is therefore NP-hard in the strong sense. It is generally considered to be one of the hardest of the NP-hard problems.

Let $A, B \in \mathbb{S}^n$, and let $\Pi_n$ be the set of $n \times n$ permutation matrices. The QAP (with the linear term with appropriate $C$ in brackets) can be stated as follows:

$$\min_{X \in \Pi_n} \text{trace}(AX^TBX) \quad (+ \text{trace}(X^TC)).$$

The QAP is extremely difficult to solve to optimality, e.g., problems with $n \geq 30$ are still considered hard. It is well known that SDP relaxations provide strong bounds, see e.g., [15, 69]. However even for sizes $n \geq 15$, it is difficult to solve the resulting SDP relaxation by interior point

5The link to the codes for the QAP can be found on the webpage www.huhao.org. The codes for the other problems require finding symmetries in the graph; and therefore these codes and details can be obtained upon request directly from the authors.
methods if one cannot exploit special structure such as symmetry. Solving the DNN relaxation is significantly more difficult.

Here, we first consider the DNN relaxation for the QAP from Povh and Rendl [50], i.e.,

$$\begin{align*}
\min & \quad \text{trace}(A \otimes B)Y \\
\text{s.t.} & \quad (J_{n^2}, Y) = n^2 \\
& \quad (I_n \otimes (J_n - I_n)) + (J_n - I_n) \otimes I_n, Y) = 0 \\
& \quad (I_n \otimes E_{ii}, Y) = 1, \forall i = 1, \ldots, n \\
& \quad (E_{ii} \otimes I_n, Y) = 1, \forall i = 1, \ldots, n \\
& \quad Y \succeq 0, Y \succeq 0, \quad (Y \in \text{DNN})
\end{align*}$$

(5.1)

where and $E_{ii} = u_i u_i^T$, where $u_i \in \mathbb{R}^n$ is $i$-th unit vector. The authors in [15, Theorem 7.1] show that one can take

$$A_G = A_{\text{aut}(A)} \otimes A_{\text{aut}(B)},$$

(5.2)

where $\text{aut}(A) := \{ P \in \Pi_n : AP = PA \}$ is the automorphism group of $A$.

**Remark 5.1.** The DNN relaxation (5.1) is known to be theoretically equivalent, yielding the same optimal value, to the DNN relaxation denoted (QAP$_{R3}$) in Zhao et al. [69]. The constraints $\langle I_n \otimes (J_n - I_n) + (J_n - I_n) \otimes I_n, Y \rangle = 0$ are generally called the gangster constraints, see Lemma 5.4. The third and fourth lines of constraints in (5.1) arise from the row and column sum constraints.

Recall that $\text{svec}$ is the linear transformation that vectorizes symmetric matrices, [69]. We define $\text{gsvec}$ to do this vectorization of symmetric matrices while ignoring the elements set to zero by the gangster constraints. Then we can eliminate the gangster constraints completely and replace the DNN constraints to get the equivalent problem to (5.1):

$$\begin{align*}
\min & \quad \text{gsvec}(A \otimes B)^T y \\
\text{s.t.} & \quad \text{gsvec}(J_{n^2})^T y = n^2 \\
& \quad \text{gsvec}(I_n \otimes E_{ii})^T y = 1, \forall i = 1, \ldots, n \\
& \quad \text{gsvec}(E_{ii} \otimes I_n)^T y = 1, \forall i = 1, \ldots, n \\
& \quad \text{gsvec}^*(y) \succeq 0, y \succeq 0.
\end{align*}$$

(5.3)

This form is now similar to our final SR reduced form before FR, see (2.14); and this emphasizes that the DNN can be represented in a split form.

In the following lemma we derive the null space of the feasible solutions of (5.1), see also Corollary 2.2.7 in [60].

**Lemma 5.2.** Let $U := \frac{1}{\sqrt{n}}(nI - J) \in \mathbb{R}^{n \times n}$, and let $Y$ be in the relative interior of the feasible set of (5.1). Then

$$\text{null}(Y) = \text{range} \left( \left[ U \otimes e_n \quad e_n \otimes U \right] \right).$$

**Proof.** Let $X \in \Pi_n$. Then $Xe_n = X^Te_n = e_n$, and thus

$$\begin{align*}
(U \otimes e_n)^T \text{vec}(X) &= U^T e_n = 0, \\
(e_n \otimes U)^T \text{vec}(X) &= U^T e_n = 0.
\end{align*}$$

Thus

$$\begin{align*}
\hat{Y} = \frac{1}{n!} \sum_{X \in \Pi_n} \text{vec}(X) \text{vec}(X)^T = \frac{1}{n^2} (J \otimes J) + \frac{1}{n^2(n-1)} (nI - J) \otimes (nI - J).
\end{align*}$$

(5.4)

26
It is proven in [60] that $\hat{Y}$ is in the relative interior of the feasible set of (5.1). Recall that every matrix $Y$ in the relative interior of a face has the same null space. This shows that $\text{null}(Y) \supseteq \text{range} \left( [U \otimes e_n \ e_n \otimes U] \right)$.

It remains to show that

$$\text{dim} \left( \text{range} \left( [U \otimes e_n \ e_n \otimes U] \right) \right) = 2(n-1).$$

To see this, we choose the square submatrix of size $2n-1$ associated to rows: $\{kn \mid k = 1, \ldots, n-1\} \cup \{n(n-1)+1, \ldots, n^2-1\}$; cols: $\{1, \ldots, n-1\} \cup \{n+1, \ldots, 2n\}$.

It has the form

$$\frac{1}{\sqrt{n}} (nI - J) \in S^{2(n-1)}.$$ 

This square submatrix clearly has rank $2(n-1)$, and thus the statement follows.

Let us now derive an exposing vector of the SDP relaxation (5.1) ignoring the nonnegativity, as we have shown we can add the nonnegativity on after the reductions.

**Lemma 5.3.** Consider (5.1) without the nonnegativity constraints. Then

$$W = I_n \otimes nJ_n + J_n \otimes (nI_n - 2J_n) \in A_G \subseteq S_{n^2}^{2},$$

and is an exposing vector of rank $2(n-1)$ in $A_G$.

**Proof.** Let $U$ be defined as in Lemma 5.2. Using the properties of the Kronecker product, we have

$$0 \preceq W = \left[ U \otimes e_n \ e_n \otimes U \right] \left[ U \otimes e_n \ e_n \otimes U \right]^T = (UU^T) \otimes J + J \otimes (UU^T) = (nI - J) \otimes J + J \otimes (nI - J) = I \otimes nJ + J \otimes (nI - 2J),$$

as $UU^T = nI - J$. From Lemma 5.2, we have $W$ is an exposing vector of rank $2(n-1)$. Let $P$ be any permutation matrix of order $n$. Then $P^T(UU^T)P = UU^T$ by construction. We now have $(P_1 \otimes P_2)^T W (P_1 \otimes P_2) = W$, for any $P_1, P_2 \in \Pi_n$; and thus $W \in A_G$. 

In the rest of this section we show how to do FR for the symmetry reduced program of (5.1). We continue to add on nonnegativity constraints to SDP relaxations as discussed above. The facially reduced formulation of (5.1) is also presented in [60]. We state it here for later use.

**Lemma 5.4 ([60]).** The facially reduced program of the doubly nonnegative, DNN (5.1) is given by

$$\min \ \langle (V^T (A \otimes B) V), R \rangle \\ \text{s.t.} \ \langle V^T J V, R \rangle = n^2 \\ G(V R V^T) = 0 \\ V R V^T \succeq 0 \\ R \in S_{n^2}^{(n-1)^2+1},$$

where, by abuse of notation, $G : S^{n^2} \to S^{n^2}$ is a linear operator defined by $G(Y) := (J - (I \otimes (J-I) + (J-I) \otimes I)) \circ Y$ \footnote{We use $G$ as the group and as a linear operator, usually referred to as the gangster operator, since the meaning is clear from the context. Here $\circ$ denotes the Hadamard product.}, and the columns of $V \in \mathbb{R}^{n^2 \times (n-1)^2+1}$ form a basis of the null space of $W$, see Lemma 5.3.
Note that the constraints \( \langle I \otimes E_{ii}, Y \rangle = 1 \) and \( \langle E_{ii} \otimes I, Y \rangle = 1 \) have become redundant after FR in (5.6).

We now discuss the symmetry reduced program. The symmetry reduced formulation of (5.1) is studied in [16]. We assume that the automorphism group of the matrix \( A \) is non-trivial. To simplify the presentation, we assume

\[
A = \sum_{i=0}^{d} a_i A_i,
\]

where \( \{A_0, \ldots, A_d\} \) is the basis of the commutant of the automorphism group of \( A \). For instance the matrices \( A_i \) \((i = 0, 1, \ldots, d)\) may form a basis of the Bose-Mesner algebra of the Hamming scheme, see Example 2.5. Further, we assume from now on that \( A_0 \) is a diagonal matrix, which is the case for the Bose-Mesner algebra of the Hamming scheme. Here, we do not assume any structure in \( B \). However the theory applies also when \( B \) has some symmetry structure and/or \( A_0 \) is not diagonal; see our numerical tests for the minimum cut problem in Section 5.2, below.

If the SDP (5.1) has an optimal solution \( Y \in S_{+}^{n^2} \), then it has an optimal solution of the form

\[
Y = \sum_{i=0}^{d} A_i \otimes Y_i
\]

for some matrix variables \( Y_0, \ldots, Y_d \in \mathbb{R}^{n \times n} \), see (5.2) and Section 2.2. We write these matrix variables in a more compact way as \( y = (\text{vec}(Y_0), \ldots, \text{vec}(Y_d)) \), if necessary.

Denote by \( \tilde{B}_k(y) \in S_{++}^{n_k} \) the \( k \)-th block of the block-diagonal matrix

\[
\tilde{B}_k(y) := (Q \otimes I)^T Y (Q \otimes I) = \sum_{i=0}^{d} (Q^T A_i Q) \otimes Y_i,
\]

where \( Q \) is the orthogonal matrix block-diagonalizing \( A_i \) \((i = 0, \ldots, d)\).

**Lemma 5.5.** The symmetry reduced program of the DNN relaxation (5.1) is given by

\[
\begin{align*}
\min \quad & \sum_{i=0}^{d} a_i \text{trace}(A_i A_i) \text{trace}(BY_i) \\
\text{s.t.} \quad & \sum_{i=0}^{d} \text{trace}(J A_i) \text{trace}(J Y_i) = n^2 \\
& \text{offDiag}(Y_0) = 0 \\
& \text{diag}(Y_i) = 0, \quad i = 1, \ldots, d \\
& \text{diag}(Y_0) = \frac{1}{n} e_n \\
& Y_j \geq 0, \quad j = 0, \ldots, d \\
& \tilde{B}_k(y) \in S_{++}^{n_k}, \quad k = 1, \ldots, t,
\end{align*}
\]

where \( \tilde{B}_k(y) \) is the \( k \)-th block from (5.7), and \( \text{offDiag}(Y_0) = 0 \) is the linear constraints that the off-diagonal elements are zero.

**Proof.** See e.g., [15,60]. \( \square \)

It remains to facially reduce the symmetry reduced program (5.8). Note that \( W \in A_G \) can be written as \( W = \sum_{i=0}^{d} A_i \otimes W_i \), for some matrices \( W_0, \ldots, W_d \in \mathbb{R}^{n \times n} \). Theorem 3.6 shows that the block-diagonal matrix

\[
\tilde{W} := (Q \otimes I)^T W (Q \otimes I) = \sum_{i=0}^{d} (Q^T A_i Q) \otimes W_i
\]

is an exposing vector of the symmetry reduced program (5.8). Further, we denote by \( \tilde{W}_k \) \((k = 1, \ldots, t)\) the \( k \)-th block of \( \tilde{W} \). Let us illustrate this with Example 5.6.
Example 5.6. Consider Example 2.5, where \( A_i \) \((i = 0, \ldots, d)\) form a basis of the Bose-Mesner algebra of the Hamming scheme. Then, the exposing vector \( W \in S_+^{n^2} \) defined in Lemma 5.3 can be written as \( W = \sum_{i=0}^{d} A_i \otimes W_i \), where

\[
W_0 = (n - 2)J + nI \quad \text{and} \quad W_i = nI_n - 2J \quad \text{for} \quad i = 1, \ldots, d. \tag{5.10}
\]

Let \( \tilde{W}_k \in S^n \) be the \( k \)-th block of \( \tilde{W} \), see (5.9). Then there are \( d + 1 \) distinct blocks given by

\[
\tilde{W}_k = \sum_{i=0}^{d} p_{i,k} W_i \in S^n \quad \text{for} \quad k = 0, \ldots, d, \quad \text{where} \quad p_{i,k} \text{ are elements in the character table} \ P \text{ of the Hamming scheme, see Example 2.5. Using the fact that} \ P e = (n, 0, \ldots, 0)^T \text{ and } p_{1,k} = 1, \text{ for every } k = 0, \ldots, d, \text{ we have}
\]

\[
\tilde{W}_0 = n^2I - nJ \quad \text{and} \quad \tilde{W}_k = nJ \quad \text{for} \quad k = 1, \ldots, d, \tag{5.11}
\]

and the matrices \( \tilde{V}_k \), whose columns form a basis of the null space of \( \tilde{W}_k \in S^n \), are given by

\[
\tilde{V}_0 = e_n \quad \text{and} \quad \tilde{V}_k = \begin{bmatrix} I_{n-1} \\ -e_{n-1}^T \end{bmatrix} \in \mathbb{R}^{n \times (n-1)} \quad \text{for} \quad k = 1, \ldots, d. \tag{5.12}
\]

Similarly, results can be derived when one uses different groups. Now we are ready to present an SDP relaxation for the QAP that is both facially and symmetry reduced.

Proposition 5.7. The facially reduced program of the symmetry reduced DNN relaxation (5.8) is given by

\[
\begin{align*}
\min & \quad \sum_{i=1}^{d} a_i \text{trace}(A_i A_i) \text{trace}(BY_i) \\
\text{s.t.} & \quad \sum_{i=0}^{d} \text{trace}(J A_i) \text{trace}(JY_i) = n^2 \\
& \quad \text{offDiag}(Y_0) = 0 \\
& \quad \text{diag}(Y_i) = 0, i = 1, \ldots, d \\
& \quad Y_j \succeq 0, j = 0, \ldots, d \\
& \quad \tilde{B}_k^*(y) = \tilde{V}_k \tilde{R}_k \tilde{V}_k^T, k = 1, \ldots, t \\
& \quad \tilde{R}_k \in S_+^{n'_k}, k = 1, \ldots, t.
\end{align*} \tag{5.13}
\]

Here, the columns of \( \tilde{V}_k \in \mathbb{R}^{n_k \times n'_k} \) form a basis of the null space of the \( \tilde{W}_k \in S^n \).

Proof. Applying Theorem 3.6 to the block-diagonal matrix \( (Q \otimes I)^T W (Q \otimes I) = \sum_{i=0}^{d} (Q^T A_i Q) \otimes W_i \), the matrices \( \tilde{W}_k \) are the exposing vectors of the symmetry reduced program (5.8), and thus \( \tilde{W}_k \tilde{B}_k^*(y) = 0 \) for every \( k = 1, \ldots, t \). This means that there exists a full column rank matrix \( \tilde{V}_k \in \mathbb{R}^{n_k \times n'_k} \) such that \( \tilde{B}_k^*(y) = \tilde{V}_k \tilde{R}_k \tilde{V}_k^T \), where \( \tilde{R}_k \in S_+^{n'_k} \) for every \( k = 1, \ldots, t \). Finally, we apply Corollary 3.21 to remove redundant constraints, see also Lemma 5.4. This yields the formulation (5.13).

Note that in the case that the basis elements \( A_i \) \((i = 0, \ldots, d)\) belong to the Hamming scheme, see Example 5.6, it follows that \( t = d + 1 \) in the above Proposition 5.7.

5.1.2 On solving QAP with ADMM

Now we discuss how to use ADMM to solve the DNN relaxation (5.13), for the particular case when \( A_i \) \((i = 0, 1, \ldots, d)\) form a basis of the Bose-Mesner algebra of the Hamming scheme. We proceed as in Section 4, and exploit properties of the known algebra, see Example 2.5. Clearly,
We obtain the following DNN algebra of the Hamming scheme. Let us present our reformulated

First, we derive the equivalent reformulation of the DNN relaxation (5.13), by exploiting the following.

1) Since we remove the repeating blocks of positive semidefinite constraints, to apply ADMM we have to reformulate the DNN in such a way that Assumption 4.1 is satisfied. Let us first derive an expression for the objective function as follows.

\[
\text{trace}((A \otimes B)Y) = \text{trace} \left( (Q \otimes I)^T (\sum_{i=0}^{d} a_i A_i \otimes B)(Q \otimes I) \right) (Q \otimes I)^T (\sum_{j=0}^{d} A_j \otimes Y_j) (Q \otimes I) \right) \\
= \text{trace} \left( \left( \sum_{i=0}^{d} (Q^T a_i A_i) \otimes B \right) \left( \sum_{j=0}^{d} (Q^T A_j) \otimes Y_j \right) \right) \\
= \sum_{k=0}^{d} \mu_k \text{trace} \left( \left( \sum_{i=0}^{d} a_i p_{i,k} B \right) \left( \sum_{j=0}^{d} p_{j,k} Y_j \right) \right) \\
= \sum_{k=0}^{d} (\tilde{C}_k, \sqrt{\mu_k} \sum_{i=0}^{d} p_{i,k} Y_i),
\]

where \( \tilde{C}_k := \sqrt{\mu_k} (\sum_{i=0}^{d} a_i p_{i,k}) B \). Recall that \( \mu = (\mu_k) \in \mathbb{R}^{d+1} \), with \( \mu_k := (d\choose k)(q-1)^k \). Then, we multiply the coupling constraints \( \tilde{B}_j^r(y) = \tilde{V}_j \tilde{R}_i \tilde{V}_j^T \) by the square root of its multiplicities. Thus, for the Bose-Mesner algebra, we end up with \( \sqrt{\mu_j} (\sum_{i=0}^{d} p_{i,j} Y_i - \tilde{V}_j \tilde{R}_j \tilde{V}_j^T) = 0 \).

2) In many applications, it is not necessary to compute high-precision solutions, and the ADMM can be terminated at any iteration. Then, one can use the dual variable \( \tilde{Z}_j \) from the current iteration to compute a valid lower bound, see Lemma 5.9. By adding redundant constraints, this lower bound is improved significantly when the ADMM is terminated with low-precision. Therefore we add the following redundant constraints

\[
Y_0 = \frac{1}{n} I, \quad \text{trace}(\tilde{R}_j) = \sqrt{\mu_j} p_{0,j} \text{ for } j = 0, \ldots, d.
\]

To see the redundancy of the last \( d+1 \) constraints above, we use the fact that the columns of \( \tilde{V}_j \) are orthonormal, and that \( \text{diag}(Y_i) = 0, i = 1, \ldots, d \), to derive

\[
\text{trace}(\tilde{R}_j) = \text{trace}(\tilde{V}_j \tilde{R}_j \tilde{V}_j^T) = \sqrt{\mu_j} (\sum_{i=0}^{d} p_{i,j} Y_i) = \sqrt{\mu_j} p_{0,j}.
\]

This technique can also be found in [26, 35, 44, 48].

We would like to emphasize that the techniques above are not restricted to the Bose-Mesner algebra of the Hamming scheme. Let us present our reformulated DNN relaxation for ADMM. Define

\[
P := \left\{ (Y_0, \ldots, Y_d) \mid \sum_{i=0}^{d} \binom{d}{i}(q-1)^i q^d \text{trace}(JY_i) = n^2, \quad Y_0 = \frac{1}{n} I, \quad \text{diag}(Y_i) = 0, Y_j \geq 0, i = 1, \ldots, d \right\},
\]

and

\[
\tilde{R} := \{ (\tilde{R}_0, \ldots, \tilde{R}_d) \mid \text{trace}(\tilde{R}_j) = \sqrt{\mu_j} p_{0,j}, \tilde{R}_i \in S^n_+, i = 0, \ldots, d \}.
\]

We obtain the following DNN relaxation for our ADMM

\[
p^* := \min \sum_{j=0}^{d} (\tilde{C}_j, \sqrt{\mu_j} \sum_{i=0}^{d} p_{i,j} Y_i) \\
\text{s.t.} \quad (Y_0, \ldots, Y_d) \in P \\
(\tilde{R}_0, \ldots, \tilde{R}_d) \in \tilde{R} \\
\sqrt{\mu_j} (\sum_{i=0}^{d} p_{i,j} Y_i - \tilde{V}_j \tilde{R}_j \tilde{V}_j^T) = 0, j = 0, \ldots, d.
\]

30
The augmented Lagrangian is
\[
\mathcal{L}(\tilde{Y}, \tilde{R}, \tilde{Z}) := \sum_{j=0}^{d} \left( (\tilde{C}_j, \sqrt{\mu_j} \sum_{i=0}^{d} p_{i,j} Y_i) + (\tilde{Z}_j, \sqrt{\mu_j} (\sum_{i=0}^{d} p_{i,j} Y_i - \tilde{V}_j \tilde{R}_j \tilde{V}_j^T) \right) + \frac{\beta}{2} \| \sqrt{\mu_j} (\sum_{i=0}^{d} p_{i,j} Y_i - \tilde{V}_j \tilde{R}_j \tilde{V}_j^T) \|^2.
\]

The $Y$-subproblem, the $\tilde{R}$-subproblem and the dual update are represented below.

1. The $Y$-subproblem:
\[
\begin{align*}
\min & \quad \sum_{j=0}^{d} \| \sqrt{\mu_j} \sum_{i=0}^{d} p_{i,j} Y_i - \sqrt{\mu_j} \tilde{V}_j \tilde{R}_j \tilde{V}_j^T + \frac{\beta}{2} \| \sqrt{\mu_j} (\sum_{i=0}^{d} p_{i,j} Y_i - \tilde{V}_j \tilde{R}_j \tilde{V}_j^T) \|^2 \\
\text{s.t.} & \quad Y_0 = \frac{1}{n} I \\
& \quad \text{diag}(Y_i) = 0, \ i = 1, \ldots, d \\
& \quad \sum_{i=0}^{d} (\begin{pmatrix} d & 1 \end{pmatrix} (q-1)^i q^d \text{trace}(J Y_i)) = n^2 \\
& \quad Y_i \geq 0, \ i = 0, \ldots, d.
\end{align*}
\] (5.18)

2. The $\tilde{R}$-subproblems, for $j = 0, \ldots, d$:
\[
\begin{align*}
\min & \quad \| \tilde{R}_j - \tilde{V}_j^T (\sum_{i=0}^{d} p_{i,j} Y_i + \frac{\tilde{Z}_j}{\text{diag}(\mu_j)}) \tilde{V}_j \| \\
\text{s.t.} & \quad \tilde{R}_j \in S^{n_j^2}.
\end{align*}
\] (5.19)

3. Update the dual variable:
\[
\tilde{Z}_j \leftarrow \tilde{Z}_j + \gamma \beta \sqrt{\mu_j} (\sum_{i=0}^{d} p_{i,j} Y_i - \tilde{V}_j \tilde{R}_j \tilde{V}_j^T), \quad j = 0, \ldots, d.
\] (5.20)

Clearly, the $\tilde{R}$-subproblems can be solved in the same way as (4.1). To see that the $Y$-subproblem can also be solved efficiently, let us show that it is a problem of the form (4.3), and thus satisfies Assumption 4.1.

Let $\lambda_j = (p_{0,j}, \ldots, p_{d,j})^T$, 
\[
y = \begin{bmatrix} \text{vec}(Y_0) \\ \vdots \\ \text{vec}(Y_d) \end{bmatrix} \quad \text{and} \quad \hat{y} = \begin{bmatrix} \text{vec}(\sqrt{\mu_0} \tilde{V}_0 \tilde{R}_0 \tilde{V}_0^T - \frac{\tilde{C}_0 + \tilde{Z}_0}{\beta}) \\ \vdots \\ \text{vec}(\sqrt{\mu_d} \tilde{V}_d \tilde{R}_d \tilde{V}_d^T - \frac{\tilde{C}_d + \tilde{Z}_d}{\beta}) \end{bmatrix}.
\]

Define the linear transformation $\mathcal{T}^* : \mathbb{R}^{(d+1)n^2} \rightarrow \mathbb{R}^{(d+1)n^2}$ by
\[
\mathcal{T}^*(y) = \begin{bmatrix} \sqrt{\mu_0} (\lambda_0^T \otimes I_{n^2}) \\ \vdots \\ \sqrt{\mu_d} (\lambda_d^T \otimes I_{n^2}) \end{bmatrix} y.
\]

**Lemma 5.8.** The $Y$-subproblem (5.18) is equivalent to the following projection to the weighted simplex problem
\[
\begin{align*}
\min & \quad \| \mathcal{T}^*(y) - \hat{y} \|^2 \\
\text{s.t.} & \quad y_i = 0, \ i \in \mathcal{I} \\
& \quad w^T y = n^2 \\
& \quad y \geq 0,
\end{align*}
\] (5.21)

where $w := q^d (\mu \otimes c_{n^2}) \in \mathbb{R}^{(d+1)n^2}$, and $\mathcal{I}$ contains the indices of $y$ associated to the off-diagonal entries of $Y_0$. Furthermore, the problem (5.21) satisfies Assumption 4.1.
Proof. One can verify that (5.18) and (5.21) are equivalent. Furthermore, it holds that
\[ T(T^*(y)) = \left( \sum_{j=0}^{d} \mu_j \lambda_j T^2 \right) \]  
(5.17)  
Proof. One can verify that (5.18) and (5.21) are equivalent. Furthermore, it holds that
\[ T(T^*(y)) = \left( \sum_{j=0}^{d} \mu_j \lambda_j T^2 \right) \]  
(5.17)  
Proof. One can verify that (5.18) and (5.21) are equivalent. Furthermore, it holds that
\[ T(T^*(y)) = \left( \sum_{j=0}^{d} \mu_j \lambda_j T^2 \right) \]  
(5.17)  
Proof. One can verify that (5.18) and (5.21) are equivalent. Furthermore, it holds that
\[ T(T^*(y)) = \left( \sum_{j=0}^{d} \mu_j \lambda_j T^2 \right) \]  
(5.17)  
Proof. One can verify that (5.18) and (5.21) are equivalent. Furthermore, it holds that
\[ T(T^*(y)) = \left( \sum_{j=0}^{d} \mu_j \lambda_j T^2 \right) \]  
(5.17)
Let \( p_k \) be the ADMM objective value, and \( d_k := g(\tilde{Z}) \) the dual objective value at some dual feasible point at the \( k \)-th iteration, see (5.22). If the duality gap is not improving significantly, i.e.,

\[
gap = \frac{p_{100k} - d_{100k}}{1 + p_{100k} + d_{100k}} < 10^{-4},
\]

for 20 consecutive integers \( k \), then we conclude that there is stagnation in the objective value. We measure the gap only every 100-th iteration due to the expense of computing the dual objective value \( d_k \).

In our QAP experiments, we use \( \epsilon = 10^{-12} \) if \( n \leq 128 \), and \( \epsilon = 10^{-5} \) when \( n = 256, 512 \). The objective value from the ADMM is denoted by \( \text{OBJ} \), and the valid lower bound obtained from the dual feasible solution is denoted by \( \text{LB} \), see Lemma 5.9. The running times in all tables are reported in seconds. We also list the maximum of the primal and dual residuals, i.e., \( \text{res} := \max\{\text{pres}, \text{dres}\} \). If a result is not available, we put - in the corresponding entry.

1. The first set of test instances are from Mittelmann and Peng [42], where the authors compute SDP bounds for the QAP with \( A \) being the Hamming distance matrix. Choices of the matrix \( B \) differ for different types of instances. In particular, in the Harper instance Harper\_n where \( n = 2^d \) we have \( B_{ij} = |i - j| \) for all \( i, j = 1, \ldots, 2^d \). Further eng1\_n and end9\_n with \( n = 2^d, d = 4, \ldots, 9 \) refer to the engineering problems, and VQ\_n instances have random matrices \( B \). For details see [42]. In rand\_256 and rand\_512 instances, \( A \) is the Hamming distance matrix of appropriate size and \( B \) is a random matrix.

In the first column of Table 5.1 we list the instance names where the sizes of the QAP matrices are indicated after the underscore. Upper bounds are given in the column two. For instances with up to 128 nodes we list the upper bounds computed in [42], and for the remaining instances we use our heuristics. Since data matrices for the Harper instances are integer, we round up lower bounds to the closest integer. In the column three (resp. four) we list SDP-based lower bounds (resp. computation times in seconds) from [42]. The bounds from [42] are obtained by solving an SDP relaxation having several matrix variables on order \( n \). The bounds in [42] were computed on a 2.67GHz Intel Core 2 computer with 4GB memory. In the columns five to seven, we present the results obtained by using our ADMM algorithm.

Table 5.1 shows that we significantly improve bounds for all eng1\_n and eng9\_n instances. Moreover, we are able to compute bounds for huge QAP instances with \( n = 256 \) and \( n = 512 \) in a reasonable amount of time. Recall that for given \( n \), the order of the matrix variable in the DNN relaxation of the QAP (5.1) is \( n^2 \). However, for each instance \( xx\_n \) of Table 5.1 we have that \( n = 2^d \), and that the DNN relaxation (5.1) boils down to \( d + 1 \) positive semidefinite blocks of order \( n \). In particular, we obtain the bound for each instance in Table 5.1 by solving the facially and symmetry reduced DNN relaxation (5.13) where \( \tilde{B}_k^*(y) \in \mathcal{S}_+^n, k = 1, \ldots, d + 1 \).

\(^7\)We thank Hans Mittelman for providing us generators for the mentioned instances.
| problem | UB    | OBJ    | OBJ LB  | time  | res.  |
|---------|-------|--------|--------|-------|-------|
| Harper_16 | 2752  | 2743   | 2742   | 1.92  | 4.50e-05 |
| Harper_32 | 27360 | 27331  | 27327  | 9.70  | 1.67e-04 |
| Harper_64 | 262260| 262196 | 262160 | 56    | 36.12  |
| Harper_128 | 2479944| 2446800| 2446588| 1491  | 186.12 |
| Harper_256 | 22370940| 2236996| 2236996| -     | 36.12  |
| Harper_512 | 201329908| 201327683| 201327683| -     | 1903.66 |
| eng1_16    | 1.58049| 1.5741 | 1.5740 | 2.28  | 3.87e-05 |
| eng1_32    | 1.58528| 1.5669 | 1.5637 | 14.63 | 5.32e-06 |
| eng1_64    | 1.58297| 1.5444 | 1.5401 | 38.35 | 4.69e-06 |
| eng1_128   | 1.59692| 1.4983 | 1.4870 | 389.04| 2.37e-06 |
| eng1_256   | 1.57995| 1.4820 | 1.3222 | 971.48| 9.95e-06 |
| eng1_512   | 1.53431| 1.4553 | 1.3343 | 9220.13| 9.66e-06 |
| eng9_16    | 1.02017| 1.0014 | 1.0013 | 3.38  | 2.11e-06 |
| eng9_32    | 1.40941| 1.3507 | 1.3490 | 12.67 | 3.80e-06 |
| eng9_64    | 1.43201| 1.3534 | 1.3489 | 74.89 | 6.60e-06 |
| VQ_32      | 297.29 | 296.3241| 296.3241| 11.82 | 1.27e-05 |
| VQ_64      | 353.5  | 352.7621| 352.7621| 43.17 | 4.22e-04 |
| VQ_128     | 399.09 | 398.4269| 398.4269| 575.94| 8.46e-06 |
| rand_256   | 126630.6273| 124589.4215| 124589.4215| 2054.6143| 3.78e-05 |
| rand_512   | 577604.8759| 570935.1468| 570935.1468| 9694.7143| 1.32e-04 |

Table 5.1: Lower and upper bounds for different QAP instances.

2. The second set of test instances are Eschermann, Wunderlich, esc, instances from the QAPLIB library [9]. In esc_nx instance, the distance matrix $A$ is the Hamming distance matrix of order $n = 2^d$, whose automorphism group is the automorphism group of the Hamming graph $H(d, 2)$. In [15] the authors exploit symmetry in esc instances to solve the DNN relaxation (5.8) by the interior point method. That was the first time that SDP bounds for large QAP instances were computed by exploiting symmetry. In particular, the authors from [15] needed 13 seconds to compute the SDP bound for esc64a, and 140 seconds for computing the esc128 SDP bound, see also Table 5.2. The bounds in [15] are computed by the interior point solver SeDuMi [57] using the Yalmip interface [38] and Matlab 6.5, implemented on a PC with Pentium IV 3.4 GHz dual-core processor and 3GB of memory. Computational times in [15] include only solver time, not the time needed for Yalmip to construct the problem.

In [44] the authors approximately solve the DNN relaxation (5.8) using the ADMM algorithm, but do not exploit symmetry. Here, we compare computational results from [44] with the approach we present in this paper. All the instances from [44] were tested on an Intel Xeon Gold 6130 2.10 Ghz PC with 32 cores and 64 GB of memory and running on 64-bit Ubuntu system.

An efficient solver, called SDPNAL+, for solving large scale SDPs is presented in [59, 68]. SDPNAL+ implements an augmented Lagrangian based method. In particular, the implementation in [68] is based on a majorized semismooth Newton-CG augmented Lagrangian method, and the implementation in [59] is based on an inexact symmetric Gauss-Seidel based semi-proximal ADMM. In [59, 68], the authors present extensive numerical results that also include solving (5.1) on various instances from the QAPLIB library [9]. However,
they do not perform FR and SR. In Table 5.2 we include results from [59] for solving esc_nx, with \( n = 16, 32 \). There are no results for \( n = 64, 125 \) presented in their paper. Moreover the authors emphasize that SDPNAL+ is for solving SDPs where the maximum matrix dimension is assumed to be less than 5000. Due to the use of different computers, the times in Table 5.2 are not comparable. For example, the authors from [59] use an Intel Xeon CPU E5-2680v3, 2.50 GHz with 12 cores and 128 GB of memory.

In Table 5.2 we present the numerical result for the esc instances. In particular, we compare bounds and computational times of the relaxation (5.1) (no reductions, solved in [59]), the facially reduced relaxation (5.6) (solved in [44]), the symmetry reduced relaxation (5.8) (solved in [15]), and facially and symmetry reduced relaxation (5.13) (solved by our ADMM).

We conclude that:

1. There are notably large differences in computation times between the ADMM algorithm presented here and the one from [44], since the latter does not exploit symmetry.

2. Even if the use of different computers is taken into account, this would likely not be enough to account for the time differences observed between our ADMM and SDPNAL+ [59]. Moreover, SDPNAL+ was not able to solve several instances.

3. In [15], the authors use SeDuMi to solve a relaxation equivalent to the symmetry reduced program (5.8); and they obtain a LB of 53.0844 for esc128. However, the bounds for this instance for the facially and symmetry reduced program (5.13) computed by the Mosk interior point method solver is 51.7516; and our ADMM algorithm reports 51.7518. This illustrates our improved numerical accuracy using FR and SR, and validates the statements about singularity degree, see Section 3.4. We note in addition that we provide a theoretically guaranteed lower bound, as well as solve huge instances that are intractable for the approach in [15].

| inst. | opt | SDPNAL+ STYZ [59] | ADMM OWX [44] | SDP KS [15] | ADMM |
|-------|-----|-----------------|---------------|-------------|-----|
| esc16a | 68 | 63.2750 16 | 64 20.14 | 63.2756 0.75 | 63.2856 63.2856 2.48 1.17e-11 |
| esc16b | 292 | 289.9730 24 | 290 3.10 | 289.8817 1.04 | 290.0000 290.0000 0.78 9.95e-13 |
| esc16c | 160 | 153.9619 65 | 154 8.44 | 153.8242 1.78 | 154.0000 153.9999 2.11 2.56e-09 |
| esc16d | 16 | 13.0000 2 | 13 17.39 | 13.0000 0.89 | 13.0000 13.0000 1.04 9.94e-13 |
| esc16e | 28 | 26.3367 2 | 27 24.04 | 26.3368 0.51 | 26.3368 26.3368 1.21 9.89e-13 |
| esc16f | - | - | 0 3.22e+02 | 0 0.14 | 0 0 0.01 2.53e-14 |
| esc16g | 26 | 24.7388 4 | 25 33.54 | 24.7403 0.51 | 24.7403 24.7403 1.40 9.95e-13 |
| esc16h | 996 | 976.1857 10 | 977 4.01 | 976.2244 0.79 | 976.2293 976.2293 2.51 7.73e-13 |
| esc16i | 14 | 11.3749 6 | 12 100.79 | 11.3749 0.73 | 11.3749 11.3660 6.15 2.53e-06 |
| esc16j | 8 | 7.7938 4 | 8 56.90 | 7.7942 0.42 | 7.7942 7.7942 0.21 9.73e-13 |
| esc32a | 130 | 103.3206 333 | 104 2.89e+03 | 103.3194 114.88 | 103.3211 103.0465 12.36 3.62e-06 |
| esc32b | 168 | 131.8532 464 | 132 5.2e+03 | 131.8718 5.58 | 131.8843 131.8843 4.64 9.59e-13 |
| esc32c | 642 | 615.1600 331 | 616 4.48e+02 | 615.1400 3.70 | 615.1813 615.1813 8.04 2.05e-10 |
| esc32d | 200 | 190.2273 67 | 191 8.68e+02 | 190.2266 2.09 | 190.2271 190.2263 5.86 7.45e-08 |
| esc32e | 2 | 1.9001 149 | 2 1.81e+03 | - - | 1.9000 1.9000 0.70 4.49e-13 |
| esc32f | 6 | 5.8336 65 | 6 6.04e+02 | 5.8330 1.80 | 5.8333 5.8333 3.50 9.97e-13 |
| esc32g | 438 | 424.3256 1076 | 425 3.02e+03 | 424.3382 17.16 | 424.4027 424.3184 5.89 1.03e-08 |
| esc64a | 116 | 9.8336 98 | 98 1.64e+04 | 97.7490 12.99 | 97.7500 97.7500 5.33 8.95e-13 |
| esc128 | 64 | - - | - - | 53.0844 140.36 | 51.7518 51.7518 137.71 1.18e-12 |

Table 5.2: Esc instances (times with different computers).
5.2 The graph partition problem (GP)

The graph partition problem is the problem of partitioning the vertex set of a graph into a fixed number of sets of given sizes such that the sum of edges joining different sets is minimized. The problem is known to be NP-hard. The GP has many applications such as VLSI design, parallel computing, network partitioning, and floor planing. Graph partitioning also plays a role in machine learning (see e.g., [34]) and data analysis (see e.g., [47]). There exist several SDP relaxations for the GP of different complexity and strength, see e.g., [30,53,54,63,67].

5.2.1 The general GP

Let $G = (V, E)$ be an undirected graph with vertex set $V$, $|V| = n$ and edge set $E$, and $k \geq 2$ be a given integer. We denote by $A$ the adjacency matrix of $G$. The goal is to find a partition of the vertex set into $k$ (disjoint) subsets $S_1, \ldots, S_k$ of specified sizes $m_1 \geq \ldots \geq m_k$, where $\sum_{j=1}^k m_j = n$, such that the sum of weights of edges joining different sets $S_j$ is minimized. Let

$$P_m := \{ S = (S_1, \ldots, S_k) \mid |S_i| = m_i, \forall i, \cup_{i=1}^k S_i = V \} \quad (5.25)$$

denote the set of all partitions of $V$ for a given $m = (m_1, \ldots, m_k)$. In order to model the GP in binary variables we represent the partition $S \in P_m$ by the partition matrix $X \in \mathbb{R}^{n \times k}$ where the column $j$ is the incidence vector for the set $S_j$.

The GP can be stated as follows

$$\min_{X \in \mathcal{M}_m} \frac{1}{2} \text{trace}(AX(J_k - I_k)X^T),$$

where

$$\mathcal{M}_m = \{ X \in \{0, 1\}^{n \times k} \mid Xe_k = e_n, X^Te_n = m \} \quad (5.26)$$

is the set of partition matrices.

Here, we consider the following DNN relaxation that is equivalent to the relaxation from [67]:

$$\min \frac{1}{2} \text{trace}((A \otimes B)Y)$$

s.t.

$$\mathcal{G}(Y) = 0$$

$$\text{trace}(D_1 Y) - 2(e_n \otimes e_k)^T \text{diag}(Y) + n = 0$$

$$\text{trace}(D_2 Y) - 2(e_n \otimes m)^T \text{diag}(Y) + m^T m = 0$$

$$D_0(Y) = \text{Diag}(m)$$

$$D_e(Y) = e$$

$$\langle J, Y \rangle = n^2$$

$$Y \geq 0, Y \succeq 0,$$

where $B = J_k - I_k$, and

$$Y = \begin{bmatrix} Y^{(11)} & \ldots & Y^{(1n)} \\ \vdots & \ddots & \vdots \\ Y^{(n1)} & \ldots & Y^{(nn)} \end{bmatrix} \in \mathbb{S}^{kn}$$

with each $Y^{(ij)}$ being a $k \times k$ matrix, and

$$D_1 = I_n \otimes J_k$$

$$D_2 = J_n \otimes I_k$$

$$D_0(Y) = \sum_{i=1}^n Y^{ii} \in \mathbb{S}^k$$

$$D_e(Y) = (\text{trace} Y^{ii}) \in \mathbb{R}^n$$

$$\mathcal{G}(Y) = \langle I_n \otimes (J_k - I_k), Y \rangle.$$
Here $\mathcal{G}(\cdot)$ is the gangster operator for the $\mathbf{GP}$. To compute $\mathbf{DNN}$ bounds for the $\mathbf{GP}$, we apply $\mathbf{FR}$ for symmetry reduced relaxation (5.27). The details are similar to the $\mathbf{QAP}$, and thus omitted.

We present numerical results for different graphs from the literature. Matrix can161 is from the library Matrix Market [4], matrix grid3dt5 is 3D cubical mesh, and gridt$xx$ matrices are 2D triangular meshes. Myciel7 is a graph based on the Mycielski transformation and 1.FullIns4 graph is a generalization of the Mycielski graph. Both graphs are used in the COLOR02 symposium [28].

1. In Table 5.3

| instance   | $|V|$ | # orbits           | blocks of $A$             | $m$       |
|------------|------|--------------------|---------------------------|-----------|
| 1.FullIns4 | 93   | 3629               | (53,27,9,3,1)             | (30,31,32) |
| can161     | 161  | 921                | (20,20,20,20,20,20,11,10) | (52,53,56) |
| grid3dt5   | 125  | 4069               | (39,36,26,24)             | (40,41,44) |
| grid15     | 120  | 2432               | (80,24,16)                | (39,40,41) |
| grid17     | 153  | 3942               | (102,30,21)               | (50,51,52) |
| myciel7    | 191  | 6017               | (64,64,63)                | (62,63,66) |

Table 5.3: Graphs and partitions.

2. In Table 5.4

| instance   | IPM (Symmetry & Facial reduced) | ADMM ($\epsilon = 10^{-3}$) |
|------------|---------------------------------|-----------------------------|
|            | LB                              | OBJ | LB           | time | res     | time | res     |
|------------|---------------------------------|-----|--------------|------|---------|------|---------|
| 1.FullIns4 | 194.2825                        | 311.95 | 26          | 194.2686 | 194.0523 | 141.29 | 1.50e-01 |
| can161     | 33.0151                         | 124.32 | 19          | 33.0392  | 30.4470  | 19.74  | 2.58e-01 |
| grid3dt5   | 68.3175                         | 245.65 | 17          | 68.3029  | 68.0436  | 200.35 | 2.02e-01 |
| grid15     | 12.1153                         | 1302.10 | 41         | 12.1116  | 11.3654  | 97.17  | 1.91e-01 |
| grid17*    | 12.2482                         | 1865.67 | 21         | 12.2532  | 11.1459  | 357.53 | 1.80e-01 |
| myciel7*   | 1126.0309                       | 2579.65 | 17         | 1126.0385 | 1123.8526 | 553.67 | 9.50e-02 |

Table 5.4: Numerical results for the graph 3-partition.

we provide information on the graphs and the considered 3-partition problems. In particular, the first column specifies graphs, the second column provides the number of vertices in a graph, the third column is the number of orbits after symmetrization, the fourth column lists the number of blocks in $Q^T A Q$. Here, the orthogonal matrix $Q$ is computed by using the heuristic from [15]. The last column specifies sizes of partitions.

2. In Table 5.4

we list lower bounds obtained by using Mosek and our ADMM algorithm. The table also presents computational times required to compute bounds by both methods as well as the number of interior point method iterations. The results show that the ADMM with precision $\epsilon = 10^{-3}$ provides competitive bounds in much shorter time than the interior point method solver. In Table 5.4, some instances are marked by *. This means that our 64GB machine did not have enough memory to solve these instances by the interior point method solver, and therefore they are solved on a machine with an Intel(R) Xeon(R) Gold 6126, 2.6 GHz quad-core processor and 192GB of memory. However, the ADMM algorithm has much lower memory requirements, and thus the ADMM is able to solve all instances from Table 5.4 on the smaller machine.
5.2.2 The vertex separator problem (VSP) and min-cut (MC)

The min-cut problem is the problem of partitioning the vertex set of a graph into \( k \) subsets of given sizes such that the number of edges joining the first \( k-1 \) partition sets is minimized. The **MC** problem is a special case of the general **GP** but also arises as a subproblem of the vertex separator problem. The vertex separator problem is to find a subset of vertices (called vertex separator) whose removal disconnects the graph into \( k-1 \) components. This problem is NP-hard.

The vertex separator problem was studied by Helmberg, Mohar, Poljak and Rendl [25], Povh and Rendl [49], Rendl and Sotirov [51], Pong, Sun, Wang, Wolkowicz [48]. The **VSP** appears in many different fields such as VLSI design [3] and bioinformatics [20]. Finding vertex separators of minimal size is an important problem in communication networks [33] and finite element methods [41]. The VSP also appears in divide-and-conquer algorithms for minimizing the work involved in solving systems of equations, see e.g., [36,37].

Let us formally relate the **VSP** and the **MC** problem. Let \( \delta(S_i, S_j) \) denote the set of edges between \( S_i \) and \( S_j \), where \( S_i \) and \( S_j \) are defined as in (5.25). We denote the set of edges with endpoints in distinct partition sets \( S_1, \ldots, S_{k-1} \) by

\[
\delta(S) = \cup_{i<j<k} \delta(S_i, S_j).
\]

The min-cut problem is

\[
\text{cut}(m) = \min \{|\delta(S)| \mid S \in P_m\}.
\]

The graph has a vertex separator if there exists \( S \in P_m \) such that after the removal of \( S_k \) the induced subgraph has no edges across \( S_i \) and \( S_j \) for \( 1 \leq i < j < k \). Thus, if \( \text{cut}(m) = 0 \) or equivalently \( \delta(S) = \emptyset \), there exists a vertex separator. On the other hand \( \text{cut}(m) > 0 \) shows that no separator \( S_k \) for the cardinalities specified in \( m \) exists.

Clearly, \( |\delta(S)| \) can be represented in terms of a quadratic function of the partition matrix \( X \), i.e., as \( \frac{1}{2} \text{trace}(AXBX^T) \) where

\[
B := \left[ \begin{array}{cc}
J_{k-1} - I_{k-1} & 0 \\
0 & 0 \\
0 & 0
\end{array} \right] \in S^k.
\] (5.28)

Therefore,

\[
\text{cut}(m) = \min_{X \in M_m} \frac{1}{2} \text{trace}(AXBX^T),
\]

where \( M_m \) is given in (5.26). To compute **DNN** bounds for the **MC** problem and provide bounds for the vertex separator problem, we use the **DNN** relaxation (5.27) with \( B \) defined in (5.28).

1. We present numerical results for the Queen graphs, where the \( n \times n \) Queen graph has the squares of an \( n \times n \) chessboard for its vertices and two such vertices are adjacent if the corresponding squares are in the same row, column, or diagonal. The instances in this class come from the DIMACS challenge on graph coloring. In Table 5.5 we provide information on the Queen graphs. The table is arranged in the same way as Table 5.3.
2. In Table 5.6 we provide the numerical results for the vertex separator problem. More specifically, we are computing the largest integer $m_3$ such that the solution value of the DNN relaxation (5.27) is positive with partition

$$m = \left(\left\lfloor \frac{n - m_3}{2} \right\rfloor, \left\lceil \frac{n - m_3}{2} \right\rceil, m_3\right). \quad (5.29)$$

Then $m_3 + 1$ is a lower bound for the vertex separator problem with respect to the choice of $m$. One may tend to solve (5.27) for all possible $m_3$ between 0, 1, ..., $|V| - 1$ to find the largest $m_3$ for which the DNN bound is positive. However, the optimal value of (5.27) is monotone in $m_3$, and thus we find the appropriate $m_3$ using binary search starting with $m_3 = \left\lceil \frac{n}{2} \right\rceil$. We present the lower bound on the vertex separator, i.e., $m_3 + 1$ in the third column of Table 5.6. The total number of problems solved is listed in the fourth column of the same table. The running time given in the last two columns is the total amount of time used to find a positive lower bound for (5.27) for some $m_3$ by using Mosek and our ADMM algorithm, respectively. This task is particularly suitable for the ADMM, as we can terminate the ADMM once the lower bound in an iterate is positive. For example, it takes 786 seconds to solve the min-cut relaxation on queen12_12 by Mosek, see Table 5.7. However, though not shown in the table, it takes ADMM only 120 seconds to conclude that the optimal value is positive.

| instance | $|V|$ | $m_3 + 1$ | #problems | IPM (Symmetry&Facially reduced) time | ADMM ($\epsilon = 10^{-12}$) time |
|----------|------|----------|-----------|-------------------------------------|----------------------------------|
| queen5_5 | 25   | 17       | 4         | 7.49                                | 2.69                             |
| queen6_6 | 36   | 24       | 5         | 9.62                                | 2.91                             |
| queen7_7 | 49   | 32       | 5         | 25.34                               | 5.95                             |
| queen8_8 | 64   | 42       | 6         | 85.72                               | 34.35                            |
| queen9_9 | 81   | 53       | 6         | 304.44                              | 64.10                            |
| queen10_10 | 100 | 65       | 7         | 1309.85                             | 131.66                           |
| queen11_11 | 121 | 79       | 7         | 3416.01                             | 387.38                           |
| queen12_12 | 144 | 94       | 7         | 6147.20                             | 671.02                           |
| queen13_13 | 169 | 110      | 8         | -                                   | 1352.17                          |

Table 5.6: The vertex separator problem on the Queen graphs.

3. In Table 5.7 we compare bounds and computational times required to solve, for fixed $m$, symmetry reduced DNN relaxation (5.27) by the interior point algorithm, as well as symmetry and facially reduced relaxation (5.27) by using Mosek and our ADMM algorithm.
We conclude from Tables 5.6 and 5.7 that

- For small instances, the interior point algorithm is faster than the ADMM as shown in Table 5.7. For larger instances, the interior point algorithm has memory issues. However, the ADMM algorithm can still handle large instances due to its low memory demand.

- To obtain bounds on the vertex separator of a graph, one does not need to solve the DNN relaxation to high-precision. The ADMM is able to exploit this fact, and find a lower bound on the size of the vertex separator in significantly less amount of time than the interior point algorithm, see Table 5.6.

- The symmetry reduced program without FR is heavily ill-conditioned, and the interior point method is not able to solve it correctly for any of the instances. The running time is also significantly longer than the symmetry and facially reduced program, see Table 5.7.

Note that we have solved the queen10_10 problem with high accuracy with FR. The distance between the optimal solutions in norm was very large with no decimals of accuracy. This emphasizes the importance of FR in obtaining accuracy in solutions, see e.g., [56].

### Table 5.7: The min-cut problem on the Queen graphs.

| instance   | LB    | time | iter. | OBJ LB | OBJ | LB | time | res   |
|------------|-------|------|-------|--------|-----|-----|------|-------|
| queen5_5   | 0.0908| 1.04 | 38    | 0.1658 | 0.27| 10  | 0.1658| 6.88  |
| queen6_6   | 0.0962| 3.43 | 31    | 0.1411 | 0.91| 11  | 0.1411| 11.37 |
| queen7_7   | 0.5424| 15.42| 32    | 0.6196 | 1.92| 10  | 0.6196| 17.97 |
| queen8_8   | 0.1967| 127.60| 39    | 0.3087 | 7.38| 13  | 0.3087| 61.50 |
| queen9_9   | 0.0698| 377.77| 32    | 0.2175 | 19.98| 12  | 0.2175| 204.39|
| queen10_10 | 0.8159| 1664.09| 42    | 1.0211 | 85.42| 14  | 1.0211| 239.75|
| queen11_11 | -     | -    | -     | 0.2131 | 275.20| 16  | 0.2131| 877.85|
| queen12_12 | -     | -    | -     | 0.3248 | 786.12| 25  | 0.3248| 1474.45|
| queen13_13 | -     | -    | -     | -      | -    | -   | 0.9261| 1864.30|

6 Conclusion

In this paper we propose a method to efficiently implement facial reduction to the symmetry reduced SDP relaxation, and we demonstrated the efficiency by solving large scale NP-hard problems. More specifically, if an exposing vector of the minimal face for the input SDP is given, then we are able to construct an exposing vector of the minimal face for the symmetry reduced SDP. The resulting relaxation is symmetry reduced, satisfies the Slater condition, and thus can be solved with improved numerical stability.

We then extend our reduction technique to doubly nonnegative, DNN, programs. In fact, our approach allows the matrix variable of the original SDP, to be passed to simple nonnegative vector for the DNN. Again we exploit exposing vectors of DNN as a decomposition into a sum of a semidefinite and nonnegative exposing vectors. Further, we discuss the importance of the order of the reductions in our theory. We also show that the singularity degree of the symmetry reduced program is equal to the singularity degree of the original program.

We apply our technique to many combinatorial problems and their DNN relaxations, i.e., we facially and symmetry reduce them. The obtained relaxations can be solved extremely efficiently using the alternating direction method of multipliers. We also show that interior point methods
are more efficient on a symmetry and facially reduced relaxation. As a result, we are able to compute improved lower bounds for some QAP instances in significantly less time.

References

[1] E.D. Andersen and K.D. Andersen. The mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In High performance optimization, pages 197–232. Springer, 2000. 25

[2] C. Bachoc, D.C. Gijswijt, A. Schrijver, and F. Vallentin. Invariant semidefinite programs. In Handbook on semidefinite, conic and polynomial optimization, volume 166 of Internat. Ser. Oper. Res. Management Sci., pages 219–269. Springer, New York, 2012. 3, 5

[3] S.N. Bhatt and F.T. Leighton. A framework for solving vlsi graph layout problems. Journal of Computer and System Sciences, 28(2):300–343, 1984. 38

[4] R. F. Boisvert, R. Pozo, K. Remington, R. F. Barrett, and J. J. Dongarra. Matrix market: a web resource for test matrix collections. In Quality of Numerical Software, pages 125–137. Springer, 1997. 37

[5] J.M. Borwein and H. Wolkowicz. Characterization of optimality for the abstract convex program with finite-dimensional range. J. Austral. Math. Soc. Ser. A, 30(4):390–411, 1980/81. 3, 4

[6] J.M. Borwein and H. Wolkowicz. Facial reduction for a cone-convex programming problem. J. Austral. Math. Soc. Ser. A, 30(3):369–380, 1980/81. 3, 4

[7] J.M. Borwein and H. Wolkowicz. Regularizing the abstract convex program. J. Math. Anal. Appl., 83(2):495–530, 1981. 3, 4, 20

[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Machine Learning, 3(1):1–122, 2011. 22

[9] R.E. Burkard, S.E. Karisch, and F. Rendl. QAPLIB—a quadratic assignment problem library. J. Global Optim., 10(4):391–403, 1997. 34

[10] Yunmei Chen and Xiaojing Ye. Projection onto a simplex, 2011. 24

[11] L. Condat. Fast projection onto the simplex and the $l_1$ ball. Mathematical Programming, 158(1-2):575–585, 2016. 24

[12] E. de Klerk. Exploiting special structure in semidefinite programming: A survey of theory and applications. European J. Oper. Res., 201(1):1–10, 2010. 5, 6

[13] E. de Klerk, Fernando M. de Oliveira F., and D.V. Pasechnik. Relaxes of combinatorial problems via association schemes. In Handbook on semidefinite, conic and polynomial optimization, volume 166 of Internat. Ser. Oper. Res. Management Sci., pages 171–199. Springer, New York, 2012. 3

[14] E. de Klerk, C. Dobre, and D.V. Pasechnik. Numerical block diagonalization of matrix *-algebras with application to semidefinite programming. Math. Program., 129(1, Ser. B):91–111, 2011. 6
[15] E. de Klerk and R. Sotirov. Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem. *Math. Program.*, 122(2, Ser. A):225–246, 2010.
6, 21, 25, 26, 28, 34, 35, 37

[16] E. de Klerk and R. Sotirov. Improved semidefinite programming bounds for quadratic assignment problems with suitable symmetry. *Mathematical programming*, 133(1-2):75–91, 2012. 6, 28

[17] P. Delsarte. *An Algebraic Approach to the Association Schemes of Coding Theory*. PhD thesis, Universite Catholique de Louvain, Louvain, France, 1973. 7

[18] D. Drusvyatskiy and H. Wolkowicz. The many faces of degeneracy in conic optimization. *Foundations and Trends® in Optimization*, 3(2):77–170, 2017. 3, 4, 5, 9, 10, 12, 19

[19] J. Eckstein. Deriving solution value bounds from the ADMM. *Optimization Letters*, 2020. 32

[20] B. Fu, S.A. Oprisan, and L. Xu. Multi-directional width-bounded geometric separator and protein folding. In *International Symposium on Algorithms and Computation*, pages 995–1006. Springer, 2005. 38

[21] K. Fujisawa, M. Kojima, and K. Nakata. Exploiting sparsity in primal-dual interior-point methods for semidefinite programming. *Math. Program.*, 79:235–253, 1997. 3

[22] K. Gatermann and P.A. Parrilo. Symmetry groups, semidefinite programs, and sums of squares. *Journal of Pure and Applied Algebra*, 192(1-3):95–128, 2004. 5

[23] K. Gatermann and P.A. Parrilo. Symmetry groups, semidefinite programs, and sums of squares. *J. Pure Appl. Algebra*, 192(1-3):95–128, 2004. 6

[24] D. Gijswijt. Matrix algebras and semidefinite programming techniques for codes. *PhD Thesis*, 2010. 5

[25] C. Helmberg, F. Rendl, B. Mohar, and S. Poljak. A spectral approach to bandwidth and separator problems in graphs. *Linear and Multilinear Algebra*, 39(1-2):73–90, 1995. 38

[26] H. Hu and R. Sotirov. On solving the quadratic shortest path problem. *INFORMS Journal on Computing*, 2019. 30

[27] B. Jargalsaikhan and J.J. Rückmann. A note on strict complementarity for the doubly non-negative cone. *Optimization*, 68(2-3):457–464, 2019. 13

[28] D. S. Johnson, A. Mehrotra, and M. A. Trick. COLOR02/03/04: Graph coloring and its generalizations, 2002. 37

[29] Y. Kanno, M. Ohsaki, K. Murota, and N. Katoh. Group symmetry in interior-point methods for semidefinite program. *Optimization and Engineering*, 2:293–320, 2001. 11

[30] S.E. Karisch and F. Rendl. Semidefinite programming and graph equipartition. In *Topics in Semidefinite and Interior-Point Methods*, volume 18 of *The Fields Institute for Research in Mathematical Sciences, Communications Series*, Providence, Rhode Island, 1998. American Mathematical Society. 36
[31] M. Kojima, S. Kojima, and S. Hara. Linear algebra for semidefinite programming. Technical Report 1004, Dept. of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, 1997. Linear matrix inequalities and positive semidefinite programming (Japanese) (Kyoto, 1996). 11

[32] V. Kungurtsev and J. Marecek. A two-step pre-processing for semidefinite programming, 2020. 3

[33] F.T. Leighton. Complexity issues in vlsi: Optimal layouts for the shuffle exchange and other networks, 1983. 38

[34] M. Li, D. G. Andersen, and A. J. Smola. Graph partitioning via parallel submodular approximation to accelerate distributed machine learning. arXiv preprint arXiv:1505.04636, 2015. 36

[35] X. Li, T.K. Pong, H. Sun, and H. Wolkowicz. A strictly contractive Peaceman-Rachford splitting method for the doubly nonnegative relaxation of the minimum cut problem. Comput. Optim. Appl., 78(3):853–891, 2021. 30

[36] R.J. Lipton, D.J. Rose, and R.E. Tarjan. Generalized nested dissection. SIAM journal on numerical analysis, 16(2):346–358, 1979. 38

[37] R.J. Lipton and R.E. Tarjan. Applications of a planar separator theorem. SIAM journal on computing, 9(3):615–627, 1980. 38

[38] J. Löfberg. Yalmip: A toolbox for modeling and optimization in matlab. In Proceedings of the CACSD Conference, volume 3. Taipei, Taiwan, 2004. 34

[39] J. Löfberg. Pre- and post-processing sum-of-squares programs in practice. IEEE Transactions on Automatic Control, 54(5):1007–1011, 2009. 3, 10, 13

[40] B.F. Lourenço, M. Muramatsu, and T. Tsuchiya. Facial reduction and partial polyhedrality. SIAM J. Optim., 28(3):2304–2326, 2018. 17, 20

[41] G.L. Miller, S.-H. Teng, W. Thurston, and S.A. Vavasis. Geometric separators for finite-element meshes. SIAM Journal on Scientific Computing, 19(2):364–386, 1998. 38

[42] H. Mittelmann and J. Peng. Estimating bounds for quadratic assignment problems associated with hamming and manhattan distance matrices based on semidefinite programming. SIAM Journal on Optimization, 20(6):3408–3426, 2010. 33, 34

[43] K. Murota, Y. Kanno, M. Kojima, and S. Kojima. A numerical algorithm for block-diagonal decomposition of matrix *-algebras with application to semidefinite programming. Jpn. J. Ind. Appl. Math., 27(1):125–160, 2010. 6, 8

[44] D.E. Oliveira, H. Wolkowicz, and Y. Xu. ADMM for the SDP relaxation of the QAP. Math. Program. Comput., 10(4):631–658, 2018. 22, 30, 34, 35

[45] F. Permenter and Parrilo P.A. Dimension reduction for semidefinite programs via jordan algebras. Math. Program., 181(1):51–84, 2020. 11

[46] F.N. Permenter. Reduction methods in semidefinite and conic optimization. PhD thesis, Massachusetts Institute of Technology, 2017. 3, 5, 10, 12, 13
[47] H. Pirim, B. Ekşioğlu, A. D. Perkins, and Ç. Yüceer. Clustering of high throughput gene expression data. *Computers & operations research*, 39(12):3046–3061, 2012. 36

[48] T.K. Pong, H. Sun, N. Wang, and H. Wolkowicz. Eigenvalue, quadratic programming, and semidefinite programming relaxations for a cut minimization problem. *Comput. Optim. Appl.*, 63(2):333–364, 2016. 30, 38

[49] J. Povh and F. Rendl. A copositive programming approach to graph partitioning. *SIAM Journal on Optimization*, 18(1):223–241, 2007. 38

[50] J. Povh and F. Rendl. Copositive and semidefinite relaxations of the quadratic assignment problem. *Discrete Optim.*, 6(3):231–241, 2009. 26

[51] F. Rendl and R. Sotirov. The min-cut and vertex separator problem. *Computational Optimization and Applications*, 69(1):159–187, 2018. 38

[52] A. Schrijver. A comparison of the Delsarte and Lovász bounds. *IEEE Transactions on Information Theory*, 25(4):425–429, 1979. 3, 5

[53] R. Sotirov. An efficient semidefinite programming relaxation for the graph partition problem. *INFORMS Journal on Computing*, 26(1):16–30, 2013. 36

[54] R. Sotirov. Graph bisection revisited. *Ann. Oper. Res.*, 265(1):143–154, 2018. 36

[55] S. Sremac. *Error bounds and singularity degree in semidefinite programming*. PhD thesis, University of Waterloo, 2019. 3, 4, 5, 10, 12

[56] S. Sremac, H.J. Woerdeman, and H. Wolkowicz. Error bounds and singularity degree in semidefinite programming. *SIAM J. Optim.*, 31(1):812–836, 2021. 5, 12, 17, 40

[57] J.F Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625–653, 1999. 34

[58] J.F. Sturm. Error bounds for linear matrix inequalities. *SIAM J. Optim.*, 10(4):1228–1248 (electronic), 2000. 17

[59] D. Sun, K.C. Toh, Y. Yuan, and X.Y. Zhao. SDPNAL +: A Matlab software for semidefinite programming with bound constraints (version 1.0). *Optimization Methods and Software*, 35(1):87–115, 2020. 34, 35

[60] U. Truetsch. A semidefinite programming based branch-and-bound framework for the quadratic assignment problem. Tilburg University, 2014. 26, 27, 28

[61] L. Tungel. On the Slater condition for the SDP relaxations of nonconvex sets. *Oper. Res. Lett.*, 29(4):181–186, 2001. 15, 16, 19

[62] F. Vallentin. Symmetry in semidefinite programs. *Linear Algebra Appl.*, 430(1):360–369, 2009. 5

[63] E.R van Dam and R. Sotirov. Semidefinite programming and eigenvalue bounds for the graph partition problem. *Mathematical Programming*, 151(2):379–404, 2015. 36

[64] L. Vandenberghe and M.S. Andersen. Chordal graphs and semidefinite optimization. *Found. Trends Opt.*, 1(4):241–433, 2015. 3
[65] H. Waki and M. Muramatsu. A facial reduction algorithm for finding sparse SOS representations. *Oper. Res. Lett.*, 38(5):361–365, 2010. 3

[66] J.H.M. Wedderburn. On Hypercomplex Numbers. *Proc. London Math. Soc. (2)*, 6:77–118, 1908. 5, 8

[67] H. Wolkowicz and Q. Zhao. Semidefinite programming relaxations for the graph partitioning problem. *Discrete Appl. Math.*, 96/97:461–479, 1999. Selected for the special Editors’ Choice, Edition 1999. 36

[68] L. Yang, D. Sun, and K.-C. Toh. SDPNAL+: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Math. Program. Comput.*, 7(3):331–366, 2015. 34

[69] Q. Zhao, S.E. Karisch, F. Rendl, and H. Wolkowicz. Semidefinite programming relaxations for the quadratic assignment problem. *J. Comb. Optim.*, 2(1):71–109, 1998. Semidefinite Programming and Interior-point Approaches for Combinatorial Optimization Problems (Fields Institute, Toronto, ON, 1996). 25, 26