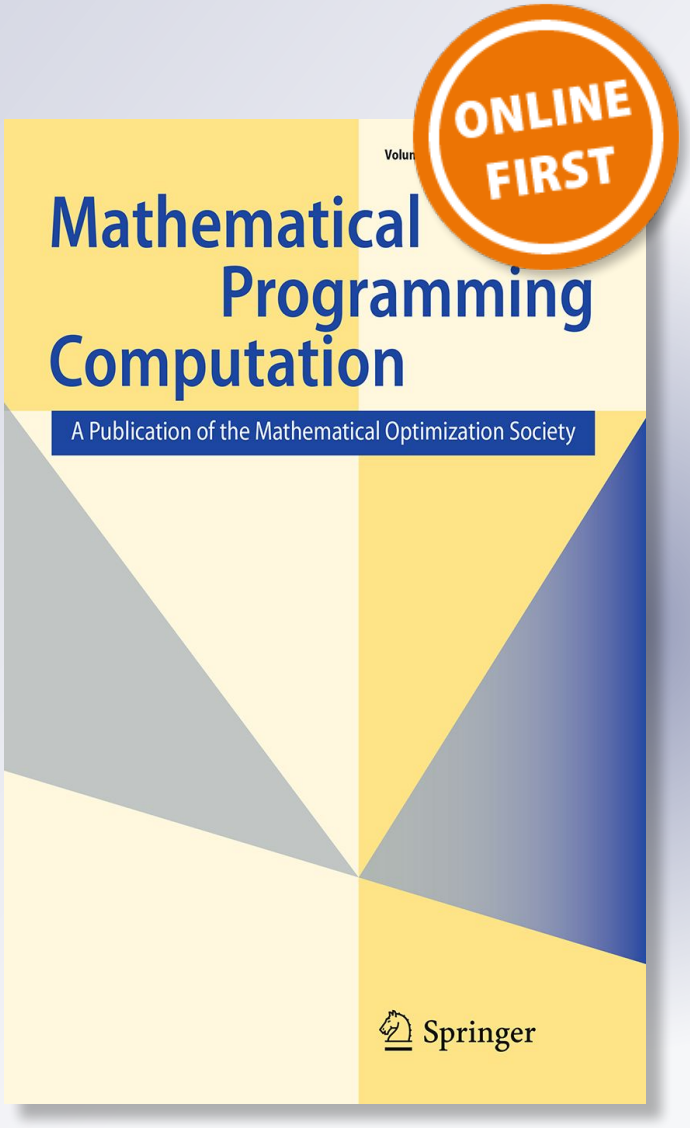
ADMM for the SDP relaxation of the QAP

Danilo Elias Oliveira, Henry Wolkowicz & Yangyang Xu

Mathematical Programming Computation
A Publication of the Mathematical Optimization Society

ISSN 1867-2949

Math. Prog. Comp.
DOI 10.1007/s12532-018-0148-3



Springer

Mathematical Programming Computation
https://doi.org/10.1007/s12532-018-0148-3

FULL LENGTH PAPER

ADMM for the SDP relaxation of the QAP

Danilo Elias Oliveira[1] · Henry Wolkowicz[1] · Yangyang Xu[2]

Received: 15 December 2015 / Accepted: 30 August 2018

Abstract

Semidefinite programming, **SDP**, relaxations have proven to be extremely strong for many hard discrete optimization problems. This is in particular true for the quadratic assignment problem, **QAP**, arguably one of the hardest NP-hard discrete optimization problems. There are several difficulties that arise in efficiently solving the **SDP** relaxation, e.g., increased dimension; inefficiency of the current primal–dual interior point solvers in terms of both time and accuracy; and difficulty and high expense in adding cutting plane constraints. We propose using the alternating direction method of multipliers **ADMM** in combination with facial reduction, **FR**, to solve the **SDP** relaxation. This first order approach allows for: inexpensive iterations, a method of cheaply obtaining low rank solutions; and a trivial way of exploiting the **FR** for adding cutting plane inequalities. In fact, we solve the doubly nonnegative, **DNN**, relaxation that includes *both* the **SDP** and *all* the nonnegativity constraints. When compared to current approaches and current best available bounds we obtain robustness, efficiency and improved bounds.

The code can be downloaded from the author's webpage https://xu-yangyang.github.io/ADMM_QAP/
The software that was reviewed as part of this submission was given the DOI (Digital Object Identifier) **10.5281/zenodo.1412139**.

This work is partially supported by NSERC and AFOSR. The first version of this paper appeared in optimization online, Dec. 16, 2015 and in arXiv:1512.05448, Dec. 17, 2015
Research supported by The Natural Sciences and Engineering Research Council of Canada and by AFOSR.
Research partly supported by NSF Grant DMS-1719549.

✉ Yangyang Xu
xuy21@rpi.edu

Henry Wolkowicz
hwolkowicz@uwaterloo.ca

[1] Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada

[2] Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, USA

Published online: 20 September 2018

Keywords Quadratic assignment problem · Semidefinite programming relaxation · Alternating direction method of multipliers · Facial reduction · Doubly nonnegative · Large scale

Mathematics Subject Classification 90C22 · 90B80 · 90C46 · 90C06 · 90-08

1 Introduction

The *quadratic assignment problem*, ***QAP***, in the trace formulation [11] is

$$(\mathbf{QAP}) \qquad p^* := \min_{X \in \Pi_n} \langle AXB - 2C, X \rangle, \tag{1.1}$$

where $A, B \in \mathbb{S}^n$ are real symmetric $n \times n$ matrices, C is a real $n \times n$ matrix, $\langle \cdot, \cdot \rangle$ denotes the trace inner product, i.e., $\langle Y, X \rangle = \text{trace}\, YX^\top$, and Π_n denotes the set of $n \times n$ permutation matrices. A typical application of the **QAP** is to assign n facilities to n locations while minimizing total cost. This total cost uses the *flow* A_{ij} between a pair of facilities i, j multiplied by the distance B_{st} between their assigned locations s, t, respectively. Included is the location cost C_{is} of placing facility i in location s. The **QAP** was first introduced as a model for analyzing the location of economic activities [17,18]. Further applications include: various layout problems, e.g., hospitals, airports, circuit boards, VLSI keyboards; bandwith minimization of a graph; image processing; molecular conformations in chemistry; scheduling; supply chains; manufacturing lines. Moreover, many well known discrete optimization problems are a special case of **QAP**, e.g., the traveling salesman problem and the maximum cut problem; see e.g., [3,20,21].

It is well known that the **QAP** is an NP-hard problem and that problems with size as moderate as $n = 30$ still remain difficult to solve, e.g., [1]. Solution techniques rely on efficiently calculating lower and upper bounds. An important tool for finding lower bounds is the work in [28] that provides a *semidefinite programmming* (**SDP**), relaxation of (1.1). In particular, this relaxation uses *facial reduction* (**FR**) to guarantee strict feasibility for both the relaxation and its dual and thus providing robustness; and **FR** greatly simplifies the constraints by making many of them redundant. The methods of choice for **SDP** are based on a *primal–dual interior-point, p–d i-p*, approach. These methods cannot solve large problems, have difficulty in obtaining high accuracy solutions, and cannot properly exploit sparsity. Moreover, it is very expensive to add on nonnegativity and other cutting plane constraints. The current state for finding bounds and solving **QAP** is given in e.g., [1,2,6,9,23,24].

1.1 Contributions

In this paper we apply the *alternating direction method of multipliers* (**ADMM**) for solving the facially reduced **SDP** relaxation of the **QAP** where we add additional elementwise nonnegativity constraints to the **SDP** constraints, i.e., an **ADMM** method for solving a *doubly nonnegative* (**DNN**) problem. Our model takes particular advantage of

the facial reduction by doubling the number of variables so that the **ADMM** approach can take advantage of separate simplified subproblems for the semidefinite constraints and the elementwise nonnegativity constraints. The recent papers [16,27] also present algorithms for solving the **DNN** relaxation of **QAP**, and their methods turn out to be very efficient for finding strong lower bounds of many **QAP** instances. However, they do not use the **FR** technique, and our lower bounds are stronger on many of our tested instances.

We compare our upper and lower bounds with: the best known results given in [24]; the best known bounds found at SDPLIB [7]; and with a p–d i-p methods based on the so-called HKM direction. We tested all symmetric instances from QAPLIB [7] with sizes up to $n = 100$. We find that our bounds strictly improve on the existing bounds in the literature and *provably solve many instances to optimality*. Moreover, we see that the **ADMM** method is significantly faster, and can often easily obtain medium-accuracy solutions, that are sufficient to provide strong lower bounds for **QAP**. This is partly due to the ability of obtaining low rank **SDP** solutions, as well as being able to solve the subproblems within the **ADMM** method fast and accurately. Finally, by exploiting low rank projections, we also obtain strong upper bounds.

1.2 Related works

A survey for various eigenvalue and **SDP** type lower bounds for **QAP** is given in [1]. Included are exact solution techniques as well. A copositive program, **CP**, is formulated in [23] and is shown to be equivalent to the **QAP**. Although the **CP** is convex, it is still intractable. Starting with the **CP**, several relaxations of **QAP** are presented in [23]. A review and a comparison with several other **SDP** relaxations is included.

Since the submission of our paper, we have become aware of the results in [15,16,27]. The work [15] studies optimization over permutation matrices. It shows that a penalized problem with the ℓ_0 seminorm can recover the solution to the original one if the penalty parameter is sufficiently large. Based on that observation, [15] uses an ℓ_p, $p \in (0, 1)$ seminorm to replace the ℓ_0 term. In addition, an ℓ_p regularization algorithm is used to find KKT points of a sequence of smoothed ℓ_p regularized problems. The algorithm is guaranteed to return a permutation matrix in a finite number of steps. Applied to the **QAP**, it will give a feasible solution and thus provide an upper bound.

General quadratic optimization with linear and also binary constraints is studied in [16]. This includes **QAP** as a special case. A Lagrangian-DNN relaxation is solved. Based on a formulation given in [23], lower bounds for some **QAP** instances are reported in [16]. It is demonstrated that the Lagrangian-DNN approach can be significantly faster than a Newton-CG **SDP** method (SDPNAL) [29], and comparable lower bounds are obtained. In contrast, our method yields the same or even better lower bounds on all the common tested instances except for `Char20c`, even though a small tolerance 10^{-12} was set in [16]. This is most likely due to the fact that **FR** was not used.

An improved version of SDPNAL, called SDPNAL+, is given in [27]. Using a good initial point found with an **ADMM** type method, SDPNAL+ applies a semismooth Newton-CG to subproblems in the augmented Lagrangian method framework. It is shown to be superior to several other **SDP** solvers and can solve many difficult **SDP**s from **QAP** instances to tolerance of order 10^{-6}. When compared to our approach, [27] obtains a better lower bound only on the instance `Tai25a`, and for many other tested instances, our results turn out to be strictly better. As noted above, this is possibly due to the use of **FR**.

More recently, [13] introduced a MATLAB based software package BBCPOP, that appears to improve further on [27] for solving the **DNN** relaxation of **QAP** . It applies the solver on the same relaxation used in [16]. It obtained a stronger lower bound than our approach on the single instance `Char20c`, while our lower bounds were strictly better on many tested instances.

We note that previous success of **ADMM** for solving **SDP** is presented in e.g., [26]. Convincing results on a few combinatorial optimization problems were obtained. A detailed survey for **ADMM** can be found in [5].

1.3 Outline

We continue in Sect. 2 with a new derivation of the facially reduced **SDP** relaxation of the **QAP** from [28]. This derivation is novel in that it directly includes the so-called *gangster constraints*. The new **ADMM** approach is presented in Sect. 3, where details of the **ADMM** subproblems are included, as well as details for obtaining the lower bounds from possibly inaccurate solutions of the **SDP** , and obtaining the upper bounds efficiently. Our numerics are presented in Sect. 4 with several tables. We conclude in Sect. 5.

2 A new derivation for the SDP relaxation

In this section we present a new derivation of the facially reduced **SDP** relaxation of the **QAP** obtained in [28]. The derivation is new in that the *gangster constraints* are obtained directly. We first briefly introduce **FR** and then derive the **SDP** relaxation from the dual of the Lagrangian dual.

2.1 Original FR for SDP relaxation of QAP

The **SDP** relaxation of the **QAP** in [28] begins with a set of quadratic constraints that represent the permutation matrices. Then, the Lagrangian relaxation (Lagrangian dual) is formed and shown to be equivalent to an **SDP**. The dual of this Lagrangian dual is then the **SDP** relaxation of the **QAP**. However, it is then shown in [28] that strict feasibility fails for this **SDP** relaxation. But one can find the barycenter, $\widehat{Y}$, of the feasible set and use the spectral decomposition

$$\widehat{Y} = [\widehat{V}\ \widehat{U}] \begin{bmatrix} D \succ 0 & 0 \\ 0 & 0 \end{bmatrix} [\widehat{V}\ \widehat{U}]^T$$

to obtain the facial reduction, minimal face $\mathcal{F}$, of all feasible Y for the **SDP** relaxation,

$$Y \in \mathcal{F} := \widehat{V}\mathbb{S}_+^{(n-1)^2+1}\widehat{V}^T \trianglelefteq \mathbb{S}^{n^2+1},$$

where $\trianglelefteq$ denotes face. Using the substitution $Y = \widehat{V}R\widehat{V}^T$ results in a smaller dimensional problem and, moreover, this substitution and the addition of the gangster constraints, makes many of the original constraints redundant. The result is an elegant, much simplified, stable **SDP** relaxation.

2.2 The new derivation

We now provide the new derivation of the facially reduced **SDP** relaxation in [28]. We start with the following equivalent quadratically constrained quadratic problem for **QAP**

$$\begin{aligned} &\min_X \langle AXB - 2C, X\rangle \\ &\text{s.t. } X_{ij}X_{ik} = 0,\ X_{ji}X_{ki} = 0,\ \forall i,\ \forall j \neq k, \\ &\quad X_{ij}^2 - X_{ij} = 0,\ \forall i, j, \\ &\quad \sum_{i=1}^n X_{ij}^2 - 1 = 0,\ \forall j,\ \sum_{j=1}^n X_{ij}^2 - 1 = 0,\ \forall i. \end{aligned} \tag{2.1}$$

Remark 2.1 Note that the quadratic orthogonality constraints $X^\top X = I$, $XX^\top = I$, and the linear row and column sum constraints $Xe = e$, $X^\top e = e$, can all be represented using linear combinations of those in (2.1). This observation avoids the need for adding all the redundant quadratic constraints and then removing redundant linear constraints in the **SDP**. Here e is the vector of all ones.

In addition, the first set of constraints, the elementwise orthogonality of the row and columns of X, are referred to as the *gangster constraints*. They are particularly strong constraints and enable many of the other constraints to be redundant. In fact, after the **FR** is done, many of these gangster constraints also become redundant.

The Lagrangian for (2.1) is

$$\begin{aligned} &\mathcal{L}_0(X, U, V, W, u, v) \\ &\quad = \langle AXB - 2C, X\rangle + \sum_{i=1}^n \sum_{j\neq k} U_{jk}^{(i)} X_{ij}X_{ik} + \sum_{i=1}^n \sum_{j\neq k} V_{jk}^{(i)} X_{ji}X_{ki} \\ &\qquad + \sum_{i,j} W_{ij}(X_{ij}^2 - X_{ij}) + \sum_{j=1}^n u_j \left(\sum_{i=1}^n X_{ij}^2 - 1\right) + \sum_{i=1}^n v_i \left(\sum_{j=1}^n X_{ij}^2 - 1\right). \end{aligned}$$

The dual problem is a maximization of the dual functional d_0,

$$\max\ d_0(U, V, W, u, v) := \min_X \mathcal{L}_0(X, U, V, W, u, v). \tag{2.2}$$

To simplify the dual problem, we homogenize $\mathcal{L}_0$ by multiplying the degree-one terms in X by a scalar variable x_0 and adding the single constraint $x_0^2 = 1$ to the dual functional. We add the additional dual variable w_0 and let

$$\begin{aligned}
&\mathcal{L}_1(X, x_0, U, V, W, w_0, u, v) \\
&\quad = \langle AXB - 2x_0C, X\rangle + \sum_{i=1}^{n}\sum_{j\neq k} U_{jk}^{(i)} X_{ij}X_{ik} + \sum_{i=1}^{n}\sum_{j\neq k} V_{jk}^{(i)} X_{ji}X_{ki} \\
&\qquad + \sum_{i,j} W_{ij}(X_{ij}^2 - x_0X_{ij}) + \sum_{j=1}^{n} u_j\left(\sum_{i=1}^{n} X_{ij}^2 - 1\right) \\
&\qquad + \sum_{i=1}^{n} v_i\left(\sum_{j=1}^{n} X_{ij}^2 - 1\right) + w_0(x_0^2 - 1).
\end{aligned}$$

This homogenization technique is the same as that in [28]. The new dual problem is

$$\max\ d_1(U, V, W, w_0, u, v) := \min_{X, x_0} \mathcal{L}_1(X, x_0, U, V, W, w_0, u, v). \tag{2.3}$$

Note that the dual functionals satisfy $d_1 \leq d_0$. Hence, our relaxation still yields a lower bound to (2.1). In fact, the relaxations give the same lower bound. This follows from strong duality of the trust region subproblem as shown in [28].

Let $x = \mathrm{vec}(X)$, $y = [x_0; x]$, and $w = \mathrm{vec}(W)$, where $\mathrm{vec}(X)$ denotes the columnwise vectorization of X. Then

$$\begin{aligned}
\mathcal{L}_1(X, x_0, U, V, W, w_0, u, v) = y^\top \big[&L_Q + \mathcal{B}_1(U) + \mathcal{B}_2(V) + \mathrm{Arrow}(w, w_0) \\
&+ \mathcal{K}_1(u) + \mathcal{K}_2(v)\big]\, y - e^\top(u + v) - w_0,
\end{aligned}$$

where

$$\begin{aligned}
&\mathcal{K}_1(u) = \mathrm{blkdiag}(0, u \otimes I), \quad \mathcal{K}_2(v) = \mathrm{blkdiag}(0, I \otimes v), \\
&\mathcal{B}_1(U) = \mathrm{blkdiag}(0, \widetilde{U}), \quad \mathcal{B}_2(V) = \mathrm{blkdiag}(0, \widetilde{V}), \\
&L_Q = \begin{bmatrix} 0 & -\mathrm{vec}(C)^\top \\ -\mathrm{vec}(C) & B \otimes A \end{bmatrix}, \quad \mathrm{Arrow}(w, w_0) = \begin{bmatrix} w_0 & -\frac{1}{2}w^\top \\ -\frac{1}{2}w & \mathrm{Diag}(w) \end{bmatrix}.
\end{aligned}$$

Here, $\otimes$ denotes the *Kronecker product*, and $\widetilde{U}$ and $\widetilde{V}$ are $n \times n$ block matrices. $\widetilde{U}$ has zero diagonal blocks and the (j, k)-th off-diagonal block is the diagonal matrix $\mathrm{Diag}(U_{jk}^{(1)}, \ldots, U_{jk}^{(n)})$, for all $j \neq k$. $\widetilde{V}$ has zero off-diagonal blocks and the i-th

diagonal block is $\begin{bmatrix} 0 & V_{12}^{(i)} & \cdots & V_{1n}^{(i)} \\ V_{21}^{(i)} & 0 & \cdots & V_{2n}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ V_{n1}^{(i)} & V_{n2}^{(i)} & \cdots & 0 \end{bmatrix}$. We use blkdiag$(A_1, A_2)$ to denote the block diagonal matrix with principal diagonal blocks A_1, A_2, cf. the same command in MATLAB. Hence, the dual problem (2.3) is equivalent to the **SDP**

$$\begin{aligned} &\max \ -e^\top(u+v) - w_0 \\ &\text{s.t. } L_Q + \mathcal{B}_1(U) + \mathcal{B}_2(V) + \text{Arrow}(w, w_0) + \mathcal{K}_1(u) + \mathcal{K}_2(v) \succeq 0. \end{aligned} \tag{2.4}$$

To obtain the **SDP** relaxation of (2.1), we further take the dual of (2.4). Before presenting the relaxation, we give a few definitions.

Definition 2.2 [block matrix $Y \in \mathbb{S}^{n^2+1}$] Given n^2 matrices $\widetilde{Y}_{ij}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n$ that satisfy $\widetilde{Y}_{ij} = \widetilde{Y}_{ji}^\top$, let $\bar{Y}$ be the $n \times n$ block matrix with $\widetilde{Y}_{ij}$ as the (i, j)-th block. We form the symmetric block matrix

$$Y = \begin{bmatrix} y_{00} & y_0^\top \\ y_0 & \bar{Y} \end{bmatrix}, \tag{2.5}$$

where y_{00} is a scalar, and y_0 is a vector in $\mathbb{R}^{n^2}$.

Definition 2.3 (*Gangster index set*) The *gangster index set, J* is defined to be the union of the top left index (00) and the set of indices $i < j$ in the matrix $\bar{Y}$ in (2.5) corresponding to:

1. the off-diagonal elements in the n diagonal blocks;
2. the diagonal elements in the off-diagonal blocks.

Definition 2.4 (*Gangster operator*) The *gangster operator*, $\mathcal{G}_J : \mathbb{S}^{n^2+1} \to \mathbb{S}^{n^2+1}$ is defined by

$$\mathcal{G}_J(Y)_{ij} = \begin{cases} Y_{ij} & \text{if } (i, j) \in J \text{ or } (j, i) \in J \\ 0 & \text{otherwise.} \end{cases}$$

By abuse of notation, we let the same symbol denote the projection onto $\mathbb{R}^{|J|}$, and thus for $y \in \mathbb{R}^{|J|}$, the adjoint yields $Y = \mathcal{G}_J^*(y) \in \mathbb{S}^{n^2+1}$ obtained by symmetrization and filling in the missing elements with zeros.

Now, taking the dual of (2.4), we have the **SDP** relaxation of (2.1):

$$\begin{aligned} \min \ & \langle L_Q, Y \rangle \\ \text{s.t. } & \mathcal{G}_J(Y) = E_{00} \\ & \text{diag}(\bar{Y}) = y_0 \\ & \text{trace}(\widetilde{Y}_{ii}) = 1, \ \forall i \\ & \textstyle\sum_{i=1}^n \widetilde{Y}_{ii} = I \\ & Y \succeq 0, \end{aligned} \tag{2.6}$$

where $E_{00} = e_0 e_0^T$ is the outer product of the first unit vector, the block matrix Y is defined in Definition 2.2 and the gangster index set J and the gangster operator $\mathcal{G}_J$ are defined in Definitions 2.3 and 2.4. Note that the variable Y in (2.6) is in a higher dimensional space compared to the original variable X in (2.1). This can be motivated from the *lifting* $Y = \begin{pmatrix} 1 \\ \mathrm{vec}(X) \end{pmatrix} \begin{pmatrix} 1 \\ \mathrm{vec}(X) \end{pmatrix}^\top$. We apply **ADMM** to an equivalent, more succinct, modification of (2.6). (See (3.1) and Theorem 3.1, below.)

Remark 2.5 If one more feasible quadratic constraint $q(X)$ can be added to (2.1), and $q(X)$ cannot be linearly represented by those in (2.1), the relaxation following the same derivation as above can be tighter. We conjecture that no more such $q(X)$ exists, and thus (2.6) is the tightest among all Lagrange dual relaxations from a quadratically constrained program like (2.1). However, this does not mean that more linear inequality constraints cannot be added, i.e., *linear cuts*.

2.3 Strict feasibility by FR

As above, let e be the vector of all ones of appropriate dimension, and let $V \in \mathbb{R}^{n\times(n-1)}$ be full column rank with $V^T e = 0$, and

$$\widehat{V} = \begin{bmatrix} 1 & 0 \\ \frac{1}{n}e & V \otimes V \end{bmatrix}. \tag{2.7}$$

FR is applied in [28] by using the substitution

$$Y = \widehat{V} R \widehat{V}^\top \in \mathbb{S}^{n^2+1}. \tag{2.8}$$

This way, it is shown that (2.6) is equivalent to

$$\begin{aligned} p_R^* := \min_R \ & \langle \widehat{V}^\top L_Q \widehat{V}, R \rangle \\ \text{s.t. } & \mathcal{G}_J(\widehat{V} R \widehat{V}^\top) = E_{00} \\ & R \succeq 0, \end{aligned} \tag{2.9}$$

a greatly simplified **SDP**. This simplification arising from **FR** allows for the **ADMM** to be applied efficiently for the **DNN** problem, i.e., we use the equivalence in (2.8) to relate Y, R and apply the gangster constraints and nonnegativity on Y while applying the semidefinite constraint on R.

Note that after **FR**, many constraints in (2.6) become redundant, and also we can remove redundant indices in J: the diagonal (zero) constraints in the last column of off-diagonal blocks and in the $(n-2, n-1)$ off-diagonal block. By abuse of notation, we use the same notation J and $\mathcal{G}_J$ after removing these indices. Another advantage of (2.9) is that strict feasibility holds, i.e., there exists a feasible $R \succ 0$, as shown in Lemma 2.6. In addition, strict feasibility holds for its dual problem, see Lemma 2.7. Both lemmas are from [28].

Lemma 2.6 *The matrix* $\widehat{R}$ *defined by*

$$\widehat{R} := \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \frac{1}{n^2(n-1)}\,(nI_{n-1} - E_{n-1}) \otimes (nI_{n-1} - E_{n-1}) \end{array}\right] \in \mathbb{S}^{(n-1)^2+1}_{++}$$

is (strictly) feasible for (2.9). □

We note that the gangster operator is self-adjoint, $\mathcal{G}_J^* = \mathcal{G}_J$. Therefore, the dual of (2.9) can be written as the following:

$$\begin{array}{rll} d_Y^* := \max\limits_{Y} & \langle E_{00}, Y\rangle & (= Y_{00}) \\ \text{s.t.} & \widehat{V}^\top \mathcal{G}_J(Y)\widehat{V} \preceq \widehat{V}^\top L_Q \widehat{V}. & \end{array} \tag{2.10}$$

Again by abuse of notation, using the same symbol twice, we get the two equivalent dual constraints:

$$\widehat{V}^\top \mathcal{G}_J(Y)\widehat{V} \preceq \widehat{V}^\top L_Q \widehat{V}; \qquad \widehat{V}^\top \mathcal{G}_J^*(y)\widehat{V} \preceq \widehat{V}^\top L_Q \widehat{V}.$$

As above, the dual variable for the first form is $Y \in \mathbb{S}^{n^2+1}$ and for the second form is $y \in \mathbb{R}^{|J|}$. We have used $\mathcal{G}^*$ for the second form to emphasize that only the first form is self-adjoint.

Lemma 2.7 *Define matrices* $\widehat{Y}$, $\widehat{Z}$, *with* $M > 0$ *sufficiently large, by*

$$\widehat{Y} := M \left[\begin{array}{c|c} n & 0 \\ \hline 0 & I_n \otimes (I_n - E_n) \end{array}\right] \in \mathbb{S}^{(n-1)^2+1}_{++}, \quad \widehat{Z} := \widehat{V}^\top L_Q \widehat{V} - \widehat{V}^\top \mathcal{G}_J(\widehat{Y})\widehat{V} \in \mathbb{S}^{(n-1)^2+1}_{++}.$$

Then they are (strictly) feasible variable and slack for (2.10). □

3 A new ADMM algorithm for the SDP relaxation

We can write (2.9) equivalently as

$$\min_{R,Y} \langle L_Q, Y\rangle \ \text{ s.t. } \mathcal{G}_J(Y) = E_{00},\ Y = \widehat{V} R \widehat{V}^\top,\ R \succeq 0. \tag{3.1}$$

The following theorem from [28] shows the equivalence between (2.6) and (3.1).

Theorem 3.1 *A matrix* Y *is feasible for* (2.6) if, and only if, *it is feasible for* (3.1). □

Therefore we can work with (3.1). The *augmented Lagrange* of (3.1) is

$$\mathcal{L}_A(R, Y, Z) = \langle L_Q, Y\rangle + \langle Z, Y - \widehat{V} R \widehat{V}^\top\rangle + \frac{\beta}{2}\|Y - \widehat{V} R \widehat{V}^\top\|_F^2. \tag{3.2}$$

Recall that (R, Y, Z) are the primal reduced, primal, and dual variables respectively. We denote (R, Y, Z) as the *current iterate*. Our new algorithm, an application of **ADMM**, uses the augmented Lagrangian in (3.2) and performs the following updates to obtain a new iterate (R_+, Y_+, Z_+):

$$R_+ = \arg\min_{R \in \mathbb{S}_+} \mathcal{L}_A(R, Y, Z), \tag{3.3a}$$

$$Y_+ = \arg\min_{Y \in \mathcal{P}_i} \mathcal{L}_A(R_+, Y, Z), \tag{3.3b}$$

$$Z_+ = Z + \gamma \cdot \beta(Y_+ - \widehat{V} R_+ \widehat{V}^\top), \tag{3.3c}$$

where the simplest case for the polyhedral constraints $\mathcal{P}_i$ is the linear manifold from the *gangster constraints*:

$$\mathcal{P}_1 = \{Y \in \mathbb{S}^{n^2+1} : \mathcal{G}_J(Y) = E_{00}\}.$$

We use this notation as we add additional simple polyhedral constraints. The second case is the polytope:

$$\mathcal{P}_2 = \mathcal{P}_1 \cap \{0 \le Y \le 1\}.$$

Let $\widehat{V}$ be normalized such that $\widehat{V}^\top \widehat{V} = I$. Then the R-subproblem can be explicitly solved by

$$\begin{aligned} R_+ &= \arg\min_{R \succeq 0} \langle Z, Y - \widehat{V} R \widehat{V}^\top \rangle + \tfrac{\beta}{2} \|Y - \widehat{V} R \widehat{V}^\top\|_F^2 \\ &= \arg\min_{R \succeq 0} \left\| Y - \widehat{V} R \widehat{V}^\top + \tfrac{1}{\beta} Z \right\|_F^2 \\ &= \arg\min_{R \succeq 0} \left\| R - \widehat{V}^\top \big(Y + \tfrac{1}{\beta} Z\big) \widehat{V} \right\|_F^2 \\ &= \mathcal{P}_{\mathbb{S}_+} \left(\widehat{V}^\top \big(Y + \tfrac{1}{\beta} Z\big) \widehat{V} \right), \end{aligned} \tag{3.4}$$

where $\mathbb{S}_+$ denotes the **SDP** cone, and $\mathcal{P}_{\mathbb{S}_+}$ is the orthogonal projection onto $\mathbb{S}_+$. For any symmetric matrix W, we have

$$\mathcal{P}_{\mathbb{S}_+}(W) = U_+ \Sigma_+ U_+^\top,$$

where (U_+, Σ_+) contains the positive eigenpairs of W; we let (U_-, Σ_-) be for the negative eigenpairs.

If $i = 1$ in (3.3b), the Y-subproblem also has a closed-form solution:

$$\begin{aligned} Y_+ &= \arg\min_{\mathcal{G}_J(Y) = E_{00}} \langle L_Q, Y \rangle + \langle Z, Y - \widehat{V} R_+ \widehat{V}^\top \rangle + \frac{\beta}{2} \|Y - \widehat{V} R_+ \widehat{V}^\top\|_F^2 \\ &= \arg\min_{\mathcal{G}_J(Y) = E_{00}} \left\| Y - \widehat{V} R_+ \widehat{V}^\top + \frac{L_Q + Z}{\beta} \right\|_F^2 \end{aligned}$$

$$= E_{00} + \mathcal{G}_{J^c}\left(\widehat{V}R_+\widehat{V}^\top - \frac{L_Q + Z}{\beta}\right). \tag{3.5}$$

One major advantage of using **ADMM** is that the complexity increases marginally when we add constraints to (2.9) and tighten the **SDP** relaxation. If $0 \le \widehat{V}R\widehat{V}^\top \le 1$ is added in (2.9), then we simply add the constraints $0 \le Y \le 1$ to (3.1). This yields the new problem

$$p^*_{RY} := \min_{R,Y}\{\langle L_Q, Y\rangle : \mathcal{G}_J(Y) = E_{00},\ 0 \le Y \le 1,\ Y = \widehat{V}R\widehat{V}^\top,\ R \succeq 0\}. \tag{3.6}$$

The **ADMM** for solving (3.6) has the same R-update and Z-update as those in (3.3). The Y-update is changed to

$$Y_+ = E_{00} + \min\left(1,\ \max\left(0,\ \mathcal{G}_{J^c}\big(\widehat{V}R_+\widehat{V}^\top - \frac{L_Q + Z}{\beta}\big)\right)\right). \tag{3.7}$$

The nonnegativity constraint means that the ≤ 1 constraint is redundant. But the inclusion makes the algorithm converge faster and avoid roundoff error. We emphasize again that it is the **FR** that allows for the splitting into polyhedral and semidefinite constraints. The update for R_+ is a nearest semidefinite problem and we can efficiently *cheat* and reduce the number of eigenvalues we allow to be positive by using the Eckart–Young Theorem, [10]. The update for Y_+ is a projection onto a simple polyhedral set and is very efficient and accurate.

3.1 Lower bound

If we solve (3.6) to high accuracy, we get a lower bound for the original **QAP**. However, the problem size of (3.6) can be extremely large, and it would be very expensive to obtain a highly accurate solution. In the following, we provide an inexpensive way to get a valid lower bound from the output of our algorithm that solves (3.6) to a moderate accuracy. Our method is to find a feasible solution of the dual problem of (3.6). The lemma below shows that any feasible dual solution provides a valid lower bound to (3.6) and thus the original **QAP**.

Lemma 3.2 (Lagrangian dual problem) *Let*

$$\mathcal{R} := \{R : R \succeq 0\}, \quad \mathcal{Y} := \{Y : \mathcal{G}_J(Y) = E_{00},\ 0 \le Y \le 1\}, \quad \mathcal{Z} := \{Z : \widehat{V}^\top Z\widehat{V} \preceq 0\}.$$

Define

$$g(Z) := \min_{Y\in\mathcal{Y}}\langle L_Q + Z, Y\rangle.$$

Then the dual problem of (3.6) *is* $d^*_Z := \max_{Z\in\mathcal{Z}} g(Z)$, *and the weak duality holds, i.e,* $d^*_Z \le p^*_{RY}$, *where* p^*_{RY} *is the optimal objective value of* (3.6).

Proof The dual problem of (3.6) can be derived as

$$\begin{aligned} d_Z^* &:= \max_Z \min_{R\in\mathcal{R}, Y\in\mathcal{Y}} \langle L_Q, Y\rangle + \langle Z, Y - \widehat{V} R \widehat{V}^\top\rangle \\ &= \max_Z \min_{Y\in\mathcal{Y}} \langle L_Q, Y\rangle + \langle Z, Y\rangle + \min_{R\in\mathcal{R}} \langle Z, -\widehat{V} R \widehat{V}^\top\rangle \\ &= \max_Z \min_{Y\in\mathcal{Y}} \langle L_Q, Y\rangle + \langle Z, Y\rangle + \min_{R\in\mathcal{R}} \langle \widehat{V}^\top Z \widehat{V}, -R\rangle \\ &= \max_{Z\in\mathcal{Z}} \min_{Y\in\mathcal{Y}} \langle L_Q + Z, Y\rangle \\ &= \max_{Z\in\mathcal{Z}} g(Z), \end{aligned}$$

where the fourth equality holds because if $Z \notin \mathcal{Z}$, then $\min_{R\in\mathcal{R}} \langle \widehat{V}^\top Z \widehat{V}, -R\rangle = -\infty$. Weak duality follows in the usual way by exchanging the max and min. □

For any $Z \in \mathcal{Z}$, we have $g(Z) \le d_Z^*$. Hence, from the above lemma, it follows that $g(Z)$ is a lower bound of (3.6) and thus of the original **QAP**. In addition, note that $g(Z)$ is easy to evaluate. Let $(R^{out}, Y^{out}, Z^{out})$ be the output of the **ADMM** for (3.6). We use the dual function value at the projected point $\mathcal{P}_{\mathcal{Z}}(Z^{out})$, namely $g\big(\mathcal{P}_{\mathcal{Z}}(Z^{out})\big)$, as the lower bound. Below we show how to get $\mathcal{P}_{\mathcal{Z}}(\widetilde{Z})$ for any symmetric matrix $\widetilde{Z}$.

Let $\widehat{V}_\perp$ be the orthonormal basis of the null space of $\widehat{V}$. Then $\bar{V} = (\widehat{V}, \widehat{V}_\perp)$ is an orthogonal matrix. Given any $Z \in \mathcal{Z}$, let $W = \bar{V}^\top Z \bar{V}$, and we write W into the 2×2 block matrix $\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22}\end{bmatrix}$. We have

$$Z \in \mathcal{Z} \Leftrightarrow \widehat{V}^\top Z \widehat{V} \preceq 0 \Leftrightarrow \widehat{V}^\top Z \widehat{V} = \widehat{V}^\top \bar{V} W \bar{V}^\top \widehat{V} = W_{11} \preceq 0.$$

Hence,

$$\mathcal{P}_{\mathcal{Z}}(\widetilde{Z}) = \arg\min_{Z\in\mathcal{Z}} \|Z - \widetilde{Z}\|_F^2 = \bar{V} W^* \bar{V}^\top,$$

where

$$\begin{aligned} W^* &= \arg\min_{W_{11}\preceq 0} \|\bar{V} W \bar{V}^\top - \widetilde{Z}\|_F^2 \\ &= \arg\min_{W_{11}\preceq 0} \|W - \bar{V}^\top \widetilde{Z} \bar{V}\|_F^2 \\ &= \begin{bmatrix} \mathcal{P}_{\mathbb{S}_-}(\widetilde{W}_{11}) & \widetilde{W}_{12} \\ \widetilde{W}_{21} & \widetilde{W}_{22}\end{bmatrix}. \end{aligned}$$

Here $\mathbb{S}_-$ denotes the negative semidefinite cone, and we have assumed $\bar{V}^\top \widetilde{Z} \bar{V} = \begin{bmatrix} \widetilde{W}_{11} & \widetilde{W}_{12} \\ \widetilde{W}_{21} & \widetilde{W}_{22}\end{bmatrix}$. Note that $\mathcal{P}_{\mathbb{S}_-}(W_{11}) = -\mathcal{P}_{\mathbb{S}_+}(-W_{11})$.

3.2 Upper bound from feasible solution

Let $(R^{out}, Y^{out}, Z^{out})$ be the output of the **ADMM** for (3.6). Assume the largest eigenvalue and the corresponding eigenvector of Y are λ and v, respectively. Then $\lambda vv^\top$ is a *best* rank-one approximation of Y. We let X^{out} be the square matrix reshaped from the second through the last elements of the first column of $\lambda vv^\top$. This is our approximation to (a multiple of) the optimal permutation matrix. Note that for any permutation matrix X we have trace $X^T X = n$. This implies that

$$\|X^{out} - X\|_F^2 = -2 \operatorname{trace} X^T X^{out} + \text{ constant}.$$

Thus to find the nearest permutation matrix to our approximation, we can take advantage of the Birkoff–von Neumann Theorem e.g., [4], that the permutation matrices are the extreme points of the doubly stochastic matrices. We only need to solve the *linear program*

$$\max_X \left\{ \langle X^{out}, X \rangle : Xe = e,\ X^\top e = e,\ X \geq 0 \right\} \tag{3.8}$$

by a simplex method that gives a basic feasible optimal solution, i.e., a permutation matrix.

3.3 Low-rank solution

Instead of finding a feasible solution with (3.8), we can directly get one by restricting R to a rank-one matrix, i.e., $\operatorname{rank}(R) = 1$ and $R \succeq 0$. With this constraint, the R-update can be modified to

$$R_+ = \mathcal{P}_{\mathbb{S}_+ \cap \mathcal{R}_1} \left(\widehat{V}^\top \left(Y + \frac{Z}{\beta} \right) \widehat{V} \right), \tag{3.9}$$

where $\mathcal{R}_1 = \{R : \operatorname{rank}(R) = 1\}$ denotes the set of rank-one matrices. For a symmetric matrix W with largest eigenvalue $\lambda > 0$ and corresponding eigenvector w, we have

$$\mathcal{P}_{\mathbb{S}_+ \cap \mathcal{R}_1} = \lambda w w^\top.$$

Despite of the nonconvexity of the rank-one constraint, we observed empirically that our algorithm almost always converged to a solution satisfying all the constraints in (3.6). Therefore, we obtained a permutation matrix from the lower bound.

3.4 Different choices for V, $\widehat{V}$

The matrix $\widehat{V}$ is essential in the steps of the algorithm, see e.g., (3.4). A sparse $\widehat{V}$ helps in the projection if one is using a sparse eigenvalue code. We have compared several. One is based on applying a QR algorithm to the original simple V from the definition

of $\widehat{V}$ in (2.7). The other two are based on the approach in [22] and we present the most successful here. The orthogonal V we use is

$$V = \begin{bmatrix} \begin{bmatrix} I_{\lfloor \frac{n}{2} \rfloor} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \\ 0_{(n-2\lfloor \frac{n}{2} \rfloor), \lfloor \frac{n}{2} \rfloor} \end{bmatrix} \end{bmatrix} \begin{bmatrix} \begin{bmatrix} I_{\lfloor \frac{n}{4} \rfloor} \otimes \frac{1}{2} \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \\ 0_{(n-4\lfloor \frac{n}{4} \rfloor), \lfloor \frac{n}{4} \rfloor} \end{bmatrix} \end{bmatrix} [\ldots] [\widehat{V}] \end{bmatrix}_{n \times n-1}$$

i.e., the block matrix consisting of t blocks formed from Kronecker products along with one block $\widehat{V}$ to complete the appropriate size so that $V^\top V = I_{n-1}$, $V^\top e = 0$. We take advantage of the 0, 1 structure of the Kronecker blocks and delay the scaling for the normalization till the end. The main work in the low rank projection part of the algorithm is to evaluate one (or a few) eigenvalues of $W = \widehat{V}^\top (Y + \frac{1}{\beta} Z) \widehat{V}$ to obtain the update R_+. Here

$$Y + \frac{1}{\beta} Z = \begin{bmatrix} \rho & w^\top \\ w & \bar{W} \end{bmatrix}.$$

We let

$$K := V \otimes V, \quad \alpha = 1/\sqrt{2}, \quad v = \frac{1}{\sqrt{2n}} e, \quad x = \begin{pmatrix} x_1 \\ \bar{x} \end{pmatrix}.$$

The structure for $\widehat{V}$ in (2.7) means that we can evaluate the product for Wx as

$$\begin{aligned} &\begin{bmatrix} \alpha & 0 \\ v & K \end{bmatrix}^\top \begin{bmatrix} \rho & w^\top \\ w & \bar{W} \end{bmatrix} \begin{bmatrix} \alpha & 0 \\ v & K \end{bmatrix} x = \begin{bmatrix} \alpha & 0 \\ v & K \end{bmatrix}^\top \begin{bmatrix} \rho & w^\top \\ w & \bar{W} \end{bmatrix} \begin{pmatrix} \alpha x_1 \\ x_1 v + K\bar{x} \end{pmatrix} \\ &= \begin{bmatrix} \alpha & v^\top \\ 0 & K^\top \end{bmatrix} \begin{pmatrix} \rho\alpha x_1 + w^\top (x_1 v + K\bar{x}) \\ \alpha x_1 w + \bar{W}(x_1 v + K\bar{x}) \end{pmatrix} \\ &= \begin{pmatrix} \rho\alpha^2 x_1 + \alpha w^\top (x_1 v + K\bar{x}) + v^\top \left(\alpha x_1 w + \bar{W}(x_1 v + K\bar{x})\right) \\ K^\top \left(\alpha x_1 w + \bar{W}(x_1 v + K\bar{x})\right) \end{pmatrix} \\ &= \begin{pmatrix} \rho\alpha^2 x_1 + \left(\alpha w^\top + v^\top \bar{W}\right)(x_1 v + K\bar{x}) + v^\top (\alpha x_1 w) \\ K^\top \left(\alpha x_1 w + \bar{W}(x_1 v + K\bar{x})\right) \end{pmatrix}. \end{aligned}$$

We emphasize that $V \otimes V = (\bar{V} \otimes \bar{V})(D \otimes D)^{-1}$, where $\bar{V}$ denotes the unscaled V, and D is the diagonal matrix of scale factors to obtain the orthogonality in V. Therefore, we can evaluate

$$K^\top \bar{W} K = (V \otimes V)^\top \bar{W} (V \otimes V) = (\bar{V} \otimes \bar{V})^\top \left[(D \otimes D)^{-1} \bar{W} (D \otimes D)^{-1}\right] (\bar{V} \otimes \bar{V}).$$

4 Numerical experiments

In this section we present the results of extensive numerical tests using our proposed methods. We used MATLAB version 2018a. All **QAP** symmetric instances from [7,8] with size up to $n = 100$ were used in our tests, while the instances bur26a–bur26h are not symmetric and not used. We divided them into two sets: QAPLIB instances I and QAPLIB instances II. All the instances were tested on an Intel Xeon Gold 6130 2.10 Ghz PC with 32 cores and 64 Gigabyte memory and running on 64-bit Ubuntu system.

4.1 Parameter settings

The parameters β and γ in the updates (3.3) play important roles on the speed of the **ADMM** method. Running the algorithm on a few small-sized problems, we heuristically set $\gamma = 1.618$ and $\beta = \frac{n}{3}$. Unless specified, the algorithm was terminated if it reached a maximum number of iterations or the following conditions hold in 5 consecutive iterations:

$$\max\left(\frac{\|Y^k - \widehat{V}R^k\widehat{V}\|_F}{\|Y^k\|_F},\ \beta\|Y^{k+1} - Y^k\|\right) \le \text{tol}, \tag{4.1}$$

where "tol" is a specified tolerance. In (4.1), the first term on the left hand side measures the residual of primal feasibility while the second term measures the dual feasibility; see [5, Sect. 3.3]. Although we have the rank-1 constraint, the stopping conditions in (4.1) were still met for most instances.

4.2 Results on QAPLIB instances I

Two stopping tolerances 10^{-5} and 10^{-12} were used for **ADMM** on QAPLIB instances I, and the maximum number of iterations was set to 40,000. Solving the **SDP** to the higher accuracy rarely improved the bounds. The results of lower and upper bounds are listed in Table 1; and the CPU times and iteration numbers of the algorithm for both tolerances are in Table 2. Failure of an algorithm is marked by -1111.

- In Table 1 the columns are:

  0. Instance name;
  1. Opt value: the globally optimal value of each instance, except for problem Tai30a, where optimality of the value is still not known;
  2. Bundle LowBnd: current best known lower bound from [24];
  3. HKM-FR LowBnd: the lower bound found using the p–d i-p approach with facial reduction and the HKM search direction and the code SDPT3 [25];[1]

[1] We do not include the times as they were much greater than those by the ADMM approach, e.g., hours instead of minutes and a day instead of an hour.

Table 1 Results of lower and upper bounds for each instance in **QAPLIB** Instances I

Problem	1 Opt value	2 Bundle [24] LowBnd	3 HKM-FR LowBnd	4 Tol5 ADMM LowBnd	5 Tol5 feas UpBnd	6 Tol12 ADMM LowBnd	7 Tol12 feas UpBnd	8 Tol5 ADMM %gap	9 ADMM Tol5 vs Bundle %Impr LowBnd
Esc16a	68	59	50	64	78	64	78	20.59	7.35
Esc16b	292	288	276	290	294	290	294	1.37	0.68
Esc16c	160	142	132	154	170	154	170	10.00	7.50
Esc16d	16	8	− 12	13	20	13	20	43.75	31.25
Esc16e	28	23	13	27	34	27	34	25.00	14.29
Esc16g	26	20	11	25	34	25	34	34.62	19.23
Esc16h	996	970	909	977	1012	977	1012	3.51	0.70
Esc16i	14	9	− 21	12	14	12	14	14.29	21.43
Esc16j	8	7	− 4	8	8	8	8	0.00	12.50
Had12	1652	1643	1641	1652	1652	1652	1652	0.00	0.54
Had14	2724	2715	2709	2724	2724	2724	2724	0.00	0.33
Had16	3720	3699	3678	3720	3720	3720	3720	0.00	0.56
Had18	5358	5317	5287	5358	5358	5358	5358	0.00	0.77
Had20	6922	6885	6848	6922	6930	6922	6930	0.12	0.53
Kra30a	88,900	77,647	− 1111	86,838	104050	86,838	105,900	19.36	10.34
Kra30b	91,420	81,156	− 1111	87,858	114950	87,858	114,950	29.63	7.33
Kra32	88,700	79,659	− 1111	85,775	111450	85,775	111,450	28.95	6.90
Nug12	578	557	530	568	654	568	654	14.88	1.90
Nug14	1014	992	960	1011	1022	1011	1022	1.08	1.87
Nug15	1150	1122	1071	1141	1196	1141	1196	4.78	1.65
Nug16a	1610	1570	1528	1600	1610	1600	1610	0.62	1.86
Nug16b	1240	1188	1139	1219	1438	1219	1438	17.66	2.50
Nug17	1732	1669	1622	1708	1756	1708	1756	2.77	2.25

Table 1 continued

Problem	1 Opt value	2 Bundle [24] LowBnd	3 HKM-FR LowBnd	4 Tol5 ADMM LowBnd	5 Tol5 feas UpBnd	6 Tol12 ADMM LowBnd	7 Tol12 feas UpBnd	8 Tol5 ADMM %gap	9 ADMM Tol5 vs Bundle %Impr LowBnd
Nug18	1930	1852	1802	1894	2160	1894	2160	13.78	2.18
Nug20	2570	2451	2386	2507	2732	2507	2732	8.75	2.18
Nug21	2438	2323	2386	2382	2672	2382	2672	11.89	2.42
Nug22	3596	3440	3396	3529	3856	3529	3856	9.09	2.47
Nug24	3488	3310	−1111	3402	3658	3402	3658	7.34	2.64
Nug25	3744	3535	−1111	3626	4052	3626	4052	11.38	2.43
Nug27	5234	4965	−1111	5130	5602	5130	5602	9.02	3.15
Nug28	5166	4901	−1111	5026	5534	5026	5534	9.83	2.42
Nug30	6124	5803	−1111	5950	6578	5950	6578	10.25	2.40
Rou12	235,528	223,680	221,161	235,528	235,528	235,528	235,528	0.00	5.03
Rou15	354,210	333,287	323,235	350,217	367,782	350,217	367,782	4.96	4.78
Rou20	725,522	663,833	642,856	695,181	765,390	695,181	765,390	9.68	4.32
Scr12	31,410	29,321	23,973	31,410	44,360	31,410	44,360	41.23	6.65
Scr15	51,140	48,836	42,204	51,140	58,304	51,140	58,304	14.01	4.51
Scr20	110,030	94,998	83,302	106,803	149,038	106,803	149,038	38.38	10.73
Tai12a	224,416	222,784	215,637	224,416	224,416	224,416	224,416	0.00	0.73
Tai15a	388,214	364,761	349,586	377,101	412,760	377,101	412,760	9.19	3.18
Tai17a	491,812	451,317	441,294	476,525	546,366	476,525	546,366	14.20	5.13
Tai20a	703,482	637,300	619,092	671,675	750,450	671,676	750,450	11.20	4.89
Tai25a	1,167, 256	1,041, 337	−1111	1,096, 657	1,271, 696	1,096, 658	1,271, 696	15.00	4.74
∗Tai30a	1,818, 146	1,652, 186	−1111	1,706, 871	1,942, 086	1,706, 872	1,942, 086	12.94	3.01
Tho30	149,936	136,059	−1111	143,576	169,958	143,576	169,958	17.60	5.01

Failure of an algorithm is marked by −1111, and the optimal value of the instance marked by ∗ is still unknown

Table 2 CPU times (in seconds) and iteration numbers by different approaches on **QAPLIB** Instances I. Failure of an algorithm is marked by −1111

	1 Tol5 cpusec HighRk	2 Tol5 cpusec LowRk	3 HKM cpuratio Tol 9	4 Tol5 iterations HighRk	5 Tol5 iterations LowRk	6 Tol12 iterations HighRk	7 Tol12 residual HighRk	8 Tol12 iterations LowRk
Esc16a	20.14	2.64	9.37	2053	280	7309	9.87e−13	305
Esc16b	3.10	2.93	8.08	338	311	641	3.94e−13	334
Esc16c	8.44	3.68	4.88	961	403	3751	9.69e−13	592
Esc16d	17.39	2.18	10.22	1889	236	7812	9.87e−13	270
Esc16e	24.04	2.63	8.79	2719	288	11784	9.93e−13	310
Esc16g	33.54	2.61	8.63	3839	285	9096	9.87e−13	304
Esc16h	4.01	2.73	10.60	433	300	886	8.47e−13	354
Esc16i	100.79	2.26	8.76	11653	290	27,106	9.96e−13	323
Esc16j	56.90	2.67	7.93	6898	306	29,743	9.95e−13	338
Had12	8.39	0.53	5.91	2682	157	2845	8.64e−13	178
Had14	23.07	0.99	10.46	3919	169	4747	2.35e−13	181
Had16	111.92	1.88	12.51	14,179	210	14,362	6.80e−13	228
Had18	268.58	3.57	13.28	18,068	259	40,000	2.07e−06	271
Had20	196.70	6.17	14.53	9038	309	40,000	5.55e−07	321
Kra30a	988.47	62.61	−1111	8466	632	40,000	2.08e−07	654
Kra30b	1481.32	63.31	−1111	12,882	623	40,000	8.73e−07	645
Kra32	1355.11	92.43	−1111	9020	720	40,000	5.28e−07	737
Nug12	22.27	0.53	5.93	5813	146	40,000	3.82e−09	163
Nug14	49.76	1.01	8.43	7667	167	40,000	2.94e−07	186
Nug15	53.68	1.49	7.79	6547	200	40,000	2.11e−07	221
Nug16a	117.57	1.76	12.24	11,591	193	40,000	1.46e−06	208
Nug16b	62.72	1.98	11.83	6410	207	40,000	5.87e−10	234
Nug17	135.80	2.31	13.13	10,727	204	40,000	9.12e−07	215
Nug18	250.85	3.22	15.23	15,862	226	40,000	1.79e−06	240
Nug20	238.68	5.82	14.35	9786	276	40,000	4.55e−07	289
Nug21	651.15	8.27	14.95	22,465	322	40,000	3.62e−06	340
Nug22	942.50	9.84	13.90	27,839	325	40,000	5.69e−06	338
Nug24	572.04	13.47	−1111	12,148	335	40,000	7.55e−07	346
Nug25	1308.41	18.38	−1111	24,051	375	40,000	5.05e−06	386
Nug27	1875.89	30.54	−1111	25,201	454	40,000	4.16e−06	465
Nug28	1658.48	34.50	−1111	18,417	447	40,000	2.73e−06	461
Nug30	2584.42	48.92	−1111	22,613	469	40,000	3.06e−06	478
Rou12	23.19	0.44	6.90	6327	127	6360	2.02e−13	142
Rou15	19.00	1.27	9.46	2219	170	19,769	6.08e−13	184
Rou20	88.20	5.60	16.08	3684	263	40,000	2.08e−07	275

Table 2 continued

	1 Tol5 cpusec HighRk	2 Tol5 cpusec LowRk	3 HKM cpuratio Tol 9	4 Tol5 iterations HighRk	5 Tol5 iterations LowRk	6 Tol12 iterations HighRk	7 Tol12 residual HighRk	8 Tol12 iterations LowRk
Scr12	3.71	0.48	5.79	1135	142	2878	6.65e−13	160
Scr15	8.06	1.14	10.75	1061	158	2023	8.11e−13	176
Scr20	858.08	5.94	17.96	34,679	264	40,000	7.68e−06	276
Tai12a	1.56	0.50	6.70	421	127	454	1.38e−13	145
Tai15a	17.01	1.22	10.34	1955	157	29,673	5.41e−13	170
Tai17a	39.60	2.31	12.04	2997	216	22,276	7.29e−13	234
Tai20a	66.02	5.62	15.85	2755	252	40,000	1.72e−08	267
Tai25a	128.14	17.20	−1111	2244	350	12,809	6.33e−13	362
Tai30a	433.54	55.82	−1111	3698	527	39,288	3.74e−13	539
Tho30	2045.32	51.37	−1111	17,854	522	40,000	2.23e−06	533

4. Tol5 ADMM LowBnd: the lower bound found by running **ADMM** without the rank-1 constraint, with the tolerance 10^{-5}, and evaluating the dual objective using the approach in Sect. 3.1;
5. Tol5 feas UpBnd: the stronger upper bound found by running **ADMM** with the rank-1 constraint and tolerance 10^{-5}, and also by running **ADMM** without the rank-1 constraint, with tolerance 10^{-5}, and then using the approach in Sect. 3.2;
6. Tol12 ADMM LowBnd: the lower bound found by running **ADMM** without rank-1 constraint to the tolerance 10^{-12} and then evaluating the dual objective through the approach discussed in Sect. 3.1;
7. Tol12 feas UpBnd: the stronger upper bound found by **ADMM** with the rank-1 constraint and tolerance 10^{-12}, and also **ADMM** without the rank-1 constraint with tolerance 10^{-12} and then using Sect. 3.2;
8. Tol5 ADMM % gap: the percentage gap between the lower and upper bounds found by our proposed approach with tolerance 10^{-5};
9. ADMM Tol5 vs Boundle %Impr LowBnd: the percentage improvement by our proposed approach with tolerance 10^{-5} over the current best known lower bound from [24].

Remark 4.1 (Table 1) From column 9, we see that our approach improves the currently best-known bounds for every instance. In addition, we have *provably* found the global optimal solution for the seven instances:

Esc16j, Had12, Had14, Had16, Had18, Rou12, Tai12a.

This is mainly due to the inclusion of all the nonnegativity constraints and the projection onto [0, 1], all with essentially zero extra computational cost, see (3.7). Note that adding the nonnegativity constraints would be too expensive within an interior point approach. In addition, the bounds rarely improved when using the smaller tolerance 10^{-12}.

Table 3 Results of lower and upper bounds, iteration numbers, and also CPU times (in seconds) by **ADMM** for each instance in **QAPLIB** Instances II with size no larger than 64. Optimal values of the instances marked by $*$ are still unknown

Problem	1. opt value	2. ADMM LowBnd	3. feas UpBnd	4. $ADMM$ $\%gap$	5 Tol5 cpusec HighRk	6 Tol5 cpusec LowRk	7 ADMM iterations HighRk	8 ADMM iterations LowRk
Chr12a	9552	9552	9552	0.00	6.53e+01	4.08e−01	21,061	117
Chr12b	9742	9742	9742	0.00	3.32e+01	4.11e−01	10,592	119
Chr12c	11156	11,156	11,156	0.00	7.42e+01	3.96e−01	23,982	115
Chr15a	9896	9896	9896	0.00	2.07e+02	1.28e+00	31,937	173
Chr15b	7990	7990	7990	0.00	2.69e+01	9.84e−01	3976	133
Chr15c	9504	9504	9504	0.00	1.54e+01	1.06e+00	2192	147
Chr18a	11,098	11,098	11,098	0.00	4.94e+02	2.86e+00	40,000	198
Chr18b	1534	1534	2264	32.24	5.72e+01	3.08e+00	3843	243
Chr20a	2192	2192	2192	0.00	7.40e+02	4.31e+00	40,000	217
Chr20b	2298	2298	2298	0.00	1.42e+02	5.31e+00	6355	243
Chr20c	14,142	14,139	14,142	0.02	7.28e+02	5.03e+00	40,000	232
Chr22a	6156	6156	6156	0.00	4.02e+02	9.37e+00	14,051	310
Chr22b	6194	6194	6194	0.00	3.80e+02	9.45e+00	11,418	304
Chr25a	3796	3796	3796	0.00	3.06e+02	1.70e+01	6164	355
Els19	17,212,548	17,209,789	17,212,548	0.02	6.17e+02	4.48e+00	40,000	269
Esc16f	0	0	0	0.00	3.22e+02	3.39e+02	40,000	40,000
Esc32a	130	104	168	38.10	2.89e+03	9.16e+01	20,398	700
Esc32b	168	132	264	50.00	2.52e+03	8.31e+01	17,920	658
Esc32c	642	616	686	10.20	4.48e+02	1.01e+02	3177	780
Esc32d	200	191	228	16.23	8.68e+02	1.09e+02	6334	825

Table 3 continued

Problem	1. opt value	2. ADMM LowBnd	3. feas UpBnd	4. $ADMM\ \%gap$	5 Tol5 cpusec HighRk	6 Tol5 cpusec LowRk	7 ADMM iterations HighRk	8 ADMM iterations LowRk
Esc32e	2	2	2	0.00	1.81e+03	1.05e+02	13,040	836
Esc32f	2	2	2	0.00	1.80e+03	1.07e+02	13,040	836
Esc32g	6	6	8	25.00	6.04e+02	1.06e+02	4405	855
Esc32h	438	425	482	11.83	3.02e+03	1.00e+02	21,515	795
∗Sko42	15,812	15,335	17,086	10.25	1.06e+04	3.87e+02	21,013	911
∗Sko49	23,386	22,653	25,076	9.66	3.03e+04	1.18e+03	28,771	1316
∗Sko56	34,458	33,390	36,580	8.72	3.90e+04	2.68e+03	21,106	1664
Ste36a	9526	9259	13,866	33.23	1.02e+04	1.87e+02	40,000	851
Ste36b	15,852	15,668	25,878	39.45	1.01e+04	1.56e+02	40,000	700
Ste36c	8,239,110	8,134,720	11,152,926	27.06	1.01e+04	1.69e+02	40,000	798
∗Tai35a	2,422,002	2,216,645	2,599,924	14.74	7.40e+02	1.33e+02	3225	661
∗Tai40a	3,139,370	2,843,312	3,392,692	16.19	1.94e+03	2.99e+02	4665	852
∗Tai50a	4,938,796	4,390,976	5,332,790	17.66	6.36e+03	1.33e+03	5393	1348
∗Tho40	240,516	226,522	269,452	15.93	8.52e+03	2.90e+02	21,131	828
∗Wil50	48,816	48,125	50,040	3.83	1.73e+04	1.43e+03	15,370	1473

Table 4 Results of lower and upper bounds and also CPU times (in seconds) by **ADMM** for each instance in **QAPLIB** Instances II with size at least 64. Optimal values of the instances marked by ∗ are still unknown

Problem	1. opt value	2. ADMM LowBnd	3. feas UpBnd	4. ADMM %gap	5 Tol5 cpusec HighRk	6 Tol5 cpusec LowRk
Esc64a	116	98	120	18.33	1.64e+04	1.11e+04
∗Sko64	48,498	46,888	50,840	7.77	1.56e+04	1.13e+04
∗Sko72	66,256	64,205	70,672	9.15	3.01e+04	2.07e+04
∗Sko81	90,998	87,756	96,456	9.02	5.94e+04	3.77e+04
∗Sko90	115,534	111,300	121,390	8.31	9.32e+04	6.72e+04
∗Sko100a	152,002	145,775	160,794	9.34	1.38e+05	9.37e+04
∗Sko100b	153,890	147,332	162,004	9.06	1.38e+05	9.45e+04
∗Sko100c	147,862	142,018	156,230	9.10	1.38e+05	9.46e+04
∗Sko100d	149,576	143,205	157,100	8.84	1.39e+05	9.53e+04
∗Sko100e	149,150	142,977	155,858	8.26	1.38e+05	9.51e+04
∗Sko100f	149,036	142,413	156,088	8.76	1.40e+05	9.70e+04
∗Tai60a	7,205,962	6,319,630	7,759,332	18.55	1.34e+04	1.01e+04
Tai64c	1,855,928	1,809,370	1,917,484	5.64	1.65e+04	1.14e+04
∗Tai80a	13,499,184	11,613,474	14,618,694	20.56	5.17e+04	3.08e+04
∗Tai100	21,052,466	17,704,527	22,641,778	21.81	1.53e+05	9.33e+04
∗Wil100	273,038	267,469	278,898	4.10	1.41e+05	9.67e+04

- In Table 2 the columns are:

  0. Instance name;
  1. Tol5 cpusec HighRk: CPU times (in seconds) of **ADMM** without the rank-1 constraint and with tolerance 10^{-5};
  2. Tol5 cpusec LowRk: CPU times (in seconds) of **ADMM** with the rank-1 constraint and with tolerance 10^{-5};
  3. HKM cpuratio Tol 9: the ratio between the CPU times by the p–d i-p approach and **ADMM** without the rank-1 constraint and with tolerance 10^{-5};
  4. Tol5 iterations HighRk: iteration numbers of **ADMM** without the rank-1 constraint and with tolerance 10^{-5};
  5. Tol5 iterations LowRk: iteration numbers of **ADMM** with the rank-1 constraint and with tolerance 10^{-5};
  6. Tol12 iterations HighRk: iteration numbers of **ADMM** without the rank-1 constraint and with tolerance 10^{-12};
  7. Tol12 residual HighRk: residual of the output measured as in (4.1) of **ADMM** without the rank-1 constraint and with tolerance 10^{-12};
  8. ADMM Tol12 iterations LowRk: the iteration numbers of **ADMM** with the rank-1 constraint and with tolerance 10^{-12}.

Remark 4.2 (Table 2) We see that **ADMM** with rank-1 constraint is much faster than that without the rank-1 constraint to reach the same tolerance. In addition, we notice

Table 5 Lower and upper bounds by **ADMM** for solving **SDP** relaxation with or without the restriction $0 \leq Y \leq 1$ on certain instances in **QAPLIB** Instances I

Problem	ADMM feas. Y as in (3.7)				ADMM feas. Y as in (3.5)			
	LowBnd	UpBnd	Iter.HighRk	Iter.LowRk	LowBnd	UpBnd	Iter.HighRk	Iter.LowRk
Kra30a	86,838	104,050	8466	632	78,687	1,04,050	25,974	40,000
Kra30b	87,858	114,950	12,882	623	79,510	108,550	21,248	40,000
Kra32	85,775	111,450	9020	720	77,130	105,500	3904	40,000
Nug24	3402	3658	12,148	335	3235	3844	4629	39,616
Nug25	3626	4052	24,051	375	3454	4078	8974	40,000
Nug27	5130	5602	25,201	454	4922	5708	20,763	40,000
Nug28	5026	5534	18,417	447	4813	5466	15,916	40,000
Nug30	5950	6578	22,613	469	5694	6630	10,524	40,000
Tai30a	1,706,871	1,942,086	3698	527	1,578,074	1,963,808	1072	40,000
Tho30	143,576	169,958	17,854	522	136,004	170,390	34,289	40,000

Table 6 Lower and upper bounds by **ADMM** for solving **SDP** relaxation with or without the restriction $0 \leq Y \leq 1$ on certain instances in **QAPLIB** Instances II

Problem	ADMM feas. Y as in (3.7)				ADMM feas. Y as in (3.5)			
	LowBnd	UpBnd	Iter.HighRk	Iter.LowRk	LowBnd	UpBnd	Iter.HighRk	Iter.LowRk
Chr18b	1534	2264	3843	243	477	2446	19,642	21,302
Esc32c	616	686	3177	780	529	692	2460	40,000
Esc32h	425	482	21,515	795	330	522	1685	40,000
Tai35a	2,216,645	2,599,924	3225	661	2,030,958	2,728,422	1222	40,000
Tai40a	2,843,312	3,392,692	4665	852	2,594,394	3,475,274	1433	40,000
Tho40	226,522	269,452	21,131	828	215,639	290,124	40,000	40,000

Table 7 New lower bounds by the proposed approaches for **QAPLIB** unsolved instances

Problem	QAPLIB LowBnd	ADMM LowBnd	% Impr LowBnd	QAPLIB %gap	New %gap
Sko42	14,934	15,335	2.61	5.55	3.02
Sko49	22,004	22,653	2.86	5.91	3.13
Sko56	32,610	33,390	2.34	5.36	3.10
Sko64	45,736	46,888	2.46	5.70	3.32
Sko72	62,691	64,205	2.36	5.38	3.10
Sko81	86,072	87,756	1.92	5.41	3.56
Sko90	109,030	111,300	2.04	5.63	3.66
Sko100a	143,846	145,775	1.32	5.37	4.10
Sko100b	145,522	147,332	1.23	5.44	4.26
Sko100c	139,881	142,018	1.50	5.40	3.95
Sko100d	141,289	143,205	1.34	5.54	4.26
Sko100e	140,893	142,977	1.46	5.54	4.14
Sko100f	140,691	142,413	1.21	5.60	4.44
Tai30a	1,706,855	1,706,871	<0.01	6.12	6.12
Tai35a	2,216,627	2,216,645	<0.01	8.48	8.48
Tai40a	2,843,274	2,843,312	<0.01	9.43	9.43
Tai50a	4,390,920	4,390,976	<0.01	11.09	11.09
Tai60a	5,578,356	6,319,630	11.73	22.59	12.30
Tai80a	10,501,941	11,613,474	9.57	22.20	13.97
Tai100	15,844,731	17,704,527	10.50	24.74	15.90
Tho40	224,414	226,522	0.93	6.69	5.82
Wil50	47,098	48,125	2.13	3.52	1.42
Wil100	264,442	267,469	1.13	3.15	2.04

that for all instances, **ADMM** can reach an accuracy of 10^{-5}. However, for most instances, it cannot reach the accuracy of 10^{-12} even though running to 40,000 iterations.

4.3 Results on QAPLIB instances II

Since the tests on QAPLIB instances I show no improvement from the smaller tolerance 10^{-12}, we simply set the tolerance to 10^{-5} for the tests on QAPLIB instances II. For the instances with size $n < 60$, we set the maximum number of iterations to 40,000. For larger instances, to reduce cputime, we simply run **ADMM** with the rank-1 constraint and **ADMM** without rank-1 constraint, but each to a maximum 2,000 iterations. For the former, at every 100 iterations, we found a feasible solution (thus an upper bound) by the method in Sect. 3.2. For the latter, at every 100 iterations we obtain a lower bound and also an upper bound by the methods in Sects. 3.1 and 3.2. We reported the best lower and upper bounds that we obtained. The results are shown in Table 3 for

instances of size $n < 60$ and in Table 4 for instances of size $n \geq 60$. The columns used are similar to those in Tables 1 and 2.

Remark 4.3 From Tables 3 and 4, we see that our method *provably* found exact optimal solutions for the 15 instances:

chr12a, chr12b, chr12c, chr15a, chr15b, chr15c, chr18a, chr20a, chr20b, chr22a, chr22b, chr25a, Esc16f, Esc32e, Esc32f.

For the rest of the instances, our method yielded a relative gap smaller than 20% for 26 instances, between 20% to 40% for 8 instances, and greater than 40% for only 1 instance. In addition, the **ADMM** with the rank-1 constraint reached the same stopping tolerance in much less time.

4.4 Influence of the nonnegativity constraints

To highlight the importance of the nonnegativity constraints in strengthening the bounds, i.e., in using a **DNN** model, we now compare results with and without the restriction $0 \leq Y \leq 1$, i.e., Y is updated according to (3.5) or (3.7). For the instances in Table 1 with $n \leq 24$, we obtained the same lower bounds as those from the HKM p–d i-p approach by updating Y according to (3.5). The upper and lower bounds for the remaining 10 instances by **ADMM** with updates (3.5) and (3.7) are shown in Table 5. We see that for all those 10 problems, **ADMM** using (3.7) obtained better lower bounds. **ADMM** with the rank-1 constraint can hardly achieve the tolerance 10^{-5} if the bound constraint is not enforced. In addition, except for Kra30b, Kra32, and Nug28, better upper bounds were also obtained by using (3.7).

Moreover, for the instances in Table 3 that were solved to optimality, if we update Y according to (3.5), the generated solution will not be optimal any more. For most of these 15 instances, **ADMM** with update (3.5) yielded the trivial lower bound 0. In Table 6, we present the 6 instances, for which **ADMM** with (3.7) improved the relative gap significantly over that with (3.5).

4.5 Improved lower bounds

For the problems marked with $*$ in Tables 1, 3, and 4, their optimal values are still unknown, and we obtained better lower bounds than those given in [7]. In Table 7, the fourth column shows the improvement percentage of the lower bounds for those 23 instances. Its last two columns list the gap between current lower bound and the best known feasible solutions according to Tables 1, 3, and 4, and also the improved gap by the proposed approach. We note that around 10% improvement has been achieved on instances Tai60a, Tai80a, and Tai100a, 2% on 6 instances, and less than 0.01% improvement on the other 4 instances.

5 Conclusion

In this paper we have shown the efficiency of using the **ADMM** approach for solving the *facially reduced* **SDP** relaxation of the **QAP** problem with added nonnegativity constraints, i.e., the usually hard-to-solve **DNN** relaxation. We exploited the **FR** relation $Y = VRV^T$ by applying the polyhedral constraints to Y and the positive semidefinite and rank constraints to R. The addition of the nonnegativity constraints to Y causes essentially no extra cost but significantly improves the bounds. For most instances in QAPLIB, we have improved both lower and upper bounds for the **QAP**, and in several instances, the bounds provably find the optimal permutation matrix.

In a forthcoming study, begun in [19], we propose to include this in a branch and bound framework and implement it in a parallel programming approach, see e.g., [14]. In addition, we propose to test the possibility of using *warm starts* in the branching/bounding process and test it on the larger test sets such as used in e.g., [9].

The most expensive steps of our code was the matrix multiplication $W = V\bar{W}V^T$ and the eigenvalue decomposition of W. We hope that a more efficient approach for this special matrix multiplication can be found. Moreover, since only a few eigenvalues of W are needed it is hoped that a more efficient algorithm can be used, e.g., the MATLAB code *eigifp* based on [12].

References

1. Anstreicher, K.M.: Recent advances in the solution of quadratic assignment problems. Math. Program. **97**(1–2), 27–42 (2003)
2. Anstreicher, K.M., Brixius, N.W.: A new bound for the quadratic assignment problem based on convex quadratic programming. Math. Program. **89**(3), 341–357 (2001)
3. Bhati, R.K., Rasool, A.: Quadratic assignment problem and its relevance to the real world: a survey. Int. J. Comput. Appl. **96**(9), 42–47 (2014)
4. Birkhoff, G.: Three observations on linear algebra. Univ. Nac. Tucumán. Revista A **5**, 147–151 (1946)
5. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011)
6. Burer, S., Monteiro, R.D.C.: Local minima and convergence in low-rank semidefinite programming. Math. Program. **103**(3), 427–444 (2005)
7. Burkard, R.E., Karisch, S., Rendl, F.: QAPLIB–a quadratic assignment problem library. Eur. J. Oper. Res. **55**, 115–119 (1991)
8. Burkard, R.E., Karisch, S.E., Rendl, F.: QAPLIB–a quadratic assignment problem library. J. Global Optim. **10**(4), 391–403 (1997)
9. de Klerk, E., Sotirov, R.: Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem. Math. Program. **122**(2), 225–246 (2010)
10. Eckart, C., Young, G.: The approximation of one matrix by another of lower rank. Psychometrica **1**(3), 211–218 (1936)
11. Edwards, C.S.: A branch and bound algorithm for the Koopmans–Beckmann quadratic assignment problem. Math. Program. Study **13**, 35–52 (1980)
12. Golub, G.H., Ye, Q.: An inverse free preconditioned Krylov subspace method for symmetric generalized eigenvalue problems. SIAM J. Sci. Comput. **24**(1), 312–334 (2002). (electronic)
13. Ito, N., Kim, S., Kojima, M., Takeda, A., Toh, K.C.: Bbcpop: a sparse doubly nonnegative relaxation of polynomial optimization problems with binary, box and complementarity constraints. arXiv preprint arXiv:1804.00761 (2018)
14. Jain, R., Yao, P.: A parallel approximation algorithm for positive semidefinite programming. In: 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science—FOCS 2011, pp. 463–471. IEEE Computer Soc., Los Alamitos, CA (2011)

15. Jiang, Bo, Liu, Ya-Feng, Wen, Zaiwen: l_p-norm regularization algorithms for optimization over permutation matrices. SIAM J. Optim. **26**(4), 2284–2313 (2016)
16. Kim, S., Kojima, M., Toh, K.-C.: A Lagrangian-DNN relaxation: a fast method for computing tight lower bounds for a class of quadratic optimization problems. Math. Program. **156**(1–2), 161–187 (2016)
17. Koopmans, T.C., Beckmann, M.J.: Assignment problems and the location of economic activities. Econometrica **25**, 53–76 (1957)
18. Lawler, E.L.: The quadratic assignment problem. Manag. Sci. **9**, 586–599 (1963)
19. Liao, Z.: Branch and bound via ADMM for the quadratic assignment problem. Master's thesis, University of Waterloo (2016)
20. Pardalos, P., Rendl, F., Wolkowicz, H.: The quadratic assignment problem: a survey and recent developments. In: Pardalos, P.M., Wolkowicz, H. (eds.) Quadratic Assignment and Related Problems (New Brunswick, NJ, 1993), pp. 1–42. American Mathematical Society, Providence, RI (1994)
21. Pardalos, P., Wolkowicz, H. (eds.).: Quadratic assignment and related problems. American Mathematical Society, Providence, RI, 1994. Papers from the workshop held at Rutgers University, New Brunswick, New Jersey, May 20–21 (1993)
22. Pong, T.K., Sun, H., Wang, N., Wolkowicz, H.: Eigenvalue, quadratic programming, and semidefinite programming relaxations for a cut minimization problem. Comput. Optim. Appl. **63**(2), 333–364 (2016)
23. Povh, J., Rendl, F.: Copositive and semidefinite relaxations of the quadratic assignment problem. Discret. Optim. **6**(3), 231–241 (2009)
24. Rendl, F., Sotirov, R.: Bounds for the quadratic assignment problem using the bundle method. Math. Program. **109**(2–3), 505–524 (2007)
25. Toh, K.C., Todd, M.J., Tütüncü, R.H.: SDPT3—a MATLAB software package for semidefinite programming, version 1.3. Optim. Methods Softw. **11**(1–4), 545–581 (1999)
26. Wen, Z., Goldfarb, D., Yin, W.: Alternating direction augmented Lagrangian methods for semidefinite programming. Math. Program. Comput. **2**(3–4), 203–230 (2010)
27. Yang, L., Sun, D., Toh, K.-C.: SDPNAL+: a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. Math. Program. Comput. **7**(3), 331–366 (2015)
28. Zhao, Q., Karisch, S.E., Rendl, F., Wolkowicz, H.: Semidefinite programming relaxations for the quadratic assignment problem. J. Comb. Optim. **2**(1), 71–109 (1998)
29. Zhao, X.Y., Sun, D., Toh, K.C.: A Newton-CG augmented lagrangian method for semidefinite programming. SIAM J. Optim. **20**(4), 1737–1765 (2010)