Partitioning through projections: strong SDP bounds for large graph partition problems

Frank de Meijer*†(iD)   Renata Sotirov*(iD)   Angelika Wiegele‡ §(iD)

Shudian Zhao‡§(iD)

Abstract

The graph partition problem (GPP) aims at clustering the vertex set of a graph into a fixed number of disjoint subsets of given sizes such that the sum of weights of edges joining different sets is minimized. This paper investigates the quality of doubly nonnegative (DNN) relaxations, i.e., relaxations having matrix variables that are both positive semidefinite and nonnegative, strengthened by additional polyhedral cuts for two variations of the GPP: the k-equipartition and the graph bisection problem. After reducing the size of the relaxations by facial reduction, we solve them by a cutting-plane algorithm that combines an augmented Lagrangian method with Dykstra's projection algorithm. Since many components of our algorithm are general, the algorithm is suitable for solving various DNN relaxations with a large number of cutting planes.

We are the first to show the power of DNN relaxations with additional cutting planes for the GPP on large benchmark instances up to 1,024 vertices. Computational results show impressive improvements in strengthened DNN bounds.

Keywords— graph partition problems, semidefinite programming, cutting planes, Dykstra's projection algorithm, augmented Lagrangian method

1 Introduction

The graph partition problem (GPP) is the problem of partitioning the vertex set of a graph into a fixed number of subsets, say k, of given sizes such that the sum of weights of edges joining different sets is minimized. In the case that all sets are of equal sizes we refer to the resulting GPP as the k-equipartition problem (k-EP). The case of the graph partition problem with $k = 2$ is known as the graph bisection problem (GBP). In the GBP the sizes of two subsets might differ. The special case of the GBP where both subsets have the same size is known in the literature as the equicut problem, see e.g. [33].

The graph partition problem is known to be NP-hard [24]. It is a fundamental problem that is extensively studied, mostly due to its applications in numerous fields, including VLSI design, social networks, floor planning, data mining, air traffic, image processing, image segmentation, parallel computing and telecommunication, see e.g., the book [6] and the references therein. Recent studies in quantum circuit design [28] also relate to the general graph partition problem. Furthermore, the GPP is used to compute bounds for the bandwidth problem [47]. For an overview of recent advances in graph partitioning, we refer the reader to [10].

There exist bounding approaches for the GPP that are valid for all variations of the GPP. We list some of them in this paragraph. Donath and Hoffman [15] derive an eigenvalue bound for

---

the graph partition problem. That bound is improved by Rendl and Wolkowicz [46]. Wolkowicz and Zhao [57] derive semidefinite programming (SDP) relaxations for the graph partition problem that are based on the so-called vector lifting approach. That is, relaxations in [57] have matrix variables of order $nk + 1$, where n is the number of vertices in the graph. An SDP relaxation based on the so-called matrix lifting approach is derived in [50], resulting in a compact relaxation for the GPP having matrix variables of order n. The relaxation from [50] is a doubly nonnegative (DNN) relaxation, which is an SDP relaxation over the set of nonnegative matrices. In general, vector lifting relaxations provide stronger bounds than matrix lifting relaxations, see e.g., [35, 50].

For the k-equipartition problem, Karisch and Rendl [32], among others, show how to reformulate the Donath-Hoffman and the Rendl-Wolkowicz relaxations as semidefinite programming problems. Karisch and Rendl also present several SDP relaxations with increasing complexity for the k-EP with matrix variables of order n that dominate relaxations from [15, 46]. The strongest SDP relaxation from [32] is a DNN relaxation with additional triangle and independent set inequalities. That relaxation provides the best known SDP bounds for the k-EP. However, it is difficult to compute those bounds for general graphs with more than 300 vertices when using interior point methods. In [14], the authors compute bounds from [32] for highly symmetric graphs by aggregating the triangle and independent set inequalities. Their approach results in significantly reduced computational effort for solving SDP relaxations with a large number of cutting planes for highly symmetric graphs. De Klerk et al. [34] exploit the fact that the k-EP is a special case of the quadratic assignment problem (QAP) to derive a DNN relaxation for the k-EP from a DNN relaxation for the QAP. The size of the resulting relaxation is reduced by exploiting symmetry. Numerical results show that the relaxation from [34] does not dominate the strongest relaxation from [32].

There exist several SDP relaxations for the graph bisection problem. Two equivalent SDP relaxations for the GBP whose matrix variables have order n are given in [33, 50]. The strongest matrix lifting DNN relaxation with a matrix variable of order n is derived in [51]. That relaxation is equivalent to the vector lifting DNN relaxation from [57], which has matrix variables of order $2n + 1$. For comparisons of the above mentioned SDP relaxations for the k-EP and the GBP, see e.g., [49] and [50], respectively.

Finally, we list below exact methods for solving the graph partition problem and its variants. Karisch et al. [33] develop a branch-and-bound algorithm that is based on a cutting-plane approach. In particular, the algorithm combines semidefinite programming and polyhedral constraints, i.e., triangle inequalities, to solve instances of the GBP. The algorithm from [33] solves problem instances with up to 90 vertices. Hager et al. [26] present a branch-and-bound algorithm for the graph bisection problem that exploits continuous quadratic programming formulations of the problem. The use of SDP bounds leads to the best performance of the algorithm from [26]. Armbruster et al. [4] evaluate the strength of a branch-and-cut framework on large and sparse instances, using linear and SDP relaxations of the GBP. The numerical results in [4] show that in the majority of the cases the semidefinite programming approach outperforms the linear one. Since prior to [4] it was widely believed that SDP relaxations are useful only for small and dense instances, the results of [4] influenced recent solver developments. As observed by [10], all mentioned exact approaches typically solve only very small problems with very large running times. The aim of this paper is to further investigate the quality of DNN relaxations on large graphs, that are strengthened by polyhedral cuts and solved by a first order method.

Main results and outline

Doubly nonnegative relaxations are known to provide superior bounds for various optimization problems. Although additional cutting planes further improve DNN relaxations, it is extremely difficult to compute the resulting bounds already for relaxations with matrix variables of order 300 via interior point methods. We design an efficient algorithm for computing DNN bounds with a huge number of additional cutting planes and show the power of the resulting bounds for two variations of the graph partition problem.

We conduct a study for the k-equipartition problem and the graph bisection problem. Although there exists a DNN relaxation for the GPP [57] that is suitable for both problems, we study the prob-

lems separately. Namely, the k-EP allows various equivalent SDP relaxations having different sizes of matrix variables due to the problem's invariance under permutations of the subsets. Since one can solve DNN relaxations with smaller matrix variables more efficiently than those with larger matrix variables, we consider the matrix lifting DNN relaxation for the k-EP from [32] that is strengthened by the triangle and independent set inequalities. On the other hand, the vector lifting DNN relaxation for the GBP from [57] is known to dominate matrix lifting DNN relaxations for the same problem. Therefore, we consider the vector lifting DNN relaxation for the GBP and strengthen it by adding boolean quadric polytope (BQP) inequalities.

Prior to solving the DNN relaxations, we use facial reduction to obtain equivalent smaller dimensional relaxations that are strictly feasible. The approach we use for the GBP is based on the dimension of the underlying polytope. Although strict feasibility of an SDP is not required for our solver, it makes the procedure more efficient.

To solve the DNN relaxations with additional polyhedral inequalities, we design a cutting-plane algorithm called the cutting-plane ADMM (CP–ADMM) algorithm. Our algorithm combines the Alternating Direction Method of Multipliers (ADMM) with Dykstra's projection algorithm. The ADMM exploits the natural splitting of the relaxations that arises from the facial reduction. Dykstra's cyclic projection algorithm finds projections onto polyhedra induced by the violated cuts. Since facial reduction eliminates redundant constraints and projects a relaxation onto a smaller dimensional space, the projections in the CP–ADMM are easier and faster. To further improve efficiency of the CP–ADMM, we cluster non-overlapping cuts, which allows us to perform the projections in each cluster simultaneously. Efficiency of the algorithm is also due to the exploitation of warm starts, as well as the use efficient separation routines. Since we present the various components of the CP–ADMM in a general way, the algorithm is suitable for solving various DNN relaxations incorporating additional cutting planes.

Our numerical results show that the CP–ADMM algorithm computes strong GPP bounds for graphs with up to 1,024 vertices by adding at most 50,000 cuts in less than two hours. Since our algorithm does not require lots of memory, we are able to compute strong bounds for even larger graphs than presented here. Numerical results also show that the additional cutting planes significantly improve the DNN relaxations, and that the resulting bounds can close gaps for instances with up to 500 vertices.

The paper is structured as follows. In Section 2 we introduce the graph partition problem. Section 2.1 presents DNN relaxations for the k-EP. In Section 2.2 we present DNN relaxations for the GBP and show how to apply facial reduction by exploiting the dimension of the bisection polytope. Our cutting-plane augmented Lagrangian algorithm is introduced in Section 3. The main ingredients of the algorithm are given in Section 3.1 and Section 3.2. In particular, Section 3.1 explains steps of the ADMM and Section 3.2 introduces Dykstra's projection algorithm and its semi-parallelized version. The CP–ADMM algorithm is outlined in Section 3.3. Section 4 considers various families of cutting planes that are used to strengthen DNN relaxations for the k-EP and GBP. Numerical results are given in Section 5.

Notation

The set of $n \times n$ real symmetric matrices is denoted by $\mathcal{S}^n$. The space of symmetric matrices is considered with the trace inner product, which for any $X, Y \in \mathcal{S}^n$ is defined as $\langle X, Y \rangle := \text{tr}(XY)$. The associated norm is the Frobenius norm $\|X\|_F := \sqrt{\text{tr}(XX)}$. The cone of symmetric positive semidefinite matrices of order n is defined as $\mathcal{S}_+^n := \{X \in \mathcal{S}^n : X \succeq \mathbf{0}\}$.

We use $[n]$ to denote the set of integers $\{1, \ldots, n\}$. The $\text{vec}(\cdot)$ operator maps an $n \times m$ matrix to a vector of length nm by stacking the columns of the matrix on top of one another. The Kronecker product $A \otimes B$ of matrices $A \in \mathbb{R}^{p \times q}$ and $B \in \mathbb{R}^{r \times s}$ is defined as the $pr \times qs$ matrix composed of pq blocks of size $r \times s$, with block ij given by $a_{ij}B$, $i \in [p]$, $j \in [q]$. We use the following properties of the Kronecker product and the vec operator:

$$\text{vec}(AXB) = (B^\top \otimes A)\text{vec}(X) \tag{1}$$

$$\mathrm{tr}(AB) = \mathrm{vec}(A^\top)^\top \mathrm{vec}(B). \tag{2}$$

The Hadamard product of two matrices $X = (x_{ij})$ and $Y = (y_{ij})$ of the same size is denoted by $X \circ Y$ and is defined as $(X \circ Y)_{ij} := x_{ij}y_{ij}$.

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

The operator $\mathrm{diag} \colon \mathbb{R}^{n \times n} \to \mathbb{R}^n$ maps a square matrix to a vector consisting of its diagonal elements. Its adjoint operator is denoted by $\mathrm{Diag} \colon \mathbb{R}^n \to \mathbb{R}^{n \times n}$. The rank of matrix X is denoted by $\mathrm{rank}(X)$.

2 The graph partition problem

Let $G = (V, E)$ be an undirected graph with vertex set V, $|V| = n$, and edge set E. Let $w \colon E \to \mathbb{R}$ be a weight function on the edges. Let k be a given integer such that $2 \le k \le n - 1$. The graph partition problem is to partition the vertex set of G into k disjoint sets $S_1, \ldots, S_k$ of specified sizes $m_1 \ge \cdots \ge m_k \ge 1$, $\sum_{j=1}^k m_j = n$ such that the total weight of edges joining different sets S_j is minimized. If $k = 2$, then we refer to the corresponding graph partition problem as the graph bisection problem. If $m_1 = \cdots = m_k = n/k$, then the resulting GPP is known as the k-equipartition problem.

Let A_w denote the weighted adjacency matrix of G with respect to w and $\mathbf{m} = (m_1, \ldots, m_k)^\top$. For a given partition of G into k subsets, let $P = (P_{ij}) \in \{0,1\}^{n \times k}$ be the partition matrix defined as follows:

$$P_{ij} := \begin{cases} 1, & i \in S_j \\ 0, & \text{otherwise} \end{cases} \qquad i \in [n], \ j \in [k]. \tag{3}$$

Thus, the j-th column of P is the characteristic vector of S_j. The total weight of the partition, i.e., the sum of weights of edges that join different sets equals

$$\frac{1}{2}\mathrm{tr}\left(A_w(\mathbf{J}_n - PP^\top)\right) = \frac{1}{2}\mathrm{tr}(LPP^\top),$$

where $L := \mathrm{Diag}(A_w\mathbf{1}_n) - A_w$ is the weighted Laplacian matrix of G. The GPP can be formulated as the following binary optimization problem:

$$\min_P \frac{1}{2}\langle L, PP^\top \rangle \tag{4a}$$

$$(GPP) \qquad \text{s.t. } P\mathbf{1}_k = \mathbf{1}_n, \tag{4b}$$

$$P^\top \mathbf{1}_n = \mathbf{m}, \tag{4c}$$

$$P_{ij} \in \{0,1\} \quad \forall i \in [n], \ j \in [k], \tag{4d}$$

where $P \in \mathbb{R}^{n \times k}$. Note that the objective function (4a) is quadratic. The constraints (4b) ensure that each vertex must be in exactly one subset. The cardinality constraints (4c) take care that the number of vertices in subset S_j is m_j for $j \in [k]$.

Let us briefly consider the polytope induced by all feasible partitions of G. Let $F_n^k(\mathbf{m})$ be the set of all characteristic vectors representing a partition of n vertices into k disjoint sets corresponding to the cardinalities in $\mathbf{m}$. In other words, $F_n^k(\mathbf{m})$ contains binary vectors of the form $\mathrm{vec}(P)$, where P is a partition matrix, see (3):

$$F_n^k(\mathbf{m}) = \left\{ x \in \{0,1\}^{kn} \ : \ \begin{pmatrix} \mathbf{I}_k \otimes \mathbf{1}_n^\top \\ \mathbf{1}_k^\top \otimes \mathbf{I}_n \end{pmatrix} x = \begin{pmatrix} \mathbf{m} \\ \mathbf{1}_n \end{pmatrix} \right\}. \tag{5}$$

We now define $\mathrm{Conv}(F_n^k(\mathbf{m}))$ as the *k-partition polytope*. Since the constraint matrix defining $F_n^k(\mathbf{m})$ is totally unimodular, this polytope can be explicitly written as follows:

$$\mathrm{Conv}(F_n^k(\mathbf{m})) = \left\{ x \in \mathbb{R}^{kn} \ : \ \begin{pmatrix} \mathbf{I}_k \otimes \mathbf{1}_n^\top \\ \mathbf{1}_k^\top \otimes \mathbf{I}_n \end{pmatrix} x = \begin{pmatrix} \mathbf{m} \\ \mathbf{1}_n \end{pmatrix}, \ x \geq \mathbf{0} \right\}. \tag{6}$$

The k-partition polytope can be seen as a special case of a transportation polytope, see e.g., [16]. We now derive the dimension of the k-partition polytope, which will be exploited in Section 2.2. The following result is implied by the dimension of the transportation polytope [16]. However, we add a proof for completeness.

Theorem 1. *The dimension of $\mathrm{Conv}(F_n^k(\mathbf{m}))$ equals $(k-1)(n-1)$.*

Proof. Let $B := \begin{pmatrix} \mathbf{I}_k \otimes \mathbf{1}_n^\top \\ \mathbf{1}_k^\top \otimes \mathbf{I}_n \end{pmatrix}$. Since for all $i \in [kn]$ there exists a partition such that $x_i = 1$, we know that $\dim(\mathrm{Conv}(F_n^k(\mathbf{m}))) = \dim(\mathrm{Nul}(B))$.

Let $b_1, \ldots, b_{kn}$ denote the columns of B. It is obvious that the set $\{b_1, \ldots, b_n, b_{n+1}, b_{2n+1}, \ldots, b_{(k-1)n+1}\}$ is linearly independent. From this it follows that $\mathrm{rank}(B) \geq n + k - 1$. Next, we define for all $l = 1, \ldots, k-1$ and $i = 2, \ldots, n$ a vector $w^{l,i} \in \mathbb{R}^{kn}$ as follows:

$$(w^{l,i})_j = \begin{cases} +1 & \text{if } j = 1 \text{ or } j = l \cdot n + i, \\ -1 & \text{if } j = i \text{ or } j = l \cdot n + 1, \\ 0 & \text{otherwise.} \end{cases}$$

One can verify that $Bw^{l,i} = \mathbf{0}$ for all $l = 1, \ldots, k-1$ and $i = 2, \ldots, n$. Moreover, since $w^{l,i}$ is the only vector that has a nonzero entry on position $ln + i$ among all defined vectors, the set $\{w^{l,i} \ : \ l = 1, \ldots, k-1, i = 2, \ldots, n\}$ is a linearly independent set. This proves that $\dim(\mathrm{Nul}(B)) \geq (k-1)(n-1)$.

Since $\mathrm{rank}(B) + \dim(\mathrm{Nul}(B)) = kn$, we conclude that $\dim(\mathrm{Nul}(B)) = (k-1)(n-1)$. $\qquad \square$

2.1 DNN relaxations for the k-equipartition problem

There exist several ways to obtain semidefinite programming relaxations for the k-EP. Namely, to obtain an SDP relaxation for the k-EP one can linearize the objective function of (GPP) by introducing a matrix variable of order n, which results in a matrix lifting relaxation, see e.g., [32, 50]. Another approach is to linearize the objective function by lifting the problem in the space of $(nk+1) \times (nk+1)$ matrices, which results in a vector lifting relaxation, see [57]. We call a DNN relaxation basic if it does not contain additional cutting planes such as triangle inequalities, etc. It is proven in [49] that the basic matrix and vector lifting DNN relaxations for the k-EP are equivalent. A more elegant proof of the same result can be found in Kuryatnikova et al. [35]. Since one can solve the basic matrix lifting relaxation from [32] more efficiently than the equivalent vector lifting relaxation from [57], we develop our algorithm for the matrix lifting relaxation for the k-EP.

To linearize the objective from (GPP) we replace $PP^\top$ by a matrix variable $Y \in \mathcal{S}^n$. From (4b) it follows that $Y_{ii} = \sum_{j=1}^k P_{ij}^2 = \sum_{j=1}^k P_{ij} = 1$ for all $i \in [n]$. From (4b)–(4c) we have $Y\mathbf{1}_n = PP^\top \mathbf{1}_n = \frac{n}{k} P\mathbf{1}_k = \frac{n}{k}\mathbf{1}_n$. After putting those constraints together, adding $Y \geq \mathbf{0}$ and $Y \succeq \mathbf{0}$, we arrive at the following DNN relaxation introduced by Karisch and Rendl [32]:

$$(DNN_{EP}) \qquad \begin{array}{ll} \min\limits_Y & \dfrac{1}{2}\langle L, Y \rangle \\[2mm] \text{s.t.} & \mathrm{diag}(Y) = \mathbf{1}_n, \\[2mm] & Y\mathbf{1}_n = \dfrac{n}{k}\mathbf{1}_n, \\[2mm] & Y \succeq \mathbf{0}, \quad Y \geq \mathbf{0}. \end{array} \tag{7}$$

We refer to (DNN_{EP}) as the basic matrix lifting relaxation. We show below that the nonnegativity constraints in (DNN_{EP}) are redundant for the equicut problem.

Lemma 1. *Let $k = 2$ and $Y \in \mathcal{S}_+^n$ be such that $diag(Y) = \mathbf{1}_n$ and $Y\mathbf{1}_n = \frac{n}{2}\mathbf{1}_n$. Then $Y \geq \mathbf{0}$.*

Proof. From $Y\mathbf{1}_n = \frac{n}{2}\mathbf{1}_n$ it follows that $\mathbf{1}_n$ is an eigenvector of Y corresponding to the eigenvalue $n/2$. Then, the eigenvalue decomposition of Y is

$$Y = \frac{1}{2}\mathbf{J}_n + \sum_{i=2}^{n} \lambda_i v_i v_i^\top,$$

where v_i is the eigenvector of Y corresponding to the eigenvalue λ_i for $i = 2, \ldots, n$. Moreover, eigenvectors v_i are orthogonal to $\mathbf{1}_n$. Thus $2Y - \mathbf{J}_n = 2\sum_{i=2}^{n} \lambda_i v_i v_i^\top \succeq \mathbf{0}$.

Now, let $Z := 2Y - \mathbf{J}_n$. From $diag(Y) = \mathbf{1}_n$ it follows that $diag(Z) = \mathbf{1}_n$. Since $Z \succeq \mathbf{0}$ we have that $-1 \leq Z_{ij} \leq 1$ for all $i, j \in [n]$, which implies that $Y_{ij} \geq 0$ for all $i, j \in [n]$. $\qquad\square$

For a different proof of Lemma 1 see e.g., Theorem 4.3 in [32]. The relaxation (DNN_{EP}) can be further strengthened by adding triangle and independent set inequalities, see Section 4.1 and 4.2, respectively. This strengthened relaxation is proposed in [32] and provides currently the strongest SDP bounds for the k-EP.

As proposed in [32], one can eliminate $Y\mathbf{1}_n = \frac{n}{k}\mathbf{1}_n$ in (7) and project the relaxation onto a smaller dimensional space, by exploiting the following result.

Lemma 2 ([32]). *Let $V \in \mathbb{R}^{n \times (n-1)}$ such that $V^\top \mathbf{1}_n = \mathbf{0}$ and $rank(V) = n - 1$. Then*

$$\left\{ Y \in \mathcal{S}^n : diag(Y) = \mathbf{1}_n, Y\mathbf{1}_n = \frac{n}{k}\mathbf{1}_n \right\} = \left\{ \frac{1}{k}\mathbf{J}_n + VRV^\top : R \in \mathcal{S}^{n-1}, \; diag(VRV^\top) = \frac{k-1}{k}\mathbf{1}_n \right\}.$$

The matrix V in Lemma 2 can be any matrix which columns form a basis for $\mathbf{1}_n^\perp$, e.g.,

$$V = \begin{pmatrix} \mathbf{I}_{n-1} \\ -\mathbf{1}_{n-1}^\top \end{pmatrix} \tag{8}$$

We use the result of Lemma 2 and replace Y by $\frac{1}{k}\mathbf{J}_n + VRV^\top$ in (DNN_{EP}), which leads to the following equivalent relaxation:

$$\min_R \quad \langle L_{EP}, VRV^\top \rangle$$
$$\text{s.t. } diag(VRV^\top) = \frac{k-1}{k}\mathbf{1}_n, \tag{9}$$
$$VRV^\top \geq -\frac{1}{k}\mathbf{J}_n, \quad R \succeq \mathbf{0},$$

were $R \in \mathcal{S}_+^{n-1}$. Here, we exploit $\langle L, \mathbf{J}_n \rangle = 0$ to rewrite the objective, and define

$$L_{EP} := \frac{1}{2}L. \tag{10}$$

It is not difficult to verify that the matrix

$$\hat{R} = \frac{n(k-1)}{k(n-1)}\mathbf{I}_{n-1} - \frac{(k-1)}{k(n-1)}\mathbf{J}_{n-1}$$

is feasible for (9), see also [32]. The matrix $\hat{R}$ has two distinct eigenvalues, namely $\frac{n(k-1)}{k(n-1)}$ with multiplicity $n - 2$ and $\frac{(k-1)}{k(n-1)}$ with multiplicity one. This implies that $\hat{R} \succ 0$. Also,

$$\frac{1}{k}\mathbf{J}_n + V\hat{R}V^\top = \frac{n(k-1)}{k(n-1)}\mathbf{I}_n + \frac{(n-k)}{k(n-1)}\mathbf{J}_n > \mathbf{0},$$

and thus $V\hat{R}V^\top > -\frac{1}{k}\mathbf{J}_n$. This shows that $\hat{R}$ is a Slater feasible point of (9).

6

For future reference, we define the following sets:

$$\mathcal{R}_{EP} := \left\{ R \in \mathcal{S}^{n-1} : R \succeq \mathbf{0} \right\}, \tag{11}$$

$$\mathcal{X}_{EP} := \left\{ X \in \mathcal{S}^n : \operatorname{diag}(X) = \frac{k-1}{k}\mathbf{1}_n, \ -\frac{1}{k}\mathbf{J}_n \leq X \leq \frac{k-1}{k}\mathbf{J}_n \right\}. \tag{12}$$

Now, we rewrite the DNN relaxation (9) as follows:

$$\min \left\{ \langle L_{EP}, X \rangle : \ X = VRV^\top, \ R \in \mathcal{R}_{EP}, \ X \in \mathcal{X}_{EP} \right\}. \tag{13}$$

Note that $\mathcal{X}_{EP}$ also contains constraints that are redundant for (9). These constraints speed up the convergence of our algorithm, as explained in Section 3.1. In the same section, it becomes clear that the inclusion of redundant constraints should not complicate the structure of $\mathcal{X}_{EP}$ too much. Whether or not to include a redundant constraint, is determined by an empirical trade-off between these measures.

2.2 DNN relaxations for the bisection problem

For the graph bisection problem there exist both vector and matrix lifting SDP relaxations. The matrix lifting relaxations derived in [33, 50] are equivalent and have matrix variables of order n. A vector lifting SDP relaxation for the GBP is derived by Wolkowicz and Zhao [57] and has a matrix variable of order $2n + 1$. The DNN relaxation from [57] dominates the basic matrix lifting DNN relaxations, i.e., DNN relaxations without additional cutting planes, see [50] for a proof. In [51] a matrix lifting DNN relaxation with additional cutting planes is derived for the GBP that is equivalent to the DNN relaxation from [57]. Although the relaxation from [51] has a matrix variable of order n, we work with the vector lifting DNN relaxation because it has a more appropriate structure for our ADMM approach.

In this section we present the vector lifting DNN relaxation from [57] and show how to obtain its facially reduced equivalent version by using properties of the bisection polytope. As a byproduct, we also study properties of the feasible set of the DNN relaxation, see Theorem 2.

Let $\mathbf{m} = (m_1, m_2)^\top$ such that $m_1 + m_2 = n$ be given. To derive a vector lifting SDP relaxation for the GBP we linearize the objective from (GPP) by lifting variables into $\mathcal{S}^{2n+1}$. In particular, let $P \in \{0, 1\}^{n \times 2}$ be a partition matrix and $x = \operatorname{vec}(P)$. We use (1) and (2) to rewrite the objective as follows:

$$\operatorname{tr}(LPP^\top) = \operatorname{vec}(P)^\top (\mathbf{I}_2 \otimes L)\operatorname{vec}(P) = x^\top (\mathbf{I}_2 \otimes L)x = \langle \mathbf{I}_2 \otimes L, xx^\top \rangle.$$

Now, we replace $xx^\top$ by a matrix variable $\hat{X} \in \mathcal{S}^{2n}$. The constraint $\hat{X} = xx^\top$ can be weakened to $\hat{X} - xx^\top \succeq \mathbf{0}$, which is equivalent to $X := \begin{pmatrix} 1 & x^\top \\ x & \hat{X} \end{pmatrix} \succeq \mathbf{0}$ by the well-known Schur Complement lemma.

In the sequel, we use the following block notation for matrices in $\mathcal{S}^{2n+1}$:

$$X = \begin{pmatrix} 1 & (x^1)^\top & (x^2)^\top \\ x^1 & X^{11} & X^{12} \\ x^2 & X^{21} & X^{22} \end{pmatrix},$$

where x^1 (resp., x^2) corresponds to the first (resp., second) column in P, and X^{ij} corresponds to $x^i(x^j)^\top$ for $i, j = 1, 2$.

Now, from $\mathbf{1}_n^\top x^i = m_i$ $(i = 1, 2)$ it follows that $\operatorname{tr}(X^{ii}) = m_i$, $\operatorname{tr}(\mathbf{J}_n X^{ii}) = m_i^2$ and $\operatorname{tr}(\mathbf{J}_n(X^{12} + X^{21})) = 2m_1 m_2$. From $x^1 \circ x^2 = \mathbf{0}$ it follows that $\operatorname{diag}(X^{12}) = \mathbf{0}$.

The above derivation results in the following vector lifting SDP relaxation for the GBP [57]:

$$\min_X \quad \frac{1}{2}\langle L, X^{11} + X^{22}\rangle$$

$$\text{s.t.} \quad \text{tr}(X^{ii}) = m_i, \ \text{tr}\left(\mathbf{J}_n X^{ii}\right) = m_i^2, \ \ i = 1, 2,$$

$$(SDP_{BP}) \qquad \text{diag}(X^{12}) = \mathbf{0}, \ \text{diag}(X^{21}) = \mathbf{0}, \ \text{tr}\left(\mathbf{J}_n(X^{12} + X^{21})\right) = 2m_1m_2, \qquad (14)$$

$$X = \begin{pmatrix} 1 & (x^1)^\top & (x^2)^\top \\ x^1 & X^{11} & X^{12} \\ x^2 & X^{21} & X^{22} \end{pmatrix} \succeq \mathbf{0}, \ \ x^i = \text{diag}(X^{ii}), \ \ i = 1, 2,$$

where $X \in \mathcal{S}^{2n+1}$. By imposing nonnegativity constraints on the matrix variable in (SDP_{BP}), we obtain the following DNN relaxation:

$$(DNN_{BP}) \qquad (SDP_{BP}) \ \& \ X \geq \mathbf{0}. \qquad (15)$$

The relaxation (DNN_{BP}) can be further strengthened by additional cutting planes. We propose adding the boolean quadric polytope inequalities, see Section 4.3.

The zero pattern on off-diagonal blocks in (14) can be written using a linear operator $\mathcal{G}_{\mathcal{J}}(\cdot)$, known as the Gangster operator, see [57]. The operator $\mathcal{G}_{\mathcal{J}} \colon \mathcal{S}^{2n+1} \to \mathcal{S}^{2n+1}$ is defined as

$$\mathcal{G}_{\mathcal{J}}(X) = \begin{cases} X_{ij} & \text{if } (i,j) \in \mathcal{J}, \\ 0 & \text{otherwise}, \end{cases}$$

where

$$\mathcal{J} = \{(i,j) \ : \ i = (p-1)n + q + 1, \ j = (r-1)n + q + 1, \ q \in [n], \ p, r \in \{1, 2\}, \ p \neq r\}. \qquad (16)$$

The constraints $\text{diag}(X^{12}) = \text{diag}(X^{21}) = \mathbf{0}$ are given by $\mathcal{G}_J(X) = \mathbf{0}$.

We now show how to project the SDP relaxation (14) onto a smaller dimensional space in order to obtain an equivalent strictly feasible relaxation by facial reduction. Although such reduction is performed for the general graph partitioning problem in [57], our approach differs by relying on the polytope of all bisections. We first apply facial reduction to the relaxation (SDP_{BP}), after which we derive the facially reduced equivalent of (DNN_{BP}).

We start by deriving two properties that hold for all feasible solutions of (SDP_{BP}).

Theorem 2. *Let* $X = \begin{pmatrix} 1 & (x^1)^\top & (x^2)^\top \\ x^1 & X^{11} & X^{12} \\ x^2 & X^{21} & X^{22} \end{pmatrix}$ *with* $\hat{X} = \begin{pmatrix} X^{11} & X^{12} \\ X^{21} & X^{22} \end{pmatrix}$ *and* $x = \begin{pmatrix} x^1 \\ x^2 \end{pmatrix}$ *be feasible for* (SDP_{BP}). *Then,*

(i) $a_i^\top \left(\hat{X} - xx^\top\right) a_i = 0$ *where* $a_i = \mathbf{u}_i \otimes \mathbf{1}_n$, $\mathbf{u}_i \in \mathbb{R}^2$, $i \in [2]$;

(ii) $b_i^\top \left(\hat{X} - xx^\top\right) b_i = 0$ *where* $b_i = \mathbf{1}_2 \otimes \mathbf{u}_i$, $\mathbf{u}_i \in \mathbb{R}^n$, $i \in [n]$.

Proof. (i) Without loss of generality we take $i = 1$. Then $a_1 = \mathbf{u}_1 \otimes \mathbf{1}_n$, which yields

$$a_1^\top \left(\hat{X} - xx^\top\right) a_1 = \mathbf{1}_n^\top X^{11} \mathbf{1}_n - \mathbf{1}_n^\top x^1 (x^1)^\top \mathbf{1}_n = \text{tr}(\mathbf{J}_n X^{11}) - \text{tr}(X^{11})^2 = m_1^2 - m_1^2 = 0,$$

using the constraints of (14). The proof for $i = 2$ is similar.

(ii) First, we show that any feasible solution to (14) satisfies $\text{diag}(X^{11}) + \text{diag}(X^{22}) = \mathbf{1}_n$. For all $i \in [n]$ we define $v^i \in \mathbb{R}^{2n}$ as

$$\left(v^i\right)_j := \begin{cases} -1 & \text{if } j = i \text{ or } j = n + i, \\ 0 & \text{otherwise}. \end{cases}$$

8

From $\hat{X} - xx^\top \succeq \mathbf{0}$, we have

$$\begin{pmatrix} 1 \\ v^i \end{pmatrix}^\top \begin{pmatrix} 1 & x^\top \\ x & \hat{X} \end{pmatrix} \begin{pmatrix} 1 \\ v^i \end{pmatrix} \geq 0 \quad \Longleftrightarrow \quad 1 - X_{ii}^{11} - X_{ii}^{22} \geq 0 \quad \Longleftrightarrow \quad X_{ii}^{11} + X_{ii}^{22} \leq 1,$$

where we used the fact that $\operatorname{diag}(\hat{X}) = x$. Since

$$n = m_1 + m_2 = \operatorname{tr}(X^{11}) + \operatorname{tr}(X^{22}) = \sum_{i=1}^n \left(X_{ii}^{11} + X_{ii}^{22} \right),$$

and the latter summation consists of n elements bounded above by 1, we must have $X_{ii}^{11} + X_{ii}^{22} = 1$ for all $i \in [n]$.

Now, for $b_i = \mathbf{1}_2 \otimes \mathbf{u}_i$, $i \in [n]$ we have

$$b_i^\top \left(\hat{X} - xx^\top \right) b_i = X_{ii}^{11} + X_{ii}^{12} + X_{ii}^{21} + X_{ii}^{22} - \left(X_{ii}^{11} + X_{ii}^{22} \right)^2.$$

Since $\operatorname{diag}(X^{12}) = \operatorname{diag}(X^{21}) = \mathbf{0}$ and $\operatorname{diag}(X^{11}) + \operatorname{diag}(X^{22}) = \mathbf{1}_n$, it follows that $b_i^\top \left(\hat{X} - xx^\top \right) b_i = 1 - 1^2 = 0$ for all $i \in [n]$. $\qquad \square$

We can exploit the properties stated in Theorem 2 to identify vectors in the null space of all feasible solutions of (SDP_{BP}). In order to do so, we use the following result.

Lemma 3 ([45]). *Let $X \in \mathcal{S}^l$, $x \in \mathbb{R}^l$ and $a \in \mathbb{R}^l$ be such that $X - xx^\top \succeq \mathbf{0}$, $a^\top x = t$ for some $t \in \mathbb{R}$, and $a^\top \left(X - xx^\top \right) a = 0$. Then $[-t, a^\top]^\top$ is an eigenvector of $\begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix}$ with respect to eigenvalue 0.*

It follows from the constraints of (14) that $a_i^\top x = m_i$ for $i \in [2]$ and $b_i^\top x = 1$ for $i \in [n]$, where a_i and b_i are defined as in Theorem 2. As a result, Theorem 2 and Lemma 3 imply that

$$\begin{pmatrix} -m_i \\ \mathbf{u}_i \otimes \mathbf{1}_n \end{pmatrix}, i \in [2], \quad \text{and} \quad \begin{pmatrix} -1 \\ \mathbf{1}_2 \otimes \mathbf{u}_i \end{pmatrix}, i \in [n],$$

are eigenvectors of $\begin{pmatrix} 1 & x^\top \\ x & \hat{X} \end{pmatrix}$ with respect to eigenvalue 0. Now, let us define the matrix $T \in \mathbb{R}^{(n+2) \times (2n+1)}$ as follows:

$$T := \begin{pmatrix} -\mathbf{m} & \mathbf{I}_2 \otimes \mathbf{1}_n^\top \\ -\mathbf{1}_n & \mathbf{1}_2^\top \otimes \mathbf{I}_n \end{pmatrix}.$$

Moreover, let $\mathcal{V} = \operatorname{Nul}(T)$. Any $a \in \mathcal{V}$ defines an element $aa^\top$ exposing the feasible set of (SDP_{BP}). It follows from the facial geometry of the cone of positive semidefinite matrices that the feasible set of (SDP_{BP}) is contained in

$$S_{\mathcal{V}} := \left\{ X \in \mathcal{S}_+^{2n+1} \ : \ \operatorname{Col}(X) \subseteq \mathcal{V} \right\},$$

which is a face of $\mathcal{S}_+^{2n+1}$. It remains to prove that this is actually the minimal face of $\mathcal{S}_+^{2n+1}$ containing the feasible set of (SDP_{BP}). For that purpose, we consider the underlying bisection polytope $\operatorname{Conv}(F_n^2(\mathbf{m}))$, see (6). It follows from Theorem 1 that $\dim(\operatorname{Conv}(F_n^2(\mathbf{m}))) = n - 1$. Besides, observe that T is constructed as the constraint matrix defining $\operatorname{Conv}(F_n^2(\mathbf{m}))$ augmented with an additional column. Since this additional column does not increase its rank, we have $\operatorname{rank}(T) = n + 1$, which implies that $\dim(\mathcal{V}) = n$. Let $V \in \mathbb{R}^{(2n+1) \times n}$ be a matrix whose columns form a basis for $\mathcal{V}$. Then the face $S_{\mathcal{V}}$ can be equivalently written as:

$$S_{\mathcal{V}} = V \mathcal{S}_+^n V^\top. \tag{17}$$

To show that $S_{\mathcal{V}}$ is the minimal face containing the feasible set of (SDP_{BP}) we apply a result by Tunçel [54].

Theorem 3 ([54]). *Given $F \subseteq \mathbb{R}^l$, let $\mathcal{F} := \left\{ \begin{pmatrix} 1 & x^\top \\ x & \hat{X} \end{pmatrix} \in \mathcal{S}_+^{l+1} : \mathcal{A}\left(\begin{pmatrix} 1 & x^\top \\ x & \hat{X} \end{pmatrix} \right) = \mathbf{0} \right\}$, with $\mathcal{A} : \mathcal{S}^{l+1} \to \mathbb{R}^p$ a linear transformation, be a relaxation of the lifted polyhedron*

$$\operatorname{Conv}\left\{ \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}^\top : x \in F \right\}.$$

Suppose that $\mathcal{F} \subseteq V\mathcal{S}_+^d V^\top$ for some full rank matrix $V \in \mathbb{R}^{(l+1)\times d}$. If $\dim(\operatorname{Conv}(F)) = d - 1$, then there exists some $R \succ \mathbf{0}$ such that $VRV^\top \in \mathcal{F}$.

Based on Theorem 3, we can now show the minimality of the face $S_\mathcal{V}$ for both (SDP_{BP}) and (DNN_{BP}).

Theorem 4. *The set $S_\mathcal{V}$ is the minimal face of $\mathcal{S}_+^{2n+1}$ containing the feasible set of (SDP_{BP}). If $m_1, m_2 \geq 2$, then $S_\mathcal{V}$ is also the minimal face of $\mathcal{S}_+^{2n+1}$ containing the feasible set of (DNN_{BP}).*

Proof. The feasible region of (SDP_{BP}) can be written in the form of $\mathcal{F}$ in the statement of Theorem 3. To show minimality for (SDP_{BP}), it suffices to show that there exists a matrix $R \in \mathcal{S}_+^n$, $R \succ \mathbf{0}$ such that $VRV^\top$ is feasible for (SDP_{BP}). Since $\dim(\operatorname{Conv}(F_n^2(\mathbf{m}))) = n - 1$, it immediately follows from Theorem 3 that such matrix, say R_1, exists.

To prove the second statement, it suffices to show that there exists an $R \in \mathcal{S}_+^n$ such that $R \succ \mathbf{0}$ and $(VRV^\top)_{ij} > 0$ for all $(i, j) \in \mathcal{J}_C$, where $\mathcal{J}_C = ([2n + 1] \times [2n + 1]) \setminus \mathcal{J}$, see (16). Since $m_1, m_2 \geq 2$, it follows that for any $(i, j) \in \mathcal{J}_C$ there exists a bisection x^{ij} such that

$$\left(\begin{pmatrix} 1 \\ x^{ij} \end{pmatrix} \begin{pmatrix} 1 \\ x^{ij} \end{pmatrix}^\top \right)_{ij} > 0.$$

Let $R^{ij} \in \mathcal{S}_+^n$ denote the matrix such that $VR^{ij}V^\top = \begin{pmatrix} 1 \\ x^{ij} \end{pmatrix} \begin{pmatrix} 1 \\ x^{ij} \end{pmatrix}^\top$, which consists by construction of $S_\mathcal{V}$. Now, let R_2 be any positive convex combination of the elements in $\{R^{ij} : (i, j) \in \mathcal{J}_C\}$. By construction, $R_2 \succeq \mathbf{0}$, while $(VR_2V^\top)_{ij} > 0$ for all $(i, j) \in \mathcal{J}_C$. Finally, any positive convex combination of R_1 and R_2 provides a matrix R with the desired properties. $\square$

The result of Theorem 4 can be exploited to derive strictly feasible equivalent versions of (SDP_{BP}) and (DNN_{BP}). We focus here only on the DNN relaxation (DNN_{BP}). Theorem 4 allows us to replace X by $VRV^\top$ in (DNN_{BP}). One can take the following matrix for V:

$$V = \begin{pmatrix} 1 & \mathbf{0} \\ \frac{1}{n}\mathbf{m} \otimes \mathbf{1}_n & V_2 \otimes V_n \end{pmatrix}, \quad \text{where } V_p = \begin{pmatrix} \mathbf{I}_{p-1} \\ -\mathbf{1}_{p-1}^\top \end{pmatrix} \text{ for } p = 2, n. \tag{18}$$

Because of the structure of V, most of the constraints in (DNN_{BP}) become redundant. One can easily verify that the resulting relaxation in lower dimensional space is as follows, see e.g., [57]:

$$\begin{aligned} \min_R \quad & \operatorname{tr}(L_{BP}VRV^\top) \\ \text{s.t.} \quad & \mathcal{G}_\mathcal{J}(VRV^\top) = \mathbf{0}, \\ & (VRV^\top)_{1,1} = 1, \\ & VRV^\top \geq \mathbf{0}, \ R \in \mathcal{S}_+^n, \end{aligned} \tag{19}$$

where

$$L_{BP} := \frac{1}{2} \begin{pmatrix} 0 & \mathbf{0}^\top \\ \mathbf{0} & \mathbf{I}_2 \otimes L \end{pmatrix}, \tag{20}$$

and L is the weighted Laplacian matrix of G. Let us now define the following sets:

$$\mathcal{R}_{BP} := \{R \in \mathcal{S}^n : R \succeq \mathbf{0}\}, \tag{21}$$

$$\mathcal{X}_{BP} := \left\{ X \in \mathcal{S}^{2n+1} : \begin{array}{l} \mathcal{G}_{\mathcal{J}}(X) = \mathbf{0}, \ X_{1,1} = 1, \ \mathrm{tr}(X^{ii}) = m_i, \ i \in [2], \\ \mathrm{diag}(X^{11}) + \mathrm{diag}(X^{22}) = \mathbf{1}_n, \ X\mathbf{u}_1 = \mathrm{diag}(X), \\ \mathbf{0} \le X \le \mathbf{J} \end{array} \right\}. \tag{22}$$

Now, we are ready to rewrite the facially reduced DNN relaxation (19) as follows:

$$\min \left\{ \langle L_{BP}, X \rangle : \ X = VRV^\top, \ R \in \mathcal{R}_{BP}, \ X \in \mathcal{X}_{BP} \right\}. \tag{23}$$

Note that $\mathcal{X}_{BP}$ also contains constraints that are redundant for (19).

3   A cutting plane augmented Lagrangian algorithm

SDP has proven effective for modeling optimization problems and providing strong bounds. It is well-known that SDP solvers based on interior point methods might have considerable memory demands already for medium-scale problems. Recently, promising alternatives for solving large-scale SDP relaxations have been investigated. We refer the interested reader to [11, 38, 44, 53, 55, 58] for algorithms based on alternating direction augmented Lagrangian methods for solving SDPs. For efficient approaches to solving DNN relaxations, see also e.g., [29, 30, 36, 42, 56, 59]. To the best of our knowledge only [40] incorporates an augmented Lagrangian method into a cutting-plane framework. The authors of [40] consider only one type of cutting planes. Here, we incorporate various types of cutting planes into one framework and use a more efficient version of the ADMM than the one used in [40].

In Section 3.1 we describe variants of the ADMM that are used within our cutting-plane algorithm. Section 3.2 presents Dykstra's cyclic projection algorithm that is used for projections onto polyhedra induced by the violated cuts. Section 3.3 presents our cutting plane augmented Lagrangian algorithm.

3.1   The Alternating Direction Method of Multipliers

The ADMM is a first-order method from the 1970s that is developed for solving convex optimization problems. This method decomposes an optimization problem into several subproblems that are easier to solve than the original problem. There exist several variants of the ADMM for solving SDPs. We consider here a variant of the ADMM that resembles variants from [29, 42], where we additionally consider an adaptive stepsize term proposed by Lorenz and Tran-Dinh [37] when solving the k-EP.

In order to describe the ADMM scheme for solving SDP relaxations for both problems, the k-equipartition problem (13) and the graph bisection problem (23), we introduce the following unified notation: For the k-EP, define $\bar{L} := L_{EP}$, $\mathcal{R} := \mathcal{R}_{EP}$ and $\mathcal{X} := \mathcal{X}_{EP}$ (see resp., (10), (11), (12)), and for the GBP define $\bar{L} := L_{BP}$, $\mathcal{R} := \mathcal{R}_{BP}$ and $\mathcal{X} := \mathcal{X}_{BP}$ (see resp., (20), (21), (22)).

Let Z denote the Lagrange multiplier for the constraint $X = VRV^\top$. Then, the augmented Lagrangian function of (13) and (23) w.r.t. the constraint $X = VRV^\top$ for a penalty parameter σ is as follows:

$$\mathcal{L}_\sigma(X, R, Z) = \langle \bar{L}, X \rangle + \langle Z, X - VRV^\top \rangle + \frac{\sigma}{2} \|X - VRV^\top\|_F^2. \tag{24}$$

In each iteration, the ADMM minimizes $\mathcal{L}_\sigma(X, R, Z)$ subject to $X \in \mathcal{X}$ and $R \in \mathcal{R}$ and updates Z via a stepsize update. The ADMM update scheme requires a matrix V that has orthonormal columns that can be obtained by applying a QR-decomposition to (8) for the k-EP and to (18) for the GBP.

Let (R^p, X^p, Z^p) denote the p-th iterate of the ADMM. The next iterate $(R^{p+1}, X^{p+1}, Z^{p+1})$ is obtained as follows:

$$R^{p+1} = \underset{R \in \mathcal{R}}{\arg\min} \ \mathcal{L}_{\sigma^p}(R, X^p, Z^p), \tag{25a}$$

$$X^{p+1} = \underset{X \in \mathcal{X}}{\arg\min} \ \mathcal{L}_{\sigma^p}(R^{p+1}, X, Z^p), \tag{25b}$$

$$Z^{p+1} = Z^p + \gamma \cdot \sigma^p \cdot (X^{p+1} - VR^{p+1}V^\top), \tag{25c}$$

where $\gamma \in \left(0, \frac{1+\sqrt{5}}{2}\right)$ is a parameter for updating the dual multiplier Z^p, see e.g., [55].

There exist different ways for dealing with the stepsize term $\gamma \cdot \sigma^p$. One possibility is to keep σ^p and γ fixed during the algorithm. In this approach, σ^p depends on the problem data and γ has a value larger than one. This is known in the literature as the ADMM with larger stepsize, as originally proposed by [21]. An alternative is the ADMM with adaptive stepsize term as introduced in [37]. In that case $\gamma = 1$ and the parameter σ^p is updated as follows:

$$\sigma^{p+1} := (1 - \omega^{p+1})\sigma^p + \omega^{p+1}\mathcal{P}_{[\sigma_{\min},\sigma_{\max}]}\frac{\|Z^{p+1}\|_F}{\|X^{p+1}\|_F}, \tag{26}$$

where $\omega^{p+1} := 2^{-p/100}$ is the weight, $\sigma_{\min}$ and $\sigma_{\max}$ are the box bounds for σ^p, and $\mathcal{P}_{[\sigma_{\min},\sigma_{\max}]}$ is the projection onto $[\sigma_{\min}, \sigma_{\max}]$.

Recall that we added redundant constraints for the SDP relaxations (13) and (23) to the set $\mathcal{X}$. Those constraints are, though, not redundant in the subproblem (25b). They are included to speed up the convergence of the ADMM algorithm in practice, see e.g., [29, 42, 40].

One can solve the R-subproblem (25a) as follows:

$$R^{p+1} = \arg\min_{R \in \mathcal{R}} \mathcal{L}_{\sigma^p}(R, X^p, Z^p) = \arg\min_{R \in \mathcal{R}} \langle Z^p, -VRV^\top \rangle + \frac{\sigma^p}{2}\left\|X^p - VRV^\top\right\|_F^2,$$

$$= \arg\min_{R \in \mathcal{R}} \left\|V^\top\left(X^p + \frac{1}{\sigma^p}Z^p\right)V - R\right\|_F^2 = \mathcal{P}_{\succeq \mathbf{0}}\left(V^\top\left(X^p + \frac{1}{\sigma^p}Z^p\right)V\right),$$

where $\mathcal{P}_{\succeq \mathbf{0}}(\cdot)$ denotes the orthogonal projection onto the cone of positive semidefinite matrices.

The X-subproblem (25b) can be solved as follows:

$$X^{p+1} = \arg\min_{X \in \mathcal{X}} \mathcal{L}_{\sigma^p}(R^{p+1}, X, Z^p) = \arg\min_{X \in \mathcal{X}} \langle \bar{L} + Z^p, X \rangle + \frac{\sigma^p}{2}\left\|X - VR^{p+1}V^\top\right\|_F^2,$$

$$= \arg\min_{X \in \mathcal{X}} \left\|X - \left(VR^{p+1}V^\top - \frac{1}{\sigma^p}\left(\bar{L} + Z^p\right)\right)\right\|_F^2 = \mathcal{P}_{\mathcal{X}}\left(VR^{p+1}V^\top - \frac{1}{\sigma^p}\left(\bar{L} + Z^p\right)\right),$$

where $\mathcal{P}_{\mathcal{X}}(\cdot)$ denotes the orthogonal projection onto the polyhedral set $\mathcal{X}$. In Appendix A we show how this projection can be performed explicitly.

The performance of the ADMM greatly depends on the stepsize term. Our preliminary tests show that for the k-EP the updating scheme (25)–(26) with adaptive stepsize term outperforms the ADMM with larger stepsize. That is, our adaptive ADMM performs better than the ADMM variants from [29, 30, 42]. Moreover, preliminary results show that it outperforms the algorithm from [36]. For the GBP, however, our preliminary tests show that it is more beneficial to keep σ^p fixed and use larger γ. The resulting version of the ADMM resembles versions from [29, 30, 42]. Consequently, we initialize the ADMM algorithm (25) for the k-EP by

$$R^0 = \mathbf{0}, \qquad X^0 = \frac{k-1}{k}\mathbf{I}_n, \qquad Z^0 = \mathbf{0}, \qquad \sigma^0 = \left\lceil\frac{n}{k}\right\rceil, \qquad \gamma = 1, \tag{27}$$

and for the GBP we set

$$R^0 = \mathbf{0}, \qquad X^0 = \mathbf{u}_1\mathbf{u}_1^\top, \qquad Z^0 = \mathbf{0}, \qquad \sigma^0 = \left\lceil\left(\frac{2n}{m_1}\right)^2\right\rceil, \qquad \gamma = 1.608. \tag{28}$$

3.2 Clustered Dykstra's projection algorithm

Both DNN problems, (13) and (23), can be strengthened by adding valid cutting planes, see Section 4. Since these cutting planes are polyhedral, it is natural to include additional cuts to the set $\mathcal{X}$.

This addition will, however, spoil the easy structure of $\mathcal{X}$. As a result, finding the explicit projection onto this new polyhedral set becomes a difficult task, even after the addition of a single cut. In [40] this issue is resolved by splitting the polyhedral set into subsets and using iterative projections based on Dykstra's algorithm [8, 17]. This algorithm finds the projection onto the intersection of a finite number of polyhedral sets, assuming that the projection onto each of the separate sets is known. Although there exist many algorithms for finding such projection in the literature, the recent study of [5] shows superior behaviour of Dykstra's cyclic projection algorithm.

In this section we briefly present Dykstra's algorithm and show how to implement it efficiently by clustering non-overlapping cuts. Similar to the previous section, we present a generic version of the algorithm that can be applied to both the k-EP and the GBP.

Let us assume that $\mathcal{T}$ is an index set of cutting planes on the primal variable X in the ADMM scheme, i.e., every $t \in \mathcal{T}$ corresponds to a single cut. Also, for each $t \in \mathcal{T}$ let $\mathcal{H}_t$ be a polyhedron that is induced by the cut t. One can think of $\mathcal{H}_t$ as the halfspace induced by the cut t, where additional constraints are added as long as the projection onto $\mathcal{H}_t$ remains efficient. In Section 4 we show how the set $\mathcal{T}$ and the polyhedra $\mathcal{H}_t$ look like for cuts related to both the k-EP and the GBP, and present the projectors onto the sets $\mathcal{H}_t$.

When adding the cuts in $\mathcal{T}$ to the relaxation, the polyhedral set $\mathcal{X}$ has to be replaced by

$$\mathcal{X}_\mathcal{T} := \mathcal{X} \cap \left(\bigcap_{t \in \mathcal{T}} \mathcal{H}_t \right). \tag{29}$$

The X-subproblem of the ADMM scheme (25b) asks for the projection onto $\mathcal{X}_\mathcal{T}$. That is, for a given matrix M, one wants to solve the following best approximation problem:

$$\min_{\hat{M}} \| \hat{M} - M \|_F^2 \quad \text{s.t.} \quad \hat{M} \in \mathcal{X}_\mathcal{T}. \tag{30}$$

Since the structure of $\mathcal{X}_\mathcal{T}$ is too complex to perform the projection in one step, the idea behind Dykstra's algorithm is to use iterative projections. Let $\mathcal{P}_{\mathcal{H}_t}(\cdot)$ denote the projection onto $\mathcal{H}_t$ for each $t \in \mathcal{T}$. Also, we assume that $\mathcal{P}_\mathcal{X}(\cdot)$ is known.

Dykstra's algorithm starts by initializing the so-called normal matrices $N_\mathcal{X}^0 = \mathbf{0}$ and $N_t^0 = \mathbf{0}$ for all $t \in \mathcal{T}$. These normal matrices have the same size as the primal variable X in the ADMM scheme. Moreover, we initialize $X^0 = M$. The algorithm iterates for $q \geq 1$ as follows:

$$
\begin{aligned}
X^q &:= \mathcal{P}_\mathcal{X}\left(X^{q-1} + N_\mathcal{X}^{q-1} \right) \\
N_\mathcal{X}^q &:= X^{q-1} + N_\mathcal{X}^{q-1} - X^q \\
\left. \begin{aligned}
L_t &:= X^q + N_t^{q-1} \\
X^q &:= \mathcal{P}_{\mathcal{H}_t}(L_t) \\
N_t^q &:= L_t - X^q
\end{aligned} \right\} &\quad \text{for all } t \in \mathcal{T}
\end{aligned}
\tag{CycDyk}
$$

Since the polyhedra are considered in a cyclic order, the iterative scheme (CycDyk) is also known in the literature as Dykstra's cyclic projection algorithm. The sequence $\{X^q\}_{q \geq 1}$ strongly converges to the solution of the best approximation problem (30), see e.g., [8, 6, 22].

We perform several actions to implement the algorithm (CycDyk) as efficiently as possible. First, we can reduce the number of iterations needed to converge by adding some of the constraints of $\mathcal{X}$ also to the sets $\mathcal{H}_t$. This brings the sets $\mathcal{H}_t$ closer to the intersection $\mathcal{X}_\mathcal{T}$, leading to faster convergence. A restriction on this addition is that we should still be able to find the explicit projection onto $\mathcal{H}_t$. In Section 4.3 we show how some of the constraints from the DNN relaxation of the bisection problem are added to the polyhedra $\mathcal{H}_t$, while keeping the structure of the polyhedra sufficiently simple.

Second, as observed in [40], it is possible to partly parallelize the algorithm (CycDyk). The cuts in $\mathcal{T}$ are often very sparse. This implies that the projection onto $\mathcal{H}_t$ only involves a small number of entries, while the other entries are kept fixed. This property can be exploited by clustering non-overlapping cuts. Suppose two cuts $t_1, t_2 \in \mathcal{T}$ are such that no entry in the matrix variable of

the X-subproblem is perturbed by both $\mathcal{P}_{\mathcal{H}_{t_1}}(\cdot)$ and $\mathcal{P}_{\mathcal{H}_{t_2}}(\cdot)$. Then, we can project onto both cuts simultaneously. This idea can be generalized by creating clusters of non-overlapping cuts. Suppose we cluster the set $\mathcal{T}$ into r clusters C_i, $i \in [r]$ such that $C_1 \cup \ldots \cup C_r = \mathcal{T}$, $C_i \cap C_j = \emptyset$ for $i \neq j$, $i, j \in [r]$, and all cuts in C_i, $i \in [r]$ are non-overlapping. Then, an iterate of (CycDyk) is performed in $r + 1$ consecutive steps, instead of $|\mathcal{T}| + 1$.

To cluster the cuts, we proceed as follows. We denote by H an undirected graph in which each vertex represents a cutting plane indexed by an element from $\mathcal{T}$. Two vertices in H are connected by an edge if and only if two cuts are overlapping. Clustering $\mathcal{T}$ into non-overlapping sets corresponds to clustering vertices of H into independent sets. Therefore, clustering $\mathcal{T}$ into the smallest number of non-overlapping sets reduces to finding a minimum coloring in H. Since the graph coloring problem is NP-hard, we use an efficient heuristic algorithm from [23] to find a near-optimal coloring.

3.3 The cutting-plane ADMM

In this section we put all elements of our cutting-plane algorithm together. In particular, we combine the ADMM from Section 3.1 and the clustered implementation of Dykstra's projection algorithm from Section 3.2 into a cutting-plane ADMM-based algorithm. We refer to this algorithm as the cutting-plane ADMM. Algorithm 1 provides a pseudo-code of our algorithm. Since the CP–ADMM algorithm solves a two-block separable convex problem it is guaranteed to converge, see e.g., [7, 55]. The stopping criteria and input parameters are specified in Section 3.3.2 and Section 5, respectively.

The CP–ADMM algorithm is designed to solve DNN relaxations for the GPP with additional cutting planes. In particular, Algorithm 1 can solve the DNN relaxation for the k-EP, see (13), that is strengthened by the triangle inequalities (34) and independent set inequalities (37). Similarly, Algorithm 1 also solves the DNN relaxation for the GBP, see (23), with additional BQP inequalities (41).

Let us outline the main steps of the CP–ADMM. Initially, the set $\mathcal{T}$ is empty and the algorithm solves the basic DNN relaxation, i.e., the DNN relaxation without additional cutting planes, using the ADMM as described in Section 3.1. After one of the stopping criteria from the inner while-loop is satisfied, see Section 3.3.2, a valid lower bound is computed based on the current approximate solution, see Section 3.3.1. Then, the algorithm identifies violated cuts and adds the $numCuts$ most violated ones to $\mathcal{T}$. To increase performance, the cuts induced by tuples in $\mathcal{T}$ are clustered by using a heuristic for the graph coloring problem from [23]. The procedure is repeated, where the projection onto $\mathcal{X}_{\mathcal{T}}$, see (29), is performed by the semi-parallelized version of Dykstra's projection algorithm, see Section 3.2. The outer while-loop stops whenever one of the global stopping criteria is met.

The CP–ADMM can be extended to solve various DNN relaxations with a large number of additional cutting planes. We remark that computing such strong bounds was not possible until now even for medium-sized problems and limited number of cutting planes.

Algorithm 1: CP–ADMM for the GPP

   Data: weighted Laplacian matrix $\bar{L}$, $m_1 \geq \ldots \geq m_k$;
   Input: UB, $\varepsilon_{\text{ADMM}}$, $\varepsilon_{\text{proj}}$, $numCuts$, $maxOuterLoops$, $maxTime$;
   Output: valid lower bound $lb(R^p, Z^p)$;

1 Initialization: Set (R^0, X^0, Z^0), σ^0 and γ by using (27) or (28). Set $p = 0$, $\mathcal{T} = \emptyset$;
2 Obtain V by applying a QR-decomposition to (8) for the k-EP and to (18) for the GBP;
3 **while** *stopping criteria not met* **do**
4   **while** *stopping criteria not met* **do**
5        $R^{p+1} = \mathcal{P}_{\succeq \mathbf{0}}\left(V^\top \left(X^p + \frac{1}{\sigma^p} Z^p\right) V\right)$;
6        $X^{p+1} = \mathcal{P}_{\mathcal{X}_{\mathcal{T}}}\left(V R^{p+1} V^\top - \frac{1}{\sigma^p}\left(\bar{L} + Z^p\right)\right)$ by solving (30) using (CycDyk);
7        $Z^{p+1} = Z^p + \gamma \cdot \sigma^p \cdot (X^{p+1} - V R^{p+1} V^\top)$;
8   If adaptive stepsize term is used, update σ^{p+1} by using (26);
9        $p \leftarrow p + 1$;
10   **end**
11   Compute a valid lower bound $lb(R^p, Z^p)$ by using (32) ;
12   Identify the violated inequalities and add the $numCuts$ most violated cuts to $\mathcal{T}$;
13   Cluster the cuts in $\mathcal{T}$;
14 **end**

14

3.3.1 Valid lower bounds

There are existing several ways to obtain valid lower bounds when stopping iterative algorithms earlier, see e.g., [42, 36]. We compute valid lower bounds by exploiting the approach from [36].

We use our uniform notation for the k-EP and the GBP to derive the Lagrangian dual problem for both problems. Let $\mathcal{L}(X, R, Z) := \mathcal{L}_0(X, R, Z)$, see (24), denote the Lagrangian function of (13) and (23) with respect to dualizing $X = VRV^\top$. Then, the Lagrangian dual of (13) and (23) is

$$\max_{Z \in \mathcal{S}^q} \min_{X \in \mathcal{X}_\mathcal{T}, R \in \mathcal{R}} \mathcal{L}(X, R, Z) = \max_{Z \in \mathcal{S}^q} \left\{ \min_{X \in \mathcal{X}_\mathcal{T}} \langle \bar{L} + Z, X \rangle - \operatorname{tr}(R)\lambda_{\max}\left(V^\top Z V\right) \right\}, \tag{31}$$

where $\lambda_{\max}(V^\top Z V)$ is the largest eigenvalue of $V^\top Z V$, and q is the appropriate order of the positive semidefinite cone. In (31) we exploit the well-known Rayleigh principle. It follows from (31) that for any $Z \in \mathcal{S}^q$ one can obtain a valid lower bound by computing:

$$lb(R, Z) = \min_{X \in \mathcal{X}_\mathcal{T}} \langle \bar{L} + Z, X \rangle - \operatorname{tr}(R)\lambda_{\max}(V^\top Z V). \tag{32}$$

Since the minimization problem in (32) is a linear programming problem, the computation of valid lower bounds is efficient.

3.3.2 Stopping criteria for the CP–ADMM algorithm

We use different stopping criteria for the inner and outer while-loops in Algorithm 1. The following measure is used as one of the stopping criteria for the inner while-loop:

$$\max \left\{ \frac{\|X^p - VR^pV^\top\|_F}{1 + \|X^p\|_F}, \sigma \frac{\|X^{p+1} - X^p\|_F}{1 + \|Z^p\|_F} \right\} < \varepsilon_{\text{ADMM}},$$

where $\varepsilon_{\text{ADMM}}$ is the prescribed tolerance precision. We also stop the inner while-loop when $maxTime$ is reached.

The Dykstra's projection algorithm (CycDyk) stops when $\|X^{q+1} - X^q\|_F < \varepsilon_{\text{proj}}$ for a given input parameter $\varepsilon_{\text{proj}}$.

We consider the following types of stopping criteria for the outer while-loop:

- The algorithm stops if the gap between a valid lower bound, that is rounded up to the closest integer, and a given upper bound UB is closed.

- The algorithm stops if an improvement in lower bounds between two consecutive outer loops is less than the prescribed threshold, i.e., 0.001.

- The algorithm stops if the number of new cuts to be added in the next outer loop is small, i.e., $< 0.25n$.

- The algorithm stops if the maximum number of outer loops $maxOuterLoops$ is reached.

- The algorithms stops immediately if the maximum computation time $maxTime$ is reached.

We specify the values of the input parameters in Section 5.

3.3.3 Efficient ingredients of the CP–ADMM algorithm

Algorithm 1 is efficient due to the following ingredients:

1. Warm starts. After identifying new cuts we start the new ADMM iterate from the last obtained triple (R^p, X^p, Z^p). Observe that there is no warm start strategy for an interior point method.

2. Scaling of data. It is known that the performance of a first order method can be improved by appropriate scaling of data. Therefore, we scale the objective by a scalar $S \in \mathbb{R}$ that depends on the problem and its size. Namely, for the k-EP we set $S = 1/\|L\|_F$ for $n \leq 400$, $S = k/\|L\|_F$ for $400 < n \leq 800$, and $S = n/(k\|L\|_F)$ otherwise. For the GBP we use $S = 1$. The values for S are obtained by extensive numerical tests.

3. Clustering. A crucial ingredient for improving the performance of Dykstra's projection algorithm is clustering cuts, see Section 3.2.

4. Separation. We introduce a probabilistic independent set separation method to separate independent set inequalities, see Algorithm 2 in Section 4.

When we run the CP–ADMM without any cutting planes (i.e., $\mathcal{T} = \emptyset$), the bottleneck of the code is the projection onto the positive semidefinite cone. When we start adding cuts, Dykstra's algorithm starts taking over the major part of the computation time.

4 Valid cutting planes, their projectors and separators

In this section we consider various families of cutting planes that strengthen the DNN relaxations for the k-EP and GBP. In the light of adding them in the cutting-plane augmented Lagrangian algorithm of Section 3, we present for each cut type a polyhedral set $\mathcal{H}_t$ induced by the cut (and, possibly, a subset of the constraints from the corresponding DNN relaxation). We show how to explicitly project a matrix onto these polyhedral sets. The efficient separation of these cut types is also considered.

In total we consider three types of cutting planes: two for the k-EP and one for the GBP.

4.1 Triangle inequalities for the k-EP

Let us consider the relaxation (DNN_{EP}), see (7) for the k-equipartition problem. Marcotorchino [39] as well as Grötschel and Wakabayashi [25] observe that the linear relaxation of the k-equipartition problem can be strengthened by adding the triangle inequalities:

$$Y_{ij} + Y_{il} \leq 1 + Y_{jl} \quad \text{for all triples } (i,j,l), i \neq j, j \neq l, i \neq l. \tag{33}$$

For a given triple (i,j,l) of distinct vertices, the triangle constraint (33) ensures that if i and j are in the same set of the partition and so are i and l, then also j and l have to belong to the same set of the partition. Karisch and Rendl [32] use these inequalities to strengthen (DNN_{EP}).

To obtain the equivalent facially reduced relaxation (9), we apply the linear transformation $X = Y - \frac{1}{k}\mathbf{J}_n$, see Section 2.1. As we apply our cutting-plane algorithm on this latter relaxation, we also perform this transformation on the triangle inequalities. The transformed cuts are as follows:

$$X_{ij} + X_{il} \leq \frac{k-1}{k} + X_{jl} \quad \text{for all triples } (i,j,l), i \neq j, j \neq l, i \neq l. \tag{34}$$

Observe that there exist $3\binom{n}{3}$ triangle inequalities.

To incorporate the cutting planes (34) into our cutting-plane augmented Lagrangian algorithm, we define for each cut a polyhedral set that is induced by the cut. For each triple (i,j,l) we define the polyhedron $\mathcal{H}_{ijl}^{\Delta} \subseteq \mathcal{S}^n$ as follows:

$$\mathcal{H}_{ijl}^{\Delta} := \left\{ X \in \mathcal{S}^n \; : \; X_{ij} + X_{il} \leq \frac{k-1}{k} + X_{jl} \right\}. \tag{35}$$

Let $\mathcal{P}_{\mathcal{H}_{ijl}^{\Delta}} : \mathcal{S}^n \to \mathcal{S}^n$ denote the operator that projects a matrix in $\mathcal{S}^n$ onto $\mathcal{H}_{ijl}^{\Delta}$. As the idea behind Dykstra's cyclic projection algorithm suggests, this projector can be characterized by a closed form expression.

Lemma 4. *Let $M \in \mathcal{S}^n$ and let $\hat{M} := \mathcal{P}_{\mathcal{H}_{ijl}^\Delta}(M)$. If $M \in \mathcal{H}_{ijl}^\Delta$, then $\hat{M} = M$. If $M \notin \mathcal{H}_{ijl}^\Delta$, then $\hat{M}$ is such that*

$$
\hat{M}_{pq} = \begin{cases}
\frac{2}{3}M_{ij} - \frac{1}{3}M_{il} + \frac{1}{3}M_{jl} + \frac{1}{3} - \frac{1}{3k} & \text{if } (p,q) \in \{(i,j),(j,i)\}, \\
-\frac{1}{3}M_{ij} + \frac{2}{3}M_{il} + \frac{1}{3}M_{jl} + \frac{1}{3} - \frac{1}{3k} & \text{if } (p,q) \in \{(i,l),(l,i)\}, \\
\frac{1}{3}M_{ij} + \frac{1}{3}M_{il} + \frac{2}{3}M_{jl} - \frac{1}{3} + \frac{1}{3k} & \text{if } (p,q) \in \{(j,l),(l,j)\}, \\
M_{pq} & \text{otherwise.}
\end{cases}
$$

Proof. See Appendix B.1. □

Identifying the most violated inequalities of the form (34) can be done by a complete enumeration. This separation can be done in $O(n^3)$.

4.2 Independent set inequalities for the k-EP

Chopra and Rao [12] introduced a further type of inequalities that are valid for the linear relaxation of the k-equipartition problem, namely

$$
\sum_{i,j \in I, i < j} Y_{ij} \geq 1 \quad \text{for all } I \subseteq V \text{ with } |I| = k+1, \tag{36}
$$

which are known as the independent set inequalities. These inequalities have also been used for the SDP relaxation in the work of Karisch and Rendl [32]. The intuition behind these constraints is that for all subsets of $k+1$ nodes, there must always be two nodes that are in the same set of the partition. Thus, the graph with adjacency matrix Y has no independent set of size $k+1$.

Using the linear transformation $X = Y - \frac{1}{k}\mathbf{J}_n$, we obtain the following equivalent inequalities that are valid for the facially reduced relaxation (9):

$$
\sum_{i,j \in I, i < j} X_{ij} \geq \frac{1-k}{2} \quad \text{for all } I \text{ with } |I| = k+1. \tag{37}
$$

Observe that there are $\binom{n}{k+1}$ independent set inequalities.

Let us define for each set $I \subseteq V$ with $|I| = k+1$ a polyhedral set $\mathcal{H}_I^{IS} \subseteq \mathcal{S}^n$ that is induced by the cut, i.e.,

$$
\mathcal{H}_I^{IS} := \left\{ X \in \mathcal{S}^n : \sum_{i,j \in I, i < j} X_{ij} \geq \frac{1-k}{2} \right\}. \tag{38}
$$

We let $\mathcal{P}_{\mathcal{H}_I^{IS}} \colon \mathcal{S}^n \to \mathcal{S}^n$ denote the projector onto the polyhedron $\mathcal{H}_I^{IS}$. This projection can be performed explicitly, as shown by the following result.

Lemma 5. *Let $M \in \mathcal{S}^n$ and let $\hat{M} := \mathcal{P}_{\mathcal{H}_I^{IS}}(M)$. If $M \in \mathcal{H}_I^{IS}$, then $\hat{M} = M$. If $M \notin \mathcal{H}_I^{IS}$, then $\hat{M}$ is such that*

$$
\hat{M}_{pq} = \begin{cases}
M_{pq} - \frac{k-1}{k(k+1)} - \frac{2}{k(k+1)}\sum_{i<j, i, j \in I} M_{ij}, & \text{if } p, q \in I, \ p \neq q, \\
M_{pq}, & \text{otherwise.}
\end{cases}
$$

Proof. See Appendix B.2. □

In order to find the most violated inequalities of type (37), we need a separator for independent set inequalities. It is known that exact separation of these inequalities is NP-hard [18]. Complete enumeration leads to a running time of $O(n^{k+1})$, which is computationally tractable only for $k = 2$ and $k = 3$. For larger k, we apply a combination of two separation heuristics to identify violated inequalities. First, we apply the deterministic separation heuristic from Anjos et al. [2]. This method

efficiently generates at most n inequalities, which turn out to be effective as numerical experiments in [2] suggest.

On top of the heuristic from [2], we also introduce a probabilistic independent set inequality separation heuristic. Although this algorithm relies on the same idea as the deterministic heuristic from [2], the greedy selection of a new vertex to add in the set C is randomized with probabilities inversely proportional to their values in the current solution matrix X. A pseudo-code of this heuristic is given in Algorithm 2. The parameter N_R corresponds to the number of repetitions, while $\varepsilon > 0$ is a sensitivity parameter. Low values of ε lead to very sensitive behaviour with respect to differences in the current solution X, while the selection eventually resembles a uniform distribution when ε is increased. The advantage of this randomization is that the combination of both heuristics can yield more than n violated independent set inequalities.

Algorithm 2: Probabilistic separation method for independent set inequalities

Data: the number of partitions k, the size of graph n;
Input: output matrix X from the ADMM, number of repetitions N_R, sensitivity parameter $\varepsilon > 0$;
Output: a collection of violated distinct independent set inequalities $\mathcal{C}$, its violation vector v;

1 Initialization: $Y = X + \frac{1}{k}\mathbf{J}_n, \mathcal{C} = \emptyset$;
2 **for** $r \in [N_R]$ **do**
3   Choose vertex v uniformly at random from $[n]$;
4      $C \leftarrow \{v\}$;
5      $S \leftarrow [n] \setminus \{v\}$;
6   **for** $l \in [k]$ **do**
7   Define $p_i := \sum_{j \in C} y_{ij} + \varepsilon$ for all $i \in S$;
8   Define $q_i := \frac{(1/p_i)}{\sum_{i \in S}(1/p_i)}$ for all $i \in S$;
9   Randomly select vertex $i \in S$ according to probability mass function $\{q_i\}_{i \in S}$;
10         $C \leftarrow C \cup \{i\}, \ S \leftarrow S \setminus \{i\}$;
11   **end**
12   **if** $C \notin \mathcal{C}$ **then**
13         $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$;
14   **end**
15 **end**

4.3 BQP inqualities for the GBP

We now consider the relaxation (DNN_{BP}), see (15). The relaxation (DNN_{BP}) can be further strengthened by adding the following inequalities:

$$0 \leq X_{ij} \leq X_{ii} \tag{39}$$

$$X_{ii} + X_{jj} \leq 1 + X_{ij} \tag{40}$$

$$X_{il} + X_{jl} \leq X_{ll} + X_{ij} \tag{41}$$

$$X_{ii} + X_{jj} + X_{ll} \leq X_{ij} + X_{il} + X_{jl} + 1, \tag{42}$$

where $X = (X_{ij}) \in \mathcal{S}^{2n+1}$ and $1 \leq i,j,l \leq 2n$, $i \neq j$, $i \neq l$, $j \neq l$. The inequalities (39)–(42) are facet defining inequalities of the boolean quadric polytope [43]. Wolkowicz and Zhao [57] prove that the inequalities (39) and (40) are already implied by the constraints in (14). Moreover, preliminary numerical results show that the inequalities (41) make larger improvements in the bounds when added to the DNN relaxation than the inequalities (42). Therefore, we consider only the constraints (41) within our algorithm.

Different from the SDP relaxation of the k-EP, the polyhedral set $\mathcal{X}_{BP}$ is a subset of the lifted space $\mathcal{S}^{2n+1}$. As a result, the polyhedral part induced by a BQP cut of the form (41) is also a subset of $\mathcal{S}^{2n+1}$. For each triple (i,j,l) with $2 \leq i,j,l \leq 2n + 1$, $i \neq j$, $i \neq l$, $j \neq l$, we define the following

18

polyhedron:

$$\mathcal{H}_{ijl}^{BQP} := \left\{ \begin{pmatrix} 1 & (x^1)^\top & (x^2)^\top \\ x^1 & X^{11} & X^{12} \\ x^2 & X^{21} & X^{22} \end{pmatrix} \in \mathcal{S}^{2n+1} : \begin{array}{c} X = \begin{pmatrix} X^{11} & X^{12} \\ X^{21} & X^{22} \end{pmatrix}, X_{il} + X_{jl} \leq X_{ll} + X_{ij} \\ \mathrm{diag}(X^{11}) = x^1, \mathrm{diag}(X^{22}) = x^2, x^1 + x^2 = \mathbf{1}_n \end{array} \right\}.$$

(43)

The polyhedron $\mathcal{H}_{ijl}^{BQP}$ is not only induced by the BQP cut, it also contains a subset of the constraints of the relaxation (15). This idea is inspired by the approach in [40], where the inclusion of additional constraints in each polyhedron in Dykstra's algorithm speeds up the convergence. Since the structure of $\mathcal{H}_{ijl}^{BQP}$ must remain simple enough to project onto it via a closed form expression, it is impractical to add all constraints from (15). The set $\mathcal{H}_{ijl}^{BQP}$ is chosen such that we are still able to project onto it explicitly.

Let $\mathcal{P}_{\mathcal{H}_{ijl}^{BQP}} : \mathcal{S}^{2n+1} \to \mathcal{S}^{2n+1}$ denote the projector onto $\mathcal{H}_{ijl}^{BQP}$. Given that the matrix that is projected already satisfies the constraints $\mathrm{diag}(X^{11}) = x^1, \mathrm{diag}(X^{22}) = x^2$ and $x^1 + x^2 = \mathbf{1}_n$, which is always the case in our implementation, this projector is specified by the result below.

Lemma 6. *Let* $M = \begin{pmatrix} 1 & diag(M^{11})^\top & diag(M^{22})^\top \\ diag(M^{11}) & M^{11} & M^{12} \\ diag(M^{22}) & M^{21} & M^{22} \end{pmatrix} \in \mathcal{S}^{2n+1}$ *be such that* $diag(M^{11}) +$ $diag(M^{22}) = \mathbf{1}_n$ *and let* $\hat{M} := \mathcal{P}_{\mathcal{H}_{ijl}^{BQP}}(M)$. *If* $M_{il} + M_{jl} \leq M_{ij} + \frac{1}{6}M_{ll} + \frac{1}{3}M_{1l} - \frac{1}{6}M_{l^*l^*} - \frac{1}{3}M_{1l^*} + \frac{1}{2}$, *then*

$$\hat{M}_{pq} = \begin{cases} \frac{1}{6}M_{ll} + \frac{1}{3}M_{1l} - \frac{1}{6}M_{l^*l^*} - \frac{1}{3}M_{1l^*} + \frac{1}{2} & \text{if } (p,q) \in \{(l,l),(1,l),(l,1)\}, \\ -\frac{1}{6}M_{ll} - \frac{1}{3}M_{1l} + \frac{1}{6}M_{l^*l^*} + \frac{1}{3}M_{1l^*} + \frac{1}{2} & \text{if } (p,q) \in \{(l^*,l^*),(1,l^*),(l^*,1)\}, \\ M_{pq} & \text{otherwise.} \end{cases}$$

Otherwise, $\hat{M}$ *is such that*

$$\hat{M}_{pq} = \begin{cases} \frac{7}{10}M_{il} - \frac{3}{10}M_{jl} + \frac{3}{10}M_{ij} + \frac{1}{20}M_{ll} + \frac{1}{10}M_{1l} - \frac{1}{20}M_{l^*l^*} - \frac{1}{10}M_{1l^*} + \frac{3}{20} & \text{if } (p,q) \in \{(i,l),(l,i)\}, \\ -\frac{3}{10}M_{il} + \frac{7}{10}M_{jl} + \frac{3}{10}M_{ij} + \frac{1}{20}M_{ll} + \frac{1}{10}M_{1l} - \frac{1}{20}M_{l^*l^*} - \frac{1}{10}M_{1l^*} + \frac{3}{20} & \text{if } (p,q) \in \{(j,l),(l,j)\}, \\ \frac{3}{10}M_{il} + \frac{3}{10}M_{jl} + \frac{7}{10}M_{ij} - \frac{1}{20}M_{ll} - \frac{1}{10}M_{1l} + \frac{1}{20}M_{l^*l^*} + \frac{1}{10}M_{1l^*} - \frac{3}{20} & \text{if } (p,q) \in \{(i,j),(j,i)\}, \\ \frac{1}{10}M_{il} + \frac{1}{10}M_{jl} - \frac{1}{10}M_{ij} + \frac{3}{20}M_{ll} + \frac{1}{10}M_{1l} - \frac{3}{20}M_{l^*l^*} - \frac{3}{10}M_{1l^*} + \frac{9}{20} & \text{if } (p,q) \in \{(l,l),(1,l), \\ & \qquad (l,1)\} \\ -\frac{1}{10}M_{il} - \frac{1}{10}M_{jl} + \frac{1}{10}M_{ij} - \frac{3}{20}M_{ll} - \frac{1}{10}M_{1l} + \frac{3}{20}M_{l^*l^*} + \frac{3}{10}M_{1l^*} + \frac{11}{20} & \text{if } (p,q) \in \{(l^*,l^*),(1,l^*), \\ & \qquad (l^*,1)\} \\ M_{pq} & \text{otherwise.} \end{cases}$$

where l^* *is obtained from* l *by* $l^* := 2 + (l + n - 2) \bmod 2n$.

Proof. See Appendix B.3. $\qquad\qquad\qquad\square$

Separating the BQP inequalities (41) can be done in $O(n^3)$ by complete enumeration.

5 Numerical results

We implemented our algorithm CP–ADMM in Matlab. For efficiency some separation routines have been coded in C. In order to evaluate the quality of the bounds and the run times to compute these bounds, we test our algorithms on various instances from the literature. All experiments were run on an Intel Xeon, E5-1620, 3.70 GHz with 32 GB memory. To compute valid lower bounds after each run of the inner while-loops we use Mosek [41]. Note that the computation of a valid bound after the inner while-loops is necessary for checking the stopping criteria.

We now describe the data sets used in our evaluation. Most of these instances were also considered in [3] and [4].

- $G_{|V|,|V|p}$ and $U_{|V|,|V|\pi d^2}$: randomly generated graphs by Johnson et al. [31].

  - $G_{|V|,|V|p}$: graphs $G = (V,E)$, with $|V| \in \{124, 250, 500, 1000\}$ and four individual edge probabilities p. These probabilities were chosen depending on $|V|$, so that the average expected degree of each node was approximately $|V|p \in \{2.5, 5, 10, 20\}$.

  - $U_{|V|,|V|\pi d^2}$: graphs $G = (V,E)$, with $|V| \in \{500, 1000\}$ with distance value d such that $|V|\pi d^2 \in \{5, 10, 20, 40\}$. To form such a graph $G = (V,E)$, one chooses $2|V|$ independent numbers uniformly from the interval $(0,1)$ and views them as coordinates of $|V|$ nodes on the unit square. An edge is inserted between two vertices if and only if their Euclidean distance is at most d.

- Mesh graphs from [52, 20]: Instances from finite element meshes; all nonzero edge weights are equal to one. Graph names begin with 'mesh', followed by the number of vertices and the number of edges.

- KKT graphs: These instances originate from nested bisection approaches for solving sparse symmetric linear systems. Each instance consists of a graph that represents the support structure of a sparse symmetric linear system, for details see [27].

- Toroidal 2D- and 3D-grid graphs arise in physics when computing ground states for Ising spinglasses, see e.g., [27]. They are generated using the `rudy` graph generator [48]:

  - spinglass2pm_n_r: A toroidal 2D-grid for a spinglass model with weights $\{+1, -1\}$. The grid has size $n_r \times n_r$, i.e., $|V| = n_r^2$. The percentage of edges with negative weights is 50 %.

  - spinglass3pm_n_r: A toroidal 3D-grid for a spinglass model with weights $\{+1, -1\}$. The grid has size $n_r \times n_r \times n_r$, i.e., $|V| = n_r^3$. The percentage of edges with negative weights is 50 %.

5.1 Numerical results for the k-EP

In the CP–ADMM, see Algorithm 1, we input an upper bound UB and the parameters $maxTime$, $numCuts$, $maxOuterloops$, $\varepsilon_{\mathrm{proj}}$, and $\varepsilon_{\mathrm{ADMM}}$. We also require the bounds $\sigma_{\min}$, and $\sigma_{\max}$ for the adaptive stepsize term. The setting of these parameters is as follows:

- As an upper bound we input the values we obtained by heuristics or the optimal solution given in the literature.

- The maximal number of cuts added in each outer while-loop, $numCuts$, is $3n$ for graphs with $n \leq 300$ and $5n$ when $n > 300$. These values are determined by preliminary tests, see also Appendix C.

- The maximal number of outer while-loops is 30 for instances with $n \leq 300$, and 10 when $n > 300$.

- The precision for Dykstra's projection algorithm $\varepsilon_{\mathrm{proj}}$ is set to 10^{-4}.

- The inner precision $\varepsilon_{\mathrm{ADMM}}$ is 10^{-4} in the last iteration and 10^{-3} in all previous loops.

- The maximum computation time $maxTime$ is set to 2 hours.

- The bounds $\sigma_{\min}$ and $\sigma_{\max}$ for σ^p are 10^{-5} and 10^3, respectively.

In each outer while-loop we separate $MaxIneq$ violated inequalities. We experimented with two strategies: One strategy is to first add violated triangle inequalities and, in case less than $MaxIneq$ violated triangle inequalities are found, we add violated independent set inequalities. The other strategy is to mix the two kinds of cuts and search for violated triangle and independent set inequalities together. The experiments showed that the latter strategy obtains better results, i.e., better

bounds within the same time. Therefore, in the final setting we search for the most violated inequalities from both, the triangle and independent set inequalities.

The separation of triangle inequalities is done by complete enumeration. Searching for independent set inequalities is also done by complete enumeration if $k \in \{2, 3\}$. For $k \in \{4, 5\}$ we apply the heuristic from [2] and Algorithm 2, as explained in Section 4.2.

In Table 1 we compare the eigenvalue lower bound by Donath and Hoffman [15] (denoted by DH) with the lower bound when computing the DNN relaxation as well as the DNN relaxation with additional cuts on selected graphs from our testbed. We do not include other bounds from the literature, such as bounds from [26], as they are weaker than the DH bound. The DH bound is obtained by solving the corresponding SDP relaxation using Mosek [41]. The numbers in the table show that the DNN bound significantly improves over the DH bound. Moreover, our ADMM algorithm requires less time and memory to compute the DNN bound than Mosek to compute the DH bound. Adding cuts to the DNN gives an even more substantial improvement. Hence, including triangle and independent set inequalities is much stronger than including non-negativity constraints only. As the DH bound is not competitive with the DNN bound and far from the DNN+cuts bound, we will not include it in the subsequent presentation of the numerical results.

Computing an equipartition using commercial solvers is out of reach unless the graphs are extremely sparse. For instances in Table 1, Gurobi (with the default settings) solves the small and very sparse instances within a few seconds, but obtains a gap of more than 40 % after 2 hours for larger, but still reasonably sparse graphs. E.g., for $G_{250,5}$, a graph on 250 vertices with density 2 %, the gap after two hours is more than 40 %. For $G_{250,20}$, a graph on 250 vertices and density 8 %, the gap is even more than 80 % after two hours. This limited power of commercial solvers to tackle the k-EP is also observed in [10]. Hence, we omit the comparison to Gurobi or other LP-based solvers in the tables.

Graph	n	k	ub	DH	DNN	DNN + Cuts
mesh.70.120	70	2	7	1.93	2.91	6.02
KKT.lowt01	82	2	13	2.47	4.88	12.43
mesh.148.265	148	4	22	5.46	8.13	21.23
$G_{124,2.5}$	124	2	13	4.59	7.29	12.01
$G_{124,10}$	124	2	178	138.24	152.86	170.88
$G_{124,20}$	124	2	449	403.08	418.67	439.96
$G_{250,2.5}$	250	2	29	10.99	15.16	28.30
$G_{250,5}$	250	2	114	70.21	81.52	105.00
$G_{250,10}$	250	2	357	280.25	303.02	330.40

Table 1: Comparison of different lower bounds

In Tables 2 to 8 we give the details of our numerical results. In the first 4 columns we list the name of the instance, the number of vertices n, the partition number k and an upper bound. The upper bounds are obtained by heuristics or the optimal solution from the literature.

In columns 5 and 6 the lower bound (rounded up) and the time when solving the DNN relaxation are given. Finally, in the remaining columns we display the results when adding cuts to the DNN relaxation: we report the rounded up lower bounds, computation time and the improvement (in %) on the rounded up lower bounds with respect to the DNN relaxation without cuts. We decided to report the improvement with respect to the rounded values, since otherwise for small numbers the percentages are incredibly huge. E.g., for instance $U_{1000,5}$ and $k = 4$, the value of the DNN bound is 0.17, the DNN+cuts bound is 2.45, giving an improvement of 1,341.2 %. The rounded up values are 1 and 3, respectively, giving a 200 % improvement, which reflects the situation much better.

In columns 10 and 11 we list the number of triangle cuts and independent set cuts present when stopping the algorithm. In the last two columns, the number of iterations of the ADMM and the number of outer while-loops is reported.

As can be observed in all tables, the bounds improve drastically when adding triangle and independent set inequalities to the DNN relaxation while the time for computing these bounds is still reasonable. We remark here that we can stop the CP–ADMM algorithm at any time and provide a valid lower bound.

The results show that the CP–ADMM takes much more of the triangle inequalities when computing bounds for the k-equipartition problem. Remember that we search for triangle and independent set inequalities together. Hence, there are more triangle inequalities with a large violation which means that the triangle inequalities contribute more to the strength of the bound than the independent set inequalities.

We discuss the results in more detail in the subsequent sections.

5.1.1 Detailed results for $k = 2$

In Tables 2 and 3 we report results for $k = 2$. Table 2 includes graphs with up to 274 vertices, the DNN bound for these graphs can be computed within a few seconds. After adding in total some 500 up to 16,000 triangle inequalities and additional 300 up to roughly 7,000 independent set inequalities, the bound improves between 4.55 % and 300 %. In several cases, the bound closes the gap to the best known upper bound. Otherwise, the algorithm stops due to the little improvement of the bounds in consecutive outer while-loops. The time for computing these bounds ranges from a few seconds to 17 minutes.

In Table 3 we consider larger graphs with 500 and 1,000 vertices. The DNN bound can be computed for these graphs within 12 minutes. After adding triangle and independent set inequalities, the bound improves up to 200 % in a running of 40 seconds up to 2 hours. On the G graphs one observes that the improvement of the bound when adding cuts gets more significant as the graphs get sparser. Note that for all these instances we stop because the number of outer while-loops is reached or the improvement of the bounds in consecutive outer while-loops is too small.

We give further results for graphs from the literature in Table 13 in Appendix C. For most of these we prove optimality of the best found k-equipartition, confirming the high quality of our lower bounds.

5.1.2 Detailed results for $k > 2$

In Tables 4 to 8 we report results for $k > 2$. As in the case for $k = 2$, the bounds can be significantly improved while the time for obtaining the bounds is still reasonable. However, we close the gap for fewer instances as for $k = 2$. The improvement for the larger graphs after adding cuts to the DNN relaxation is up to 200 % for $k = 4$ and up to 300 % for $k = 5$.

For smaller graphs the CP–ADMM stops because the improvement of the lower bound compared to the previous iteration is below the threshold 0.001, see Section 3.3.2.The largest improvement in the DNN+cuts bound w.r.t. the DNN bound is 500 %. Again, we observe that for graphs with more than 250 vertices, the algorithm typically stops because the maximum number of outer loops is reached.

5.2 Numerical results for bisection

In Algorithm 1, we input an upper bound UB and the parameters $numCuts$, $maxOuterloops$, $\varepsilon_{\text{proj}}$, $\varepsilon_{\text{ADMM}}$, and $maxTime$. The setting of these parameters is as follows.

- As an upper bound we input the numbers we obtained by heuristics.

- The maximal number of cuts added in each outer while-loop, $numCuts$, is $3n$ for graphs with $n \leq 300$ and $5n$ when $n > 300$.

- The maximal number of outer while-loops is 30 for instances with $n \leq 300$, and 10 when $n > 300$.

- The precision for Dykstra's projection algorithm $\varepsilon_{\text{proj}}$ is set to 10^{-6}.

- The inner precision $\varepsilon_{\text{ADMM}}$ is 10^{-5} in the first and last inner while-loop, and 10^{-4} in all other inner while-loops.

- The maximum computation time $maxTime$ is set to 2 hours.

The parameters differ from the experiments for the k-equipartition problem since for the bisection problem the size of the matrix in the SDP is of order $2n + 1$, i.e., more than double the size than for the k-equipartition problem. Violated BQP inequalities (41) are found by an enumeration search.

We take $m_1 = \lceil np \rceil$ where n is the number of vertices in the graph and for p we choose a number out of $\{0.6, 0.65, 0.7\}$.

The results are given in Table 9 for smaller graphs and in Table 10 for larger graphs. Similar as for the k-equipartition problem, we observe a significant improvement of the bound after adding inequalities. For graphs with vertices up to 274, see Table 9, the improvement of the DNN+cuts bound over the DNN bound ranges between 2.23 % and 200 % after adding up to 16,000 BQP inequalities. For five of the graphs in Table 9 the algorithm stopped because the gap was closed, for four graphs the improvement of the lower bound was only minor and four times the number of violated cuts found was too small. Only for one graph the algorithm stopped due to the time limit. For the larger graphs we typically stop because of the time limit of 2 hours, see Table 10. For those graphs the DNN+cuts bound improves over the DNN bound between 0.44 % and 17.32 % after adding between 8,000 and 24,200 BQP inequalities.

One can observe that the results are somewhat weaker than for the k-equipartition problem. Note that the nature of the cuts added to the DNN relaxations for the k-EP and GBP differs, and the size of matrix variables in the GBP relaxations is more than twice larger than in the case of the k-EP for the same graph. For large graphs, we are able to add roughly up to 50,000 cuts for the k-EP and 24,000 cuts for the GBP within a time span of 2 hours.

6 Conclusions

This study aims to investigate and expand the boundary of obtaining strong DNN bounds with additional cutting planes for large graph partition problems. Due to memory requirements, state-of-the-art interior point methods are only capable of solving medium-size SDPs and are not suitable for handling lots of polyhedral cuts. We overcome both difficulties by utilizing a first order method in a cutting-plane framework. Our approach focuses on two variations of the graph partition problem: the k-equipartition problem and the graph bisection problem.

We first derive DNN relaxations for both problems, see (7) and (15), then we apply facial reduction to obtain strictly feasible equivalent relaxations, see (9) and (19), respectively. To prove the minimality of the face of the DNN cone containing the feasible set of (15), we exploit the dimension of the bisection polytope, see Theorem 1. After facial reduction, both relaxations enclose a natural splitting of the feasible set into polyhedral and positive semidefinite constraints. Moreover, both relaxations can be further strengthened by several types of cutting planes.

To solve both relaxations, we use an ADMM update scheme, see (25), that is incorporated in a cutting-plane framework, leading to the so-called CP–ADMM, see Algorithm 1. The cutting planes are handled in the polyhedral subproblem by exploiting a semi-parallelized version of Dykstra's cyclic projection algorithm, see Section 3.2. The CP–ADMM benefits from warm starts whenever new cuts are added, provides valid lower bounds even after solving with low precision, and can be implemented efficiently. A particular ingredient of the CP–ADMM are the projectors onto polyhedra induced by the cutting planes. Projection operators for three types of cutting planes that are effective for the graph partition problem, i.e., the triangle, independent set and BQP inequalities, are derived in Lemma 4, 5 and 6, respectively.

Numerical experiments show that using our CP–ADMM algorithm we are able to produce high-quality bounds for graphs up to 1,024 vertices. We experimented with several graph types from the literature. For structured graphs of medium size and the 2-EP we often close the gap in a few seconds or at most a couple of minutes. For bisection problems on those graphs, we also close the gaps in many cases. For larger graphs, we are able to add polyhedral cuts roughly up to 50,000 for k-EP and 24,000 for the GBP within 2 hours, which results in strong lower bounds. Our results provide benchmarks for solving medium and large scale graph partition problems.

This research can be extended in several directions. Motivated by the optimistic results for the k-EP and the GBP, we expect that strong bounds from DNN relaxations with additional cutting planes can be obtained for other graph partition problems, such as the vertex separator problem and the maximum cut problem. Moreover, since the major ingredients of our algorithm are presented generally, establishing an approach for solving general DNN relaxations with additional cutting planes is an interesting future research direction. Our results also provide new perspectives on solving large-scale optimization problems to optimality by using SDPs.

Acknowledgements

Table 2: Instances considered in [26, Table 9], $k = 2$

| graph | n | k | ub | DNN | | DNN+cuts | | | | | | |
				lb (rounded)	clocktime (s)	lb (rounded)	clocktime (s)	imp. %	# tri. cuts	# ind. set cuts	# iter.	# outer loops
$G_{124,2.5}$	124	2	13	8	1.25	13	31.56	62.50	3,298	1,166	848	12
$G_{124,5}$	124	2	63	47	0.65	62	72.30	31.91	4,354	482	1,325	13
$G_{124,10}$	124	2	178	153	0.40	171	9.86	11.76	2,663	432	570	9
$G_{124,20}$	124	2	449	419	0.39	440	6.67	5.01	2,266	509	485	8
$G_{250,2.5}$	250	2	29	16	4.83	29	968.05	81.25	15,865	2,135	2,068	24
$G_{250,5}$	250	2	114	82	2.67	105	174.11	28.05	10,063	374	1,184	14
$G_{250,10}$	250	2	357	304	2.10	331	27.74	8.88	5,352	415	575	9
$G_{250,20}$	250	2	828	747	1.77	781	10.59	4.55	4,516	617	484	7
mesh.70.120	70	2	7	3	0.19	7	2.42	133.33	474	366	351	4
mesh.74.129	74	2	8	4	0.20	8	3.74	100.00	575	313	518	4
mesh.138.232	138	2	8	2	1.42	8	124.38	300.00	5,298	3,810	1,135	22
mesh.148.265	148	2	7	3	2.25	7	6.15	133.33	1,064	1,156	481	5
mesh.274.469	274	2	7	2	13.69	7	1,000.60	250.00	14,038	7,334	1,926	26
KKT.lowt01	82	2	13	5	0.22	13	6.23	160.00	515	715	357	5

Table 3: Large G and U graphs from [31], $k = 2$

| graph | n | k | ub | DNN | | DNN+Cuts | | | | | | |
				lb (rounded)	clocktime (s)	lb (rounded)	clocktime (s)	imp. %	# tri. cuts	# ind. set cuts	# iter.	# outer loops
$G_{500,2.5}$	500	2	49	25	45.83	45	1,425.01	80.00	24,549	451	1,444	10
$G_{500,5}$	500	2	218	156	22.60	197	592.33	26.28	24,356	644	1,188	10
$G_{500,10}$	500	2	626	513	15.90	554	125.15	7.99	13,069	713	678	8
$G_{500,20}$	500	2	1,744	1,566	14.78	1,613	42.91	3.00	9,705	1,076	520	6
$G_{1000,2.5}$	1,000	2	102	45	421.80	74	7,206.86	64.44	44,815	185	1,731	9
$G_{1000,5}$	1,000	2	451	307	204.65	379	2,183.88	23.45	49,054	946	1,822	10
$G_{1000,10}$	1,000	2	1,367	1,113	152.35	1,179	523.82	5.93	25,413	1,272	1,251	7
$G_{1000,20}$	1,000	2	3,389	3,007	140.69	3,079	350.84	2.39	19,195	1,813	1,078	6
$U_{500,5}$	500	2	2	1	74.43	2	2,528.19	100.00	25,000	0	3,062	10
$U_{500,10}$	500	2	26	8	39.52	23	708.11	187.50	17,824	7,176	1,742	10
$U_{500,20}$	500	2	178	56	66.28	153	838.74	173.21	14,817	10,183	2,638	10
$U_{500,40}$	500	2	412	163	97.47	379	1,562.31	132.52	17,444	7,556	3,024	10
$U_{1000,5}$	1,000	2	1	0	725.58	0	7,200.31	0.00	50,000	0	6,237	10
$U_{1000,10}$	1,000	2	39	8	674.90	24	3,086.56	200.00	43,836	6,164	3,357	10
$U_{1000,20}$	1,000	2	222	51	213.67	136	3,877.13	166.67	42,215	7,785	3,016	10
$U_{1000,40}$	1,000	2	737	240	561.08	640	4,665.77	166.67	40,190	9,810	3,199	10

Table 4: Instances considered in [4], $k \in \{3, 4, 5, 6\}$

graph	n	k	ub	DNN lb (rounded)	clocktime (s)	DNN+cuts lb (rounded)	clocktime (s)	imp. %	# tri. cuts	# ind. set cuts	# iter.	# outer loops
mesh.69.212	69	3	10	4	0.04	9	1,093.39	125.00	1,375	4,421	1,266	28
mesh.138.232	138	3	13	2	0.30	12	1,019.33	500.00	3,862	8,558	1,877	30
$G_{124,2.5}$	124	4	23	13	0.18	21	215.18	61.54	5,124	456	1,741	15
$G_{124,5}$	124	4	104	80	0.09	94	53.86	17.50	3,160	188	1,003	9
$G_{124,10}$	124	4	290	251	0.09	261	5.54	3.98	1,519	130	325	6
$G_{124,20}$	124	4	720	660	0.07	670	3.04	1.52	949	221	234	4
mesh.148.265	148	4	22	9	0.20	22	101.94	144.44	3,382	458	1,197	10
mesh.70.120	70	5	20	12	0.04	18	97.04	50.00	1,401	279	1,570	8
KKT.putt01	115	5	104	82	0.62	101	208.23	23.17	2,255	310	2,193	19
$G_{250,2.5}$	250	5	58	27	0.67	50	839.04	85.19	12,682	68	2,047	17
$G_{250,5}$	250	5	209	147	0.37	172	97.98	17.01	6,747	3	726	9
$G_{250,10}$	250	5	629	523	0.28	540	11.74	3.25	2,605	5	257	5
$G_{250,25}$	250	5	1,418	1,263	0.25	1,277	4.90	1.11	1,545	8	176	3
mesh.138.232	138	6	24	9	0.12	21	332.90	133.33	5,974	650	1,605	16

Table 5: Two-dimensional spinglass graphs, $k \in \{3, 4, 5\}$

graph	n	k	ub	DNN lb (rounded)	clocktime (s)	DNN+cuts lb (rounded)	clocktime (s)	imp. %	# tri. cuts	# ind. set cuts	# iter.	# outer loops
spinglass2pm_18	324	3	−256	−291	11.40	−270	1,095.10	7.22	5,274	7,686	1,797	8
spinglass2pm_21	441	3	−352	−395	34.63	−370	1,204.90	6.33	3,280	5,540	2,786	4
spinglass2pm_24	576	3	−453	−512	67.53	−474	1,962.94	7.42	3,662	7,858	2,178	4
spinglass2pm_27	729	3	−580	−656	114.62	−607	3,169.04	7.47	7,289	7,291	2,587	4
spinglass2pm_30	900	3	−723	−809	219.84	−748	6,713.41	7.54	7,922	10,078	3,442	4
spinglass2pm_18	324	4	−263	−293	9.63	−271	297.85	7.51	6,417	63	1,003	4
spinglass2pm_20	400	4	−326	−361	14.45	−333	164.25	7.76	7,821	179	487	4
spinglass2pm_22	484	4	−399	−441	29.47	−408	408.03	7.48	16,686	254	535	7
spinglass2pm_24	576	4	−465	−516	47.83	−475	668.25	7.95	16,769	511	740	6
spinglass2pm_26	676	4	−551	−611	55.27	−562	897.17	8.02	19,628	652	687	6
spinglass2pm_28	784	4	−635	−709	97.88	−656	3,102.63	7.48	19,350	250	1,500	5
spinglass2pm_32	1,024	4	−823	−922	170.91	−842	3,978.22	8.68	15,198	162	1,572	3
spinglass2pm_20	400	5	−327	−362	14.31	−333	225.31	8.01	8,000	0	695	4
spinglass2pm_25	625	5	−507	−566	64.71	−515	1,221.12	9.01	12,500	0	1,150	4

| spinglass2pm_30 | 900 | 5 | −736 | −818 | 151.53 | −751 | 2,110.83 | 8.19 | 22,500 | 0 | 1,116 | 5 |

Table 6: Three-dimensional spinglass graphs, $k \in \{3, 4, 5\}$

graph	n	k	ub	DNN		DNN+cuts						
				lb (rounded)	clocktime (s)	lb (rounded)	clocktime (s)	imp. %	# tri. cuts	# ind. set cuts	# iter.	# outer loops
spinglass3pm_9	729	3	−769	−902	98.12	−821	6,438.35	8.98	20,349	12,456	2,177	9
spinglass3pm_8	512	4	−562	−643	29.31	−590	2,352.34	8.24	17,633	287	2,355	7
spinglass3pm_10	1,000	4	−1,058	−1,258	265.95	−1,152	7,202.49	8.43	24,790	210	2,523	5
spinglass3pm_10	1,000	5	−1,068	−1,263	207.50	−1,152	7,202.26	8.79	30,000	0	2,482	6

Table 7: Large G and U graphs from [31], $k = 4$

graph	n	k	ub	DNN		DNN+cuts						
				lb (rounded)	clocktime (s)	lb (rounded)	clocktime (s)	imp. %	# tri. cuts	# ind. set cuts	# iter.	# outer loops
$G_{500,2.5}$	500	4	96	43	19.91	70	1,491.29	62.79	24,966	34	1,419	10
$G_{500,5}$	500	4	375	253	10.78	297	867.97	17.39	23,646	23	1,063	10
$G_{500,10}$	500	4	1,016	806	6.81	834	70.81	3.47	8,737	36	380	5
$G_{500,20}$	500	4	2,753	2,418	5.61	2,434	27.93	0.66	5,232	44	254	4
$G_{1000,2.5}$	1,000	4	200	74	174.11	117	7,204.31	58.11	49,998	2	1,519	10
$G_{1000,5}$	1,000	4	767	485	81.82	570	3,446.47	17.53	50,000	0	1,340	10
$G_{1000,10}$	1,000	4	2,219	1,732	52.46	1,774	295.29	2.42	16,558	5	636	4
$G_{1000,20}$	1,000	4	5,422	4,620	45.23	4,640	132.93	0.43	8,229	12	414	3
$U_{500,5}$	500	4	22	2	50.02	6	1,596.20	200.00	23,854	1,146	2,966	10
$U_{500,10}$	500	4	115	25	46.02	54	1,208.51	116.00	24,672	328	1,560	10
$U_{500,20}$	500	4	358	162	68.67	304	1,502.73	87.65	24,228	772	2,061	10
$U_{500,40}$	500	4	1,020	549	94.56	927	1,644.27	68.85	23,946	1,054	2,546	10
$U_{1000,5}$	1,000	4	11	1	649.01	3	5,195.14	200.00	50,000	0	4,565	10
$U_{1000,10}$	1,000	4	182	23	218.23	51	2,929.24	121.74	49,820	180	2,081	10
$U_{1000,20}$	1,000	4	735	160	433.61	362	4,847.54	126.25	49,447	553	2,832	10
$U_{1000,40}$	1,000	4	1,596	671	534.88	1,342	5,502.61	100.00	49,100	900	3,147	10

Table 8: Large G and U graphs from [31], $k = 5$

graph	n	k	ub	DNN		DNN+cuts						
				lb (rouneded)	clocktime (s)	lb (rouneded)	clocktime (s)	imp. %	# tri. cuts	# ind. set cuts	# iter.	# outer loops
$G_{500,2.5}$	500	5	108	49	17.47	78	1,366.47	59.18	24,973	27	1,416	10
$G_{500,5}$	500	5	403	279	8.77	319	625.09	14.34	19,998	2	924	8
$G_{500,10}$	500	5	1,113	880	5.56	899	57.74	2.16	6,387	0	304	4
$G_{500,20}$	500	5	2,987	2,618	4.47	2,624	22.93	0.23	2,868	1	207	3
$G_{1000,2.5}$	1,000	5	228	83	157.92	133	7,207.64	60.24	50,000	0	1,514	10
$G_{1000,5}$	1,000	5	854	532	71.13	612	2,758.98	15.04	45,000	0	1,140	9
$G_{1000,10}$	1,000	5	2,449	1,881	40.91	1,908	205.35	1.44	11,499	0	510	3
$G_{1000,20}$	1,000	5	5,904	4,990	35.05	4,997	66.87	0.14	4,215	0	285	2
$U_{500,5}$	500	5	21	3	34.36	8	1,391.01	166.67	24,772	228	1,770	10
$U_{500,10}$	500	5	98	35	48.88	72	1,310.76	105.71	24,744	256	1,812	10
$U_{500,20}$	500	5	439	216	67.61	373	1,302.06	72.69	24,638	362	1,911	10
$U_{500,40}$	500	5	1,230	771	85.37	1,165	1,807.18	51.10	24,090	910	2,499	10
$U_{1000,5}$	1,000	5	29	1	529.28	4	5,104.28	300.00	50,000	0	3,908	10
$U_{1000,10}$	1,000	5	220	34	175.33	76	3,285.83	123.53	49,879	121	2,058	10
$U_{1000,20}$	1,000	5	716	216	411.07	469	4,581.52	117.13	49,336	664	2,706	10
$U_{1000,40}$	1,000	5	1,836	887	497.69	1,609	4,864.62	81.40	49,010	990	2,625	10

Table 9: Instances considered in [26, Table 9], $m = (m_1, n - m_1)$

graph	n	m_1	ub	DNN		DNN+cuts					
				lb (rounded)	clocktime (s)	lb (rounded)	clocktime (s)	imp. %	# BQP cuts	# iter.	# outer loops
$G_{124,2.5}$	124	75	12	7	11.29	10	7,077.11	42.86	7,399	29,486	21
$G_{124,5}$	124	87	49	42	14.23	49	616.35	16.67	4,054	15,861	13
$G_{124,10}$	124	81	159	141	12.26	150	154.13	6.38	2,429	12,633	8
$G_{124,20}$	124	75	414	399	12.93	412	373.60	3.26	2,716	19,067	8
$G_{250,2.5}$	250	175	20	12	202.63	16	7,200.52	33.33	15,750	38,639	21
$G_{250,5}$	250	163	101	74	129.11	90	1,710.97	21.62	9,702	23,906	15
$G_{250,10}$	250	150	343	289	81.51	310	441.65	7.27	5,454	16,145	9
$G_{250,20}$	250	175	673	627	94.47	641	473.62	2.23	4,416	17,655	6
mesh.138.232	138	90	6	2	21.51	6	2,647.73	200.00	7,452	24,424	17
mesh.148.265	148	89	9	4	36.96	8	3,074.68	100.00	7,104	31,656	16
mesh.274.469	274	192	10	2	259.55	5	7,200.52	150.00	15,618	43,682	19
mesh.70.120	70	46	6	4	4.54	6	374.20	50.00	1,470	12,431	6
mesh.74.129	74	45	7	5	4.02	7	28.01	40.00	1,110	7,695	4
KKT_lowt01	82	58	13	10	3.57	13	371.49	30.00	984	12,164	3

Table 10: Large G graphs from [31] and two-dimensional spinglass graphs, $m = (m_1, n - m_1)$

graph	n	m_1	ub	DNN		DNN+cuts					
				lb (rounded)	clocktime (s)	lb (rounded)	clocktime (s)	imp. %	# BQP cuts	# iter.	# outer loops
spinglass2pm_18	324	195	-222	-245	252.02	-227	7,200.07	7.35	12,960	35,081	8
spinglass2pm_20	400	280	-258	-267	735.60	-262	7,200.39	1.87	14,000	47,002	7
spinglass2pm_21	441	287	-294	-318	667.70	-304	7,200.25	4.40	15,435	41,725	7
spinglass2pm_22	484	291	-336	-369	831.91	-347	7,200.64	5.96	24,200	34,795	10
$G_{500,5}$	500	350	173	127	804.22	149	7,200.63	17.32	20,000	32,734	10
$G_{500,10}$	500	325	563	463	518.01	490	3,101.66	5.83	11,361	22,746	8
$G_{500,20}$	500	300	1,682	1,497	426.13	1,534	2,405.54	2.47	9,646	21,665	6
$G_{1000,5}$	1,000	650	387	260	5,048.12	271	7,200.38	4.23	8,000	26,970	2
$G_{1000,10}$	1,000	600	1,304	1,048	3,400.21	1,074	7,200.49	2.48	16,000	22,496	4
$G_{1000,20}$	1,000	700	2,844	2,503	4,952.74	2,514	7,200.66	0.44	12,000	26,158	3

References

[1] Man Shun Ang, Jianzhu Ma, Nianjun Liu, Kun Huang, and Yijie Wang. Fast projection onto the capped simplex with applications to sparse regression in bioinformatics. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9990–9999. Curran Associates, Inc., 2021.

[2] Miguel F. Anjos, Bissan Ghaddar, Lena Hupp, Frauke Liers, and Angelika Wiegele. Solving k-way graph partitioning problems to optimality: The impact of semidefinite relaxations and the bundle method. In Michael Jünger and Gerhard Reinelt, editors, *Facets of Combinatorial Optimization: Festschrift for Martin Grötschel*, pages 355–386. Springer Berlin Heidelberg, Berlin and Heidelberg, Germany, 2013. doi: 10.1007/978-3-642-38189-8_15.

[3] Michael Armbruster. *Branch-and-Cut for a Semidefinite Relaxation of Large-scale Minimum Bisection Problems*. PhD thesis, Technische Universität Chemnitz, Chemnitz, Germany, 2007.

[4] Michael Armbruster, Marzena Fügenschuh, Christoph Helmberg, and Alexander Martin. LP and SDP branch-and-cut algorithms for the minimum graph bisection problem: a computational comparison. *Math. Program. Comput.*, 4:275–306, 2012. doi: 10.1007/s12532-012-0040-5.

[5] Heinz H. Bauschke and Valentin R. Koch. Projection methods: Swiss army knives for solving feasibility and best approximation problems with halfspaces. In Simeon Reich and Alexander J. Zaslavski, editors, *Infinite Products of Operators and Their Applications*, volume 636, pages 1–40. CAMS and Bar-Ilan University, Providence, Rhode Island, and Ramat-Gan, Israel, 2015.

[6] Charles-Edmond Bichot and Patrick Siarry, editors. *Graph Partitioning*. ISTE Ltd, London, United Kingdom, 2011.

[7] Stephen P. Boyd, Neal Parikh, Eric King wah Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3:1–122, 2011. doi: 10.1561/2200000016.

[8] James P. Boyle and Richard L. Dykstra. A method for finding projections onto the intersection of convex sets in Hilbert spaces. In Richard Dykstra, Tim Robertson, and Farroll T. Wright, editors, *Advances in Order Restricted Statistical Inference*, volume 37, New York, NY, 1986. Springer.

[9] Lorenzo Brunetta, Michele Conforti, and Giovanni Rinaldi. A branch-and-cut algorithm for the equicut problem. *Math. Program.*, 78:243–263, 1997. doi: 10.1007/BF02614373.

[10] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in graph partitioning. In Lasse Kliemann and Peter Sanders, editors, *Algorithm Engineering: Selected Results and Surveys*, volume 9220. Springer International Publishing, Cham, Switzerland, 2016. doi: 10.1007/978-3-319-49487-6_4.

[11] Sam Burer and Dieter Vandenbussche. Solving lift-and-project relaxations of binary integer programs. *SIAM J. Optim.*, 16:726–750, 2006. doi: 10.1137/040609574.

[12] Sunil Chopra and Mendu R Rao. The partition problem. *Math. Program.*, 59:87–115, 1993. doi: 10.1007/BF01581239.

[13] Oliver Collins, Sam Dolinar, Robert McEliece, and Fabrizio Pollara. A VLSI decomposition of the de Bruijn graph. *J. ACM*, 39:931–948, 1992. doi: 10.1145/146585.146620.

[14] Edwin R. Van Dam and Renata Sotirov. Semidefinite programming and eigenvalue bounds for the graph partition problem. *Math. Program.*, 151:379–404, 2015. doi: 10.1007/s10107-014-0817-6.

[15] Wilm E. Donath and Alan J. Hoffman. Lower bounds for the partitioning of graphs. *IBM J. Res. Dev.*, 17:420–425, 1973. doi: 10.1142/9789812796936_0044.

[16] Andrew L. Dulmage and Nathan S. Mendelsohn. Matrices associated with the Hitchcock problem. *J. ACM*, 9:409–418, 1962.

[17] Richard L. Dykstra. An algorithm for restricted least squares regression. *J. Am. Stat. Assoc.*, 78:837–842, 1983. doi: 10.1080/01621459.1983.10477029.

[18] Andreas Eisenblätter. *Frequency Assignment in GSM Networks: Models, Heuristics, and Lower Bounds*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 2001.

[19] Rainer Feldmann, Burkhard Monien, Peter Mysliwietz, and Stefan Tschöke. A better upper bound on the bisection width of de bruijn networks. In Rüdiger Reischuk and Michel Morvan, editors, *STACS 97*, pages 511–522, Berlin and Heidelberg, Germany, 1997. Springer Berlin Heidelberg.

[20] Carlos Eduardo Ferreira, Alexander Martin, C Carvalho de Souza, Robert Weismantel, and Laurence A Wolsey. The node capacitated graph partitioning problem: a computational study. *Math. Program.*, 81:229–256, 1998. doi: 10.1007/BF01581107.

[21] Michel Fortin and Roland Glowinski, editors. *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*. Studies in Mathematics and Its Applications. North-Holland Publishing Comp., Amsterdam, The Netherlands, 1983.

[22] Norbert Gaffke and Rudolf Mathar. A cyclic projection algorithm via duality. *Metrika*, 36: 29–54, 1989. doi: 10.1007/BF02614077.

[23] Philippe Galinier and Jin-Kao Hao. Hybrid evolutionary algorithms for graph coloring. *J. Comb. Optim.*, 3:379–397, 1999. doi: 10.1023/A:1009823419804.

[24] Michael R. Garey, David S. Johnson, and Larry Stockmeyer. Some simplified NP-complete graph problems. *Theor. Comput. Sci.*, 1:237–267, 1976. doi: 10.1016/0304-3975(76)90059-1.

[25] Martin Grötschel and Yoshiko Wakabayashi. A cutting plane algorithm for a clustering problem. *Math. Programming*, 45:59–96, 1989. doi: 10.1007/BF01589097.

[26] William W. Hager, Dzung T. Phan, and Hongchao Zhang. An exact algorithm for graph partitioning. *Math. Program.*, 137:531–556, 2013. doi: 10.1007/s10107-011-0503-x.

[27] Christoph Helmberg. A cutting plane algorithm for large scale semidefinite relaxations. In Martin Grötschel, editor, *The Sharpest Cut: The Impact of Manfred Padberg and His Work*, pages 233–256. SIAM, Philadelphia, PA, 2004. doi: 10.1137/1.9780898718805.ch15.

[28] Roy Van Houte, Jesse Mulderij, Thomas Attema, Irina Chiscop, and Frank Phillipson. Mathematical formulation of quantum circuit design problems in networks of quantum computers. *Quantum Inf. Process*, 18, 2020. doi: 10.1007/s11128-020-02630-8.

[29] Hao Hu and Renata Sotirov. On solving the quadratic shortest path problem. *INFORMS J. Comput.*, 32:219–233, 2020. doi: 10.1287/ijoc.2018.0861.

[30] Hao Hu, Renata Sotirov, and Henry Wolkowicz. Facial reduction for symmetry reduced semidefinite doubly nonnegative programs. arXiv preprint, 2022.

[31] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning. *Oper. Res.*, 37:865–892, 1989. doi: 10.1287/opre.37.6.865.

[32] Stefan E. Karisch and Franz Rendl. Semidefinite programming and graph equipartition. In Panos M Pardalos and Henry Wolkowicz, editors, *Topics in Semidefinite and Interior-Point Methods*, pages 77–95. American Mathematical Soc., 1998.

[33] Stefan E. Karisch, Franz Rendl, and Jens Clausen. Solving graph bisection problems with semidefinite programming. *INFORMS J. Comput.*, 12:177–191, 2000. doi: 10.1287/ijoc.12.3. 177.12637.

[34] Etienne De Klerk, Dima Pasechnik, Renata Sotirov, and Cristian Dobre. On semidefinite programming relaxations of maximum k-section. *Math. Program.*, 136:253–278, 2012. doi: 10.1007/s10107-012-0603-2.

[35] Olga Kuryatnikova, Renata Sotirov, and Juan Vera. The maximum k-colorable subgraph problem and related problems. *INFORMS J. Comput.*, 34(1):656–669, 2021. doi: 10.1287/ijoc. 2021.1086.

[36] Xinxin Li, Ting Kei Pong, Hao Sun, and H. Wolkowicz. A strictly contractive Peaceman-Rachford splitting method for the doubly nonnegative relaxation of the minimum cut problem. *Comput. Optim. Appl.*, 78:853–891, 2021. doi: 10.1007/s10589-020-00261-4.

[37] Dirk A. Lorenz and Quoc Tran-Dinh. Non-stationary Douglas-Rachford and alternating direction method of multipliers: adaptive stepsizes and convergence. *Comput. Optim. Appl.*, 74: 67–92, 2019. doi: 10.1007/s10589-019-00106-9.

[38] Ngoc Hoang Anh Mai, Jean-Bernard Lasserre, Victor Magron, and Jie Wang. Exploiting constant trace property in large-scale polynomial optimization. arXiv preprint, 2020.

[39] Jean-François Marcotorchino. *Aggregation of Similarities in Automatic Classification (in French)*. PhD thesis, Université Paris VI, Paris, France, 1981.

[40] Frank De Meijer and Renata Sotirov. SDP-based bounds for the Quadratic Cycle Cover problem via cutting-plane Augmented Lagrangian methods and Reinforcement Learning. *INFORMS J. Comput.*, 33:1262–1276, 2021. doi: 10.1287/ijoc.2021.1075.

[41] Mosek ApS. The MOSEK optimization toolbox. *Version 9.1.*, 2019.

[42] Danilo E. Oliveira, Henry Wolkowicz, and Yangyang Xu. ADMM for the SDP relaxation of the QAP. *Math. Program. Comput.*, 10:631–658, 2018. doi: 10.1007/s12532-018-0148-3.

[43] Manfred Padberg. The boolean quadric polytope: some characteristics, facets and relatives. *Math. Program.*, 45:139–172, 1989. doi: 10.1007/BF01589101.

[44] Janez Povh, Franz Rendl, and Angelika Wiegele. A boundary point method to solve semidefinite programs. *Comput.*, 78:277–286, 2006. doi: 10.1007/s00607-006-0182-2.

[45] Franz Rendl and Renata Sotirov. The min-cut and vertex separator problem. *Comput. Optim. Appl.*, 69:159–187, 2018. doi: 10.1007/s10589-017-9943-4.

[46] Franz Rendl and Henry Wolkowicz. A projection technique for partitioning nodes of a graph. *Ann. Oper. Res.*, 58:155–179, 1995. doi: 10.1007/BF02032130.

[47] Franz Rendl, Renata Sotirov, and Christian Truden. Lower bounds for the bandwidth problem. *Comput. Oper. Res.*, 135:105422, 2021. doi: 10.1016/j.cor.2021.105422.

[48] Giovanni Rinaldi. Rudy: a generator for random graphs, 1998. URL `https://www-user. tu-chemnitz.de/~helmberg/sdp_software.html`. (accessed 16 November 2021).

[49] Renata Sotirov. SDP relaxations for some combinatorial optimization problems. In Miguel F. Anjos and Jean B. Lasserre, editors, *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 795–819. Springer US, Boston, MA, 2012.

[50] Renata Sotirov. An efficient semidefinite programming relaxation for the graph partition problem. *INFORMS J. Comput.*, 26:16–30, 2014. doi: 10.1287/ijoc.1120.0542.

[51] Renata Sotirov. Graph bisection revisited. *Ann. Oper. Res.*, 265:143–154, 2018. doi: 10.1007/s10479-017-2575-3.

[52] Cid Carvalho De Souza. *The graph equipartition problem: Optimal solutions, extensions and applications*. PhD thesis, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, 1993.

[53] Defeng Sun, Kim-Chuan Toh, Yancheng Yuan, and Xin-Yuan Zhao. SDPNAL+: A Matlab software for semidefinite programming with bound constraints (version 1.0). *Optim. Methods Softw.*, 35:87–115, 2020. doi: 10.1080/10556788.2019.1576176.

[54] Levent Tunçel. On the Slater condition for the SDP relaxations of nonconvex sets. *Oper. Res. Lett.*, 29:181–186, 2001. doi: 10.1016/S0167-6377(01)00093-1.

[55] Zaiwen Wen, Donald Goldfarb, and Wotao Yin. Alternating direction augmented Lagrangian methods for semidefinite programming. *Math. Program. Comput.*, 2:203–230, 2010. doi: 10.1007/s12532-010-0017-1.

[56] Angelika Wiegele and Shudian Zhao. SDP-based bounds for graph partition via extended ADMM. *Comput. Optim. Appl.*, 82:251–291, 2022. doi: 10.1007/s10589-022-00355-1.

[57] Henry Wolkowicz and Qing Zhao. Semidefinite programming relaxations for the graph partitioning problem. *Discrete Appl. Math.*, 96/97:461–479, 1999. doi: 10.1016/S0166-218X(99)00102-X.

[58] Alp Yurtsever, Joel A. Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming. *SIAM J. Math. Anal.*, 3:171–200, 2021. doi: 10.1137/19M1305045.

[59] Shudian Zhao. *Splitting into Pieces: Alternating Direction Methods of Multipliers and Graph Partitioning*. PhD thesis, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria, 2022.

A Projection onto polyhedral sets

One of the ingredients of the CP–ADMM is the orthogonal projection onto a polyhedral set. More precisely, the X-subproblem in (25b) involves a projection onto the set $\mathcal{X}$, where $\mathcal{X}$ is given in (12) and (22) for the k-EP and GBP, respectively. We focus here on the projector for the GBP. The projector for the k-EP, which has a simpler structure, can be obtained similarly.

Recall from (22) that the polyhedral set $\mathcal{X}_{BP}$ looks as follows:

$$\mathcal{X}_{BP} = \left\{ X = \begin{pmatrix} x^0 & (x^1)^\top & (x^2)^\top \\ x^1 & X^{11} & X^{12} \\ x^2 & X^{21} & X^{22} \end{pmatrix} \in \mathcal{S}^{2n+1} : \begin{matrix} \mathcal{G}_{\mathcal{J}}(X) = \mathbf{0}, \ X_{1,1} = 1, \ \mathrm{tr}(X^{ii}) = m_i, \ i \in [2], \\ \mathrm{diag}(X^{11}) + \mathrm{diag}(X^{22}) = \mathbf{1}_n, \ X\mathbf{u}_1 = \mathrm{diag}(X), \\ \mathbf{0} \le X \le \mathbf{J} \end{matrix} \right\}.$$

Let $\mathcal{P}_{\mathcal{X}_{BP}} : \mathcal{S}^{2n+1} \to \mathcal{S}^{2n+1}$ denote the projection onto $\mathcal{X}_{BP}$.

Observe that each constraint that defines $\mathcal{X}_{BP}$ either acts on the diagonal, first row, and first column of the matrix, or on the remaining entries. In the latter case, an entry X_{ij} is either bounded by 0 and 1 or equals 0 if $(i,j) \in \mathcal{J}$. These projections are very simple and are given by the operators T_{inner} and T_{box} in Table 11.

Next, we focus on the entries on the diagonal, first row, and first column of the orthogonal projection. Suppose $Y = \mathcal{P}_{\mathcal{X}_{BP}}(X)$ and let $y_1 := \mathrm{diag}(Y^{11})$ and $y_2 := \mathrm{diag}(Y^{22})$. Then y_1 and y_2 can be obtained via the following optimization problem:

$$\begin{aligned} \min_{y_1, y_2 \in \mathbb{R}^n} \quad & (y_1 - \mathrm{diag}(X^{11}))^\top (y_1 - \mathrm{diag}(X^{11})) + 2(y_1 - x^1)^\top (y_1 - x^1) \\ & + (y_2 - \mathrm{diag}(X^{22}))^\top (y_2 - \mathrm{diag}(X^{22})) + 2(y_2 - x^2)^\top (y_2 - x^2) \\ \text{s.t.} \quad & \mathbf{1}_n^\top y_1 = m_1, \ \mathbf{1}_n^\top y_2 = m_2, \ y_1 + y_2 = \mathbf{1}_n, \ y_1 \ge \mathbf{0}_n, \ y_2 \ge \mathbf{0}_n. \end{aligned} \quad (44)$$

Using basic algebra, one can show that the optimal y_1 to (44) is attained by the minimizer of the following optimization problem:

$$\min_{y_1 \in \mathbb{R}^n} \quad \left\| y_1 - \left(\frac{1}{6}(\text{diag}(X^{11}) - \text{diag}(X^{22})) + \frac{1}{3}(x^1 - x^2) + \frac{1}{2}\mathbf{1}_n \right) \right\|_2^2 \tag{45}$$
$$\text{s.t.} \quad \mathbf{1}_n^\top y_1 = m_1, \quad \mathbf{0}_n \le y_1 \le \mathbf{1}_n,$$

while the corresponding optimal y_2 to (44) is $y_2 = \mathbf{1}_n - y_1$. Observe that (45) is equivalent to a projection onto the capped simplex $\bar{\Delta}(m_1) = \{y \in \mathbb{R}^n : \mathbf{1}_n^\top y = m_1, \ \mathbf{0}_n \le y \le \mathbf{1}_n\}$. The projection onto $\bar{\Delta}(m_1)$ we denote by $\mathcal{P}_{\bar{\Delta}(m_1)} : \mathbb{R}^n \to \mathbb{R}^n$, which can be performed efficiently, see [1]. We define the operator T_{arrow}, see Table 11, to embed the optimal y_1 and y_2 in the space $\mathcal{S}^{2n+1}$.

	Operator		Description
T_{inner}	:	$\mathcal{S}^{2n+1} \to \mathcal{S}^{2n+1}$	$T_{\text{inner}}(X)_{ij} = 0$ if $i = 1$ or $j = 1$ or $i = j$ or $(i,j) \in \mathcal{J}$ and $T_{\text{inner}}(X)_{ij} = X_{ij}$ otherwise.
T_{box}	:	$\mathcal{S}^{2n+1} \to \mathcal{S}^{2n+1}$	$T_{\text{box}}(X)_{ij} = \min(\max(X_{ij}, 0), 1)$ for all (i,j).
T_{arrow}	:	$\mathbb{R}^n \to \mathcal{S}^{2n+1}$	$T_{\text{arrow}}(x) = \begin{pmatrix} 1 & x^\top & (\mathbf{1}_n - x)^\top \\ x & \text{Diag}(x) & \mathbf{0} \\ \mathbf{1}_n - x & \mathbf{0} & \text{Diag}(\mathbf{1}_n - x) \end{pmatrix}.$

Table 11: Overview of operators and their definitions.

Now, the projector $\mathcal{P}_{\mathcal{X}_{BP}}$ can be written out explicitly as follows:

$$\mathcal{P}_{\mathcal{X}_{BP}}\left(\begin{pmatrix} x^0 & (x^1)^\top & (x^2)^\top \\ x^1 & X^{11} & X^{12} \\ x^2 & X^{21} & X^{22} \end{pmatrix} \right) = T_{\text{box}}\left(T_{\text{inner}}\left(\begin{pmatrix} x^0 & (x^1)^\top & (x^2)^\top \\ x^1 & X^{11} & X^{12} \\ x^2 & X^{21} & X^{22} \end{pmatrix} \right) \right)$$

$$+ T_{\text{arrow}}\left(\mathcal{P}_{\bar{\Delta}(m_1)}\left(\frac{1}{6}(\text{diag}(X^{11}) - \text{diag}(X^{22})) + \frac{1}{3}(x^1 - x^2) + \frac{1}{2}\mathbf{1}_n \right) \right).$$

B Proofs for Cutting Plane Projectors

B.1 Proof of Lemma 4

The first statement is trivial. Now assume $M \notin \mathcal{H}_{ijl}^\Delta$, i.e., $M_{ij} + M_{il} > \frac{k-1}{k} + M_{jl}$. The projection of M onto $\mathcal{H}_{ijl}^\Delta$ is the solution to $\min_{\hat{M} \in \mathcal{S}^n} \left\{ \|\hat{M} - M\|_F^2 : \hat{M} \in \mathcal{H}_{ijl}^\Delta \right\}$. Since the inequality that describes $\mathcal{H}_{ijl}^\Delta$ only involves the submatrix induced by indices i, j and l, we can restrict ourselves to the following convex optimization problem:

$$\min_{\alpha,\beta,\gamma} \quad 2(\alpha - M_{ij})^2 + 2(\beta - M_{il})^2 + 2(\gamma - M_{jl})^2$$
$$\text{s.t.} \quad \alpha + \beta \le \frac{k-1}{k} + \gamma.$$

Let $\lambda \geq 0$ denote the Lagrange multiplier for the inequality, then the KKT conditions imply the following system:

$$\begin{cases} 4(\alpha - M_{ij}) + \lambda = 0 \\ 4(\beta - M_{il}) + \lambda = 0 \\ 4(\gamma - M_{jl}) - \lambda = 0 \\ \lambda(\alpha + \beta - \frac{k-1}{k} - \gamma) = 0 \\ \alpha + \beta \leq \frac{k-1}{k} + \gamma \\ \lambda \geq 0. \end{cases}$$

By complementarity, we have either $\lambda = 0$ or $\alpha + \beta - \frac{k-1}{k} - \gamma = 0$. The first case leads to the solution $\hat{M} = M$ that is a KKT-point if $M \in \mathcal{H}_{ijl}^{\Delta}$. If $M \notin \mathcal{H}_{ijl}^{\Delta}$, then $\alpha + \beta - \frac{k-1}{k} - \gamma = 0$. It follows from the first three conditions of the above system that we have

$$\alpha = M_{ij} - \frac{1}{4}\lambda, \quad \beta = M_{il} - \frac{1}{4}\lambda, \quad \gamma = M_{jl} + \frac{1}{4}\lambda.$$

Substitution into $\alpha + \beta - \frac{k-1}{k} - \gamma = 0$ yields

$$0 = M_{ij} - \frac{1}{4}\lambda + M_{il} - \frac{1}{4}\lambda - \frac{k-1}{k} - M_{jl} - \frac{1}{4}\lambda$$
$$\iff \quad \lambda = \frac{4}{3}\left(M_{ij} + M_{il} - M_{jl} - \frac{k-1}{k}\right).$$

Using this expression of the Lagrange multiplier gives the optimal values for α, β and γ:

$$\alpha = M_{ij} - \frac{1}{4} \cdot \frac{4}{3}\left(M_{ij} + M_{il} - M_{jl} - \frac{k-1}{k}\right) = \frac{2}{3}M_{ij} - \frac{1}{3}M_{il} + \frac{1}{3}M_{jl} + \frac{1}{3} - \frac{1}{3k},$$

$$\beta = M_{il} - \frac{1}{4} \cdot \frac{4}{3}\left(M_{ij} + M_{il} - M_{jl} - \frac{k-1}{k}\right) = -\frac{1}{3}M_{ij} + \frac{2}{3}M_{il} + \frac{1}{3}M_{jl} + \frac{1}{3} - \frac{1}{3k},$$

$$\gamma = M_{jl} + \frac{1}{4} \cdot \frac{4}{3}\left(M_{ij} + M_{il} - M_{jl} - \frac{k-1}{k}\right) = \frac{1}{3}M_{ij} + \frac{1}{3}M_{il} + \frac{2}{3}M_{jl} - \frac{1}{3} + \frac{1}{3k}.$$

Setting $\hat{M}_{ij} = \hat{M}_{ji} = \alpha$, $\hat{M}_{il} = \hat{M}_{li} = \beta$ and $\hat{M}_{jl} = \hat{M}_{lj} = \gamma$ gives the desired result. $\square$

B.2 Proof of Lemma 5

We use a similar technique as used in the proof of Lemma 4. The statement is trivial if $M \in \mathcal{H}_I^{IS}$. If $M \notin \mathcal{H}_I^{IS}$, it suffices to consider the following optimization problem:

$$\min_z \quad \sum_{i,j \in I, i<j} 2\left(z_{ij} - M_{ij}\right)^2$$

$$\text{s.t.} \quad \sum_{i,j \in I, i<j} z_{ij} \geq \frac{1-k}{2}.$$

Since this problem is convex, we restrict ourselves to solutions satisfying the KKT conditions. Let $\lambda \geq 0$ denote the Lagrange multiplier for the inequality. The KKT conditions read as follows:

$$\begin{cases} 4(z_{ij} - M_{ij}) - \lambda = 0 & \forall i,j \in I, \ i < j \\ \lambda\left(\frac{1-k}{2} - \sum_{i,j \in I, i<j} z_{ij}\right) = 0 \\ \sum_{i,j \in I, i<j} z_{ij} \geq \frac{1-k}{2} \\ \lambda \geq 0. \end{cases}$$

Complementarity implies that either $\lambda = 0$ or $\frac{1-k}{2} - \sum_{i,j\in I, i<j} z_{ij} = 0$. The former case leads to the solution $z_{ij} = M_{ij}$ for all $i,j \in I, i < j$, which is only a KKT point if $M \in \mathcal{H}_I^{IS}$. Since this is not the case, we have $\sum_{i,j\in I, i<j} z_{ij} = \frac{1-k}{2}$. It follows from the system's first equation that $z_{ij} = M_{ij} + \frac{1}{4}\lambda$ for all $i,j \in I, i < j$. Substitution into $\sum_{i,j\in I, i<j} z_{ij} = \frac{1-k}{2}$ yields:

$$
\sum_{i,j\in I, i<j}\left(M_{ij} + \frac{1}{4}\lambda\right) = \frac{1-k}{2} \quad\Longleftrightarrow\quad \binom{k+1}{2}\frac{1}{4}\lambda + \sum_{i,j\in I, i<j} M_{ij} = \frac{1-k}{2}
$$

$$
\Longleftrightarrow\quad \frac{k(k+1)}{2\cdot 4}\lambda = \frac{1-k}{2} - \sum_{i,j\in I, i<j} M_{ij}
$$

$$
\Longleftrightarrow\quad \lambda = \frac{8}{k(k+1)}\left(\frac{1-k}{2} - \sum_{i,j\in I, i<j} M_{ij}\right).
$$

Hence, this implies that for all $p,q \in I, p < q$ we have

$$
z_{pq} = M_{pq} + \frac{1}{4}\lambda = M_{pq} - \frac{k-1}{k(k+1)} - \frac{2}{k(k+1)}\sum_{i,j\in I, i<j} M_{ij}.
$$

Setting $\hat{M}_{pq} = z_{pq}$ for all $p,q \in I, p < q$, $\hat{M}_{pq} = z_{qp}$ for all $p,q \in I, p > q$, and $\hat{M}_{pq} = M_{pq}$ otherwise leads to the desired result. $\square$

B.3 Proof of Lemma 6

The projection of M onto $\mathcal{H}_{ijl}^{BQP}$ is the solution to $\min_{\hat{M}\in\mathcal{S}^{2n+1}}\left\{\|\hat{M} - M\|_F^2 \,:\, \hat{M} \in \mathcal{H}_{ijl}\right\}$. The inequality describing $\mathcal{H}_{ijl}^{BQP}$ only involves the pairs $(i,l), (j,l), (i,j)$ and (l,l). Since $\mathrm{diag}(X^{11}) = x^1, \mathrm{diag}(X^{22}) = x^2$ and $x^1 + x^2 = \mathbf{1}_n$ should be satisfied for $\hat{M}$, any change in (l,l) also has an effect on the pairs $(1,l)$, (l^*, l^*) and $(1, l^*)$, where l^* is the index corresponding to l in the diagonal block not containing l. Taking these pairs into account, we can restrict ourselves to the following convex optimization problem:

$$
\min_{\alpha,\beta,\gamma,\mu} \quad 2(\alpha - M_{il})^2 + 2(\beta - M_{jk})^2 + 2(\gamma - M_{ij})^2 + (\mu - M_{ll})^2 + 2(\mu - M_{1l})^2
$$
$$
+ (1 - \mu - M_{l^*l^*})^2 + 2(1 - \mu - M_{1l^*})^2
$$
$$
\text{s.t.} \quad \alpha + \beta \leq \gamma + \mu.
$$

Let $\lambda \geq 0$ denote the Lagrange multiplier for the inequality, then the KKT conditions imply the following system:

$$
\begin{cases}
4(\alpha - M_{il}) + \lambda = 0 \\
4(\beta - M_{jl}) + \lambda = 0 \\
4(\gamma - M_{ij}) - \lambda = 0 \\
2(\mu - M_{ll}) + 4(\mu - M_{1l}) + 2(\mu - 1 + M_{l^*l^*}) + 4(\mu - 1 + M_{1l^*}) - \lambda = 0 \\
\lambda(\alpha + \beta - \gamma - \mu) = 0 \\
\alpha + \beta \leq \gamma + \mu \\
\lambda \geq 0.
\end{cases}
$$

Complementarity implies that either $\alpha + \beta = \gamma + \mu$ or $\lambda = 0$. The latter case leads to the KKT-point $(\alpha, \beta, \gamma, \mu) = \left(M_{il}, M_{jl}, M_{ij}, \frac{1}{6}M_{ll} + \frac{1}{3}M_{1l} - \frac{1}{6}M_{l^*l^*} - \frac{1}{3}M_{1l^*} + \frac{1}{2}\right)$, which is optimal if $M_{il} + M_{jl} \leq M_{ij} + \frac{1}{6}M_{ll} + \frac{1}{3}M_{1l} - \frac{1}{6}M_{l^*l^*} - \frac{1}{3}M_{1l^*} + \frac{1}{2}$.

Now assume that $\lambda \neq 0$. Then $\alpha + \beta = \gamma + \mu$. The first four equalities of the KKT system can be rewritten as:

$$
\alpha = M_{il} - \frac{1}{4}\lambda, \quad \beta = M_{jl} - \frac{1}{4}\lambda, \quad \gamma = M_{ij} + \frac{1}{4}\lambda
$$

$$\mu = \frac{1}{6}M_{ll} + \frac{1}{3}M_{1l} - \frac{1}{6}M_{l^*l^*} - \frac{1}{3}M_{1l^*} + \frac{1}{2} + \frac{1}{12}\lambda$$

Substitution into $\alpha + \beta = \gamma + \mu$ yields

$$M_{il} - \frac{1}{4}\lambda + M_{jl} - \frac{1}{4}\lambda = M_{ij} + \frac{1}{4}\lambda + \frac{1}{6}M_{ll} + \frac{1}{3}M_{1l} - \frac{1}{6}M_{l^*l^*} - \frac{1}{3}M_{1l^*} + \frac{1}{2} + \frac{1}{12}\lambda$$

$$\iff \quad \lambda = \frac{12}{10}\left(M_{il} + M_{jl} - M_{ij} - \frac{1}{6}M_{ll} - \frac{1}{3}M_{1l} + \frac{1}{6}M_{l^*l^*} + \frac{1}{3}M_{1l^*} - \frac{1}{2}\right).$$

Substitution of this expression for the Lagrange multiplier into the remaining equalities provides the optimal values for α, β, γ and μ:

$$\alpha = \frac{7}{10}M_{il} - \frac{3}{10}M_{jl} + \frac{3}{10}M_{ij} + \frac{1}{20}M_{ll} + \frac{1}{10}M_{1l} - \frac{1}{20}M_{l^*l^*} - \frac{1}{10}M_{1l^*} + \frac{3}{20}$$

$$\beta = -\frac{3}{10}M_{il} + \frac{7}{10}M_{jl} + \frac{3}{10}M_{ij} + \frac{1}{20}M_{ll} + \frac{1}{10}M_{1l} - \frac{1}{20}M_{l^*l^*} - \frac{1}{10}M_{1l^*} + \frac{3}{20}$$

$$\gamma = \frac{3}{10}M_{il} + \frac{3}{10}M_{jl} + \frac{7}{10}M_{ij} - \frac{1}{20}M_{ll} - \frac{1}{10}M_{1l} + \frac{1}{20}M_{l^*l^*} + \frac{1}{10}M_{1l^*} - \frac{3}{20}$$

$$\mu = \frac{1}{10}M_{il} + \frac{1}{10}M_{jl} - \frac{1}{10}M_{ij} + \frac{3}{20}M_{ll} + \frac{3}{10}M_{1l} - \frac{3}{20}M_{l^*l^*} - \frac{3}{10}M_{1l^*} + \frac{9}{20}.$$

Setting $\hat{M}_{il} = \hat{M}_{li} = \alpha$, $\hat{M}_{jl} = \hat{M}_{lj} = \beta$, $\hat{M}_{ij} = \hat{M}_{ji} = \gamma$, $\hat{M}_{ll} = \hat{M}_{1l} = \hat{M}_{l1} = \mu$ and $\hat{M}_{l^*l^*} = \hat{M}_{1l^*} = \hat{M}_{l^*1} = 1 - \mu$ gives the final result. $\square$

C Additional numerical results

In this section we report additional numerical results. Table 12 serves to evaluate a quality of the DNN relaxation (13) with additional cuts for large graphs obtained after adding at most $3n$ cuts in each outer while-loop of Algorithm 1. In Table 3, Section 5 we report the results when the number of added cuts in each outer while-loop is at most $5n$. Our numerical results show that lower bounds might significantly improve when adding more cuts. Therefore, our final choice for adding cuts for large graphs is $5n$.

We furthermore give in Table 13 additional numerical results for the DNN relaxation (13) with additional cuts and $k = 2$ for (rather small) instances from the literature. All these instances have been considered in [26]. The first group of instances are grid graphs of Brunetta et al. [9]. These graphs are as follows.

- Planar grid instances: To represent instances of equicut on planar grid graphs we assign a weight from 1 to 10, drawn from a uniform distribution, to the edges of a $h \times k$ planar grid, and a 0 weight to the other edges. The names of those graphs are formed by the size followed by the letter 'g'.

- Toroidal grid instances: Same as planar grid instances but for toroidal grids. The names of those graphs are formed by the size followed by the letter 't'.

- Mixed grid instances: These are dense instances with all edges having a nonzero weight. The edges of a planar grid receive weights from 10 to 100 uniformly generated and all the other edges a weight from 1 to 10, also uniformly generated. The names of those graphs are formed by the size followed by the letter 'm'.

The second group of instances are randomly generated graphs from [26]: for a fixed density, the edges are assigned integer weights uniformly drawn from [1,10]. Graphs' names begin with 'v', 't', 'q', 'c', and 's'.

Table 13 also includes results on instances constructed with de Bruijn networks by [26], of which the original data arise in applications related to parallel computer architecture [13, 19]. Graphs'

names begin with 'db'. Finally, we test some instances from finite element meshes from [26], graphs' names begin with 'm'.

There are 64 instances in Table 13, and we prove optimality for 53 instances. The longest time required to compute a lower bound is 2.5 minutes, and computation of upper bounds is negligible.

Table 12: Large G and U graphs from [31] (Maxineq=$3n$)

graph	n	k	ub	DNN lb (rounded)	DNN clocktime (s)	DNN+cuts lb (rounded)	DNN+cuts clocktime (s)	imp. %	# tri. cuts	# ind. set cuts	# iter.	# outer loops
$G_{500,2.5}$	500	2	49	25	45.83	41	1,094.09	64.00	14,593	407	1,560	10
$G_{500,5}$	500	2	218	156	22.60	196	222.60	25.64	14,597	403	1,116	10
$G_{500,10}$	500	2	626	513	15.90	554	91.75	7.99	10,989	464	743	9
$G_{500,20}$	500	2	1,744	1,566	14.78	1,613	35.03	3.00	8,075	743	547	7
$G_{1000,2.5}$	1,000	2	102	45	421.80	72	7,206.50	60.00	29,796	204	1,979	10
$G_{1000,5}$	1,000	2	451	307	204.65	378	1,400.91	23.13	29,432	568	1,975	10
$G_{1000,10}$	1,000	2	1,367	1,113	152.35	1,179	495.80	5.93	21,285	799	1,329	9
$G_{1000,20}$	1,000	2	3,389	3,007	140.69	3,079	374.94	2.39	15,471	1,242	1,208	6
$U_{500,5}$	500	2	2	1	74.43	2	1,391.38	100.00	15,000	0	3,136	10
$U_{500,10}$	500	2	26	8	39.52	19	393.38	137.50	9,921	5,079	1,770	10
$U_{500,20}$	500	2	178	56	66.28	135	565.41	141.07	10,439	4,561	2,782	10
$U_{500,40}$	500	2	412	163	97.47	339	1,004.65	107.98	8,739	6,261	3,326	10
$U_{1000,5}$	1,000	2	1	0	725.58	0	4,498.22	0.00	30,000	0	6,792	10
$U_{1000,10}$	1,000	2	39	8	674.90	22	2,284.10	175.00	25,014	4,986	3,583	10
$U_{1000,20}$	1,000	2	222	51	213.67	133	1,840.46	160.78	21,093	8,907	2,827	10
$U_{1000,40}$	1,000	2	737	240	561.08	571	3,685.70	137.92	21,429	8,571	3,310	10

Table 13: Graphs considered in the paper of Hager, Phan and Zhang [26] (Maxineq=$3n$)

graph	n	k	ub	DNN lb (rounded)	DNN clocktime (s)	DNN+cuts lb (rounded)	DNN+cuts clocktime (s)	imp. %	# tri. cuts	# ind. set cuts	# iter.	# outer loops
10x2g	20	2	6	3	0.13	6	1.09	124.48	23	97	367	2
5x6g	30	2	19	12	0.04	19	4.34	72.20	281	169	581	5
2x16g	32	2	8	3	0.10	8	1.91	171.05	193	191	341	4
18x2g	36	2	6	3	0.19	6	1.35	149.36	146	178	251	3
2x19g	38	2	6	2	0.16	4	29.45	194.02	692	993	1,184	16
5x8g	40	2	18	12	0.09	18	0.80	59.65	91	149	254	2
3x14g	42	2	10	4	0.11	10	6.56	177.75	443	439	511	7
5x10g	50	2	22	12	0.20	22	3.31	97.19	460	290	479	5
6x10g	60	2	28	14	0.15	28	17.60	108.16	1,117	1,223	696	13
7x10g	70	2	23	11	0.27	23	16.96	110.78	1,081	1,229	792	11
4x5t	20	2	28	25	0.01	28	0.15	14.50	44	16	135	1
12x2t	24	2	9	5	0.11	9	1.34	103.58	134	82	373	3
6x5t	30	2	31	23	0.02	31	1.03	36.88	93	87	202	2
8x5t	40	2	33	21	0.05	33	1.81	62.79	348	132	277	4
23x2t	46	2	9	2	0.31	7	17.00	254.00	1,094	1,083	634	18

4x12t	48	2	24	11	0.18	24	4.41	118.42	539	325	446	6
5x10t	50	2	33	17	0.09	33	3.90	99.72	513	237	348	5
10x6t	60	2	43	20	0.08	42	126.51	115.77	1,412	928	1,537	13
7x10t	70	2	45	21	0.10	45	8.13	121.64	1,120	350	505	7
10x8t	80	2	43	24	0.15	43	62.01	83.34	2,263	2,150	890	19
2x10m	20	2	118	112	0.01	118	0.14	5.40	44	16	139	1
6x5m	30	2	270	259	0.03	270	0.60	4.59	144	36	219	2
2x17m	34	2	316	297	0.03	313	18.76	5.69	432	486	1,203	9
10x4m	40	2	436	424	0.02	436	0.91	2.89	156	84	200	2
5x10m	50	2	670	654	0.05	670	1.90	2.45	346	104	258	3
4x13m	52	2	721	695	0.05	721	4.41	3.82	459	477	349	6
13x4m	52	2	721	695	0.05	721	4.67	3.82	440	496	398	6
9x6m	54	2	792	762	0.07	792	2.13	3.96	410	238	299	4
10x6m	60	2	954	932	0.10	954	1.68	2.39	460	80	330	3
10x7m	70	2	1,288	1,259	0.14	1,288	3.36	2.36	624	216	372	4
v000	20	2	401	393	0.02	401	0.50	2.29	80	40	164	2
v090	20	2	21	19	0.02	21	0.74	14.62	57	63	190	2
t000	30	2	900	878	0.03	900	1.67	2.51	284	166	289	5
t050	30	2	397	371	0.06	397	3.49	7.28	343	197	449	6
t090	30	2	24	21	0.06	24	0.36	15.91	65	25	154	1
q000	40	2	1,606	1,578	0.03	1,606	0.92	1.80	329	151	247	4
q010	40	2	1,425	1,387	0.03	1,424	4.40	2.70	404	196	812	5
q020	40	2	1,238	1,203	0.02	1,238	4.25	2.96	487	233	657	6
q030	40	2	1,056	1,016	0.03	1,055	2.83	3.84	491	229	421	6
q080	40	2	199	169	0.04	199	5.31	18.03	521	319	542	7
q090	40	2	63	51	0.10	63	1.79	24.36	315	165	369	4
c030	50	2	1,658	1,593	0.05	1,655	4.88	3.92	725	325	529	7
c070	50	2	603	556	0.06	600	7.11	7.99	817	315	694	9
c080	50	2	368	328	0.08	367	22.48	11.94	967	353	1,124	9
c090	50	2	122	102	0.08	122	1.45	19.69	421	179	348	4
c290	52	2	123	97	0.06	123	2.71	27.58	543	237	381	5
c490	54	2	160	135	0.07	160	2.28	18.54	577	233	371	5
c690	56	2	177	150	0.11	177	2.79	18.60	623	217	415	5
c890	58	2	227	186	0.13	223	13.69	20.32	1,219	347	861	9
s090	60	2	238	201	0.09	238	5.98	18.68	886	374	539	7
db3	8	2	4	4	0.00	4	0.00	4.65	0	0	69	0
db5	32	2	10	7	0.02	10	1.41	45.26	194	94	349	3
db6	64	2	18	11	0.10	18	5.52	76.04	1,020	324	518	7
db7	128	2	30	16	0.40	30	149.44	98.16	3,650	958	910	12
m4	32	2	6	4	0.04	6	0.51	63.21	138	54	163	2
ma	54	2	2	1	0.11	2	0.26	169.34	132	30	181	1
me	60	2	3	2	0.14	3	0.40	149.76	90	90	171	1
m6	70	2	7	3	0.19	7	2.39	140.55	475	365	351	4
mb	74	2	4	2	0.23	4	5.17	264.83	626	706	446	6
mc	74	2	6	3	0.16	6	2.22	163.69	420	468	342	4

md	80	2	4	2	0.24	4	4.79	263.41	567	873	448	6
mf	90	2	4	2	0.31	4	5.15	287.21	915	705	437	6
m1	100	2	4	1	0.78	4	47.39	351.32	2,461	2,639	922	17
m8	148	2	9	3	2.33	7	49.85	160.59	2,639	3,146	864	14