Exploiting Aggregate Sparsity in Second Order Cone Relaxations for Quadratic Constrained Quadratic Programming Problems

Heejune Sheen[*]   Makoto Yamashita[†]

November, 2019

Abstract

Among many approaches to increase the computational efficiency of semidefinite programming (SDP) relaxation for quadratic constrained quadratic programming problems (QCQPs), exploiting the aggregate sparsity of the data matrices in the SDP by Fukuda et al. (2001) and second-order cone programming (SOCP) relaxation have been popular. In this paper, we exploit the aggregate sparsity of SOCP relaxation of QCQPs. Specifically, we prove that exploiting the aggregate sparsity reduces the number of second-order cones in the SOCP relaxation, and that we can simplify the matrix completion procedure by Fukuda et al. in both primal and dual of the SOCP relaxation problem without losing the max-determinant property. For numerical experiments, QCQPs from the lattice graph and pooling problem are tested as their SOCP relaxations provide the same optimal value as the SDP relaxations. We demonstrate that exploiting the aggregate sparsity improves the computational efficiency of the SOCP relaxation for the same objective value as the SDP relaxation, thus much larger problems can be handled by the proposed SOCP relaxation than the SDP relaxation.

[*]School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA (`brianshn1@gmail.com`). This research was conducted while this author was a research student at the Department of Mathematical and Computing Science, Tokyo Institute of Technology.

[†]Department of Mathematical and Computing Science, Tokyo Institute of Technology, 2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152-8552, Japan (`Makoto.Yamashita@c.titech.ac.jp`). This research was partially supported by JSPS KAKENHI (Grant number: 18K11176).

1

Key words. Quadratic constrained quadratic programming, semidefinite programming, second-order cone programming, aggregate sparsity, chordal sparsity.

AMS Classification. 90C20, 90C22, 90C25, 90C26.

1 Introduction

Quadratically constrained quadratic programming problems (QCQPs) represent an important class of optimization problems in both theory and practice. A variety of problems arising from engineering and combinatorial applications can be formulated as QCQPs, for example, quadratic assignment problem [5], radar detection [11], and graph theory [26]. More applications can be found in [7]. If QCQPs are convex, many efficient algorithms exist to find their solutions [17, 28]. Nonconvex QCQPs are, however, known as NP-hard in general [10].

Nonconvex QCQPs have been studied by relaxation methods via lifting and convexification [9], most notably, semidefinite programming (SDP) relaxation. SDP relaxation has been popular as it can obtain tight approximate optimal values of QCQPs. In fact, SDP relaxation has been applied to a broad range of problems such as the maxcut problems [14], sensor network localization [8], optimal contribution selection [31, 34], and the pooling problem [21, 30].

Solving the SDP relaxation of large QCQPs can be very time-consuming and obtaining an approximate optimal value with accuracy is often difficult due to numerical instability [31]. It is particularly true when the primal-dual interior-point methods [25, 32] are used to solve the SDP relaxation. As a result, various methods have been proposed to alleviate the difficulties. The chordal sparsity exploitation proposed by Fukuda et al. [13] for SDP problems is regarded as a systematic method that effectively utilizes the structure of the data matrices. In [13], the variable matrix was decomposed into small sub-matrices, each of which was associated with the maximal cliques of the chordal graph of the SDP. To relate the resulting sub-matrices for the equivalence to the original SDP, additional equality constraints were added to the SDP problem with the decomposed sub-matrices. After solving the SDP with the sub-matrices and equality constraints, a completion procedure to patch the sub-matrices was performed to recover the original variable matrix as the final solution. Fukuda et al. showed that the completion procedure

2

results in a matrix with the maximum determinant among all possible completed matrices. From the computational perspective, the computational gain by exploiting the chordal sparsity in SDPs is clear only when the resulting SDPs have small sizes of the sub-matrices and moderate numbers of additional equality constraints.

The chordal sparsity in SDP relaxation has been studied and implemented in many literature and softwares. The chordal sparsity was studied from various angles in [33] by Vandenberghe and Andersen. Kim et al. [20] introduced a Matlab software package called SparseCoLO [12] that exploits the chordal sparsity of matrix inequalities. SDPA-C [35] is implementation of a parallel approach for SDPs using the chordal sparsity. Mason et al. [24] applied the chordal sparsity to the Lyapunov equation. In contrast, the chordal sparsity has not been studied in the context of second-order cone programming (SOCP) relaxation.

For some classes of QCQPs, the SOCP relaxation provides the same optimal value as the SDP relaxation, though it is a weaker relaxation than the SDP relaxation in general. Kim and Kojima [19] proved that nonconvex QCQPs with non-positive off-diagonal elements can be exactly solved by the SDP or SOCP relaxations. More recently, for QCQPs with zero diagonal elements in the data matrices, Kimizuka et al. [21] showed that the SDP, SOCP and linear programming (LP) relaxations provide the same objective value. For these classes of QCQPs, the SOCP relaxations are far more efficient than the SDP relaxation.

The main purpose of this paper is to exploit the aggregate sparsity in the SOCP relaxation. Our approach in this paper has two major differences from the method utilizing the chordal sparsity in SDP [13]. First, it does not require any additional equality constraints to exploit the sparsity in the SOCP. Thus, the proposed method for the SOCP can improve the computational efficiency for solving the SOCP relaxation. The problems, for which the SDP and SOCP relaxations provide the same optimal value, especially can benefit from the increased efficiency of the proposed method. Second, our approach can generate a completed matrix that attains the maximum determinant without relying on the completion procedure as in the SDP relaxation. As a result, the completion procedure is not necessary in our approach. Thus, we can expect that the SOCP relaxation can be solved much faster.

We report numerical results to demonstrate the efficiency improved by exploiting the sparsity in the SOCP relaxation. We generated test instances to satisfy the condition of [19] using lattice graphs, and we also tested instances

3

of the pooling problems from [21]. Through the numerical experiments, we observe that the SOCP relaxation with the proposed sparsity exploitation spends much less computational time to obtain the same objective value as the SDP relaxations.

The rest of the paper is organized as follows. In Section 2, we briefly review relaxation methods for QCQPs and describe some background on exploiting the chordal sparsity in SDPs. In Section 3, we discuss the sparsity exploitation in the SOCP relaxation. We prove that the completed matrix in the proposed approach attains the maximum determinant among possible completed matrices. In Section 4, we report the numerical results on the QCQPs from the lattice graph and pooling problem. Finally, in Section 5, we give our conclusion remarks.

2   Preliminaries

We start this section by introducing some notation that will be used in this paper. We use the superscript $\top$ to denote the transpose of a matrix or a vector. Let $\mathbb{R}^n$ be a n-dimensional Euclidean space. We use $\mathbb{R}^n_+ \subset \mathbb{R}^n$ for a nonnegative orthant in n-dimensional Euclidean space. Let $\mathbb{R}^{n \times n}$ be the set of $n \times n$ real matrices, and let $\mathbb{S}^n \subset \mathbb{R}^{n \times n}$ be the set of $n \times n$ real symmetric matrices. We use $\mathbb{S}^n_+(\mathbb{S}^n_{++}) \subset \mathbb{S}^n$ for the set of positive semidefinite matrices (positive definite matrices, respectively) of dimension n. We also use $\boldsymbol{X} \succeq \boldsymbol{O}(\boldsymbol{X} \succ \boldsymbol{O})$ to denote $\boldsymbol{X}$ is a positive semidefinite matrix (a positive definite matrix, respectively). The inner product $\boldsymbol{X} \bullet \boldsymbol{Y}$ between $\boldsymbol{X}$ and $\boldsymbol{Y}$ in $\mathbb{S}^n$ is defined by $\boldsymbol{X} \bullet \boldsymbol{Y} = trace(\boldsymbol{X}\boldsymbol{Y})$.

2.1   SDP and SOCP relaxations of QCQPs

A general form of QCQPs can be given as follows:

$$
\begin{array}{rl}
\text{minimize} & : \quad \boldsymbol{x}^\top \boldsymbol{P}_0 \boldsymbol{x} + 2\boldsymbol{q}_0^\top \boldsymbol{x} + r_0 \\
\text{subject to} & : \quad \boldsymbol{x}^\top \boldsymbol{P}_k \boldsymbol{x} + 2\boldsymbol{q}_k^\top \boldsymbol{x} + r_k \leq 0 \text{ for } 1 \leq k \leq m,
\end{array} \tag{1}
$$

where the decision variable is $\boldsymbol{x} \in \mathbb{R}^n$, and the input data are $\boldsymbol{P}_k \in \mathbb{S}^n$, $\boldsymbol{q}_k \in \mathbb{R}^n$ and $r_k \in \mathbb{R}$ for $k = 0, \ldots, m$.

If $\boldsymbol{P}_k \succeq \boldsymbol{O}$ for all k, the problem (1) is a convex problem and can be solved efficiently, for example, by interior-point methods [29]. In contrast, if

4

$\boldsymbol{P}_k$ is not positive semidefinite matrix for some k, the problem is nonconvex and can be NP [10].

We define $\boldsymbol{Q}_k := \begin{bmatrix} r_k & \boldsymbol{q}_k^\top \\ \boldsymbol{q}_k & \boldsymbol{P}_k \end{bmatrix} \in \mathbb{S}^{n+1}$ for each $k = 0, \dots, m$, and introduce a variable matrix $\boldsymbol{X} \in \mathbb{S}^{n+1}$. The standard SDP relaxation based on lift-and-project convex relaxation [6, 16, 23] for QCQP (1) can be given as follows:

$$
\begin{aligned}
\text{minimize:} \quad & \boldsymbol{Q}_0 \bullet \boldsymbol{X} \\
\text{subject to:} \quad & \boldsymbol{Q}_k \bullet \boldsymbol{X} \leq 0 \text{ for } 1 \leq k \leq m \\
& \boldsymbol{H}_0 \bullet \boldsymbol{X} = 1 \\
& \boldsymbol{X} \in \mathbb{S}_+^{n+1},
\end{aligned}
\tag{2}
$$

where $\boldsymbol{H}_0 := \begin{bmatrix} 1 & \boldsymbol{0}^\top \\ \boldsymbol{0} & \boldsymbol{O} \end{bmatrix} \in \mathbb{S}^{n+1}$.

By further relaxing the positive semidefinite condition $\boldsymbol{X} \in \mathbb{S}_+^{n+1}$ with the positive semidefinite conditions of all 2×2 principal sub-matrices, Kim and Kojima [18] proposed an SOCP relaxation (3) below. This corresponds to the dual of the first level of the scaled diagonally dominant sum-of-squares (SDSOS) relaxation hierarchy discussed in [1].

$$
\begin{aligned}
\text{minimize:} \quad & \boldsymbol{Q}_0 \bullet \boldsymbol{X} \\
\text{subject to:} \quad & \boldsymbol{Q}_k \bullet \boldsymbol{X} \leq 0 \text{ for } 1 \leq k \leq m \\
& \boldsymbol{H}_0 \bullet \boldsymbol{X} = 1 \\
& \boldsymbol{X} \in \mathbb{T}_+^{n+1}.
\end{aligned}
\tag{3}
$$

Here, $\mathbb{T}_+^{n+1}$ is defined to denote the set of symmetric matrices that satisfy the positive semidefinite conditions of all 2×2 principal sub-matrices as

$$
\mathbb{T}_+^{n+1} := \{ \boldsymbol{X} \in \mathbb{S}^{n+1} | \boldsymbol{X}^{ij} \succeq \boldsymbol{O} \text{ for } (i,j) \in J \}.
$$

where $\boldsymbol{X}^{ij}$ is defined by

$$
\boldsymbol{X}^{ij} := \begin{pmatrix} X_{ii} & X_{ij} \\ X_{ij} & X_{jj} \end{pmatrix} \in \mathbb{S}^2
$$

and J is an index set defined by $J := \{(i,j) | 1 \leq i < j \leq n+1\}$. The equivalence between $\boldsymbol{X}^{ij} \succeq \boldsymbol{O}$ and $\frac{X_{ii}+X_{jj}}{2} \geq \left\| \begin{pmatrix} \frac{X_{ii}-X_{jj}}{2} \\ X_{ij} \end{pmatrix} \right\|$ enables us to

formulate (3) problem as an SOCP [18], therefore, (3) is an SOCP relaxation of (1).

Due to the relation $\mathbb{S}_+^{n+1} \subset \mathbb{T}_+^{n+1}$, the SOCP relaxation (3) is weaker than the SDP relaxation (2). However, there exist certain classes of QCQPs whose SDP and SOCP relaxations are exact, for example, [19].

2.2 The matrix completion with chordal graph

It is frequently observed in many applications that the matrices $\boldsymbol{Q}_0, \boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_m$ have some structural sparsity. Fukuda et al. [13] and Nakata et al. [27] exploited the sparsity in the data matrices using the chordal graph. We call such sparsity related to the chordal graph the chordal sparsity. We briefly describe their matrix completion technique.

For the SDP relaxation problem (2), we first define an aggregate sparsity graph $G(V, \overline{E})$ as an undirected graph with the set of vertices $V = \{1, 2, \ldots, n+1\}$ and the set of edges

$$\overline{E} = \{(i, j) \in J \mid \text{and} \ \ [\boldsymbol{Q}_k]_{ij} \neq 0 \text{ for some } k \in \{0, 1, \ldots, m\}\}.$$

We sometimes call $\overline{E}$ the aggregate sparsity pattern, and the sparsity related to $\overline{E}$ the aggregate sparsity. An undirected graph is called *chordal* if every cycle of four or more vertices has a chord. When the aggregate sparsity graph $G(V, \overline{E})$ is not chordal, we can find a chordal graph $G(V, \widehat{E})$ such that $\overline{E} \subset \widehat{E} \subset J$ by adding appropriate edges to $\overline{E}$. The graph $G(V, \widehat{E})$ is called a chordal extension, and it is known that the chordal extension is related with the sparse Cholesky factorization [33].

A vertex set $C \subset V$ is called a clique if the induced subgraph $G(C, (C \times C) \cap \overline{E})$ is a complete graph, and a clique C is called a maximal clique if it is not a subset of any other clique. When $G(V, \widehat{E})$ is a chordal graph, we can enumerate the set of maximal cliques $\Lambda = \{C_1, \ldots, C_p\}$, where p is the number of maximal cliques. Without loss of generality, we can assume that the order of $\{C_1, \ldots, C_p\}$ is a perfect elimination ordering [13, 27], and that each vertex in V is covered by at least one maximal clique, that is, $V = \cup_{l=1}^p C_l$.

We use the following notation for an edge set $E \subset J$ and a matrix $\overline{\boldsymbol{X}} \in$

$\mathbb{S}^{n+1}$:

$$
\begin{aligned}
\mathbb{S}^{n+1}(E, \overline{\boldsymbol{X}}, ?) &= \{\boldsymbol{X} \in \mathbb{S}^{n+1} | X_{ij} = \overline{X}_{ij} \text{ for } (i,j) \in E \cup D\} \\
\mathbb{S}^{n+1}_+(E, ?) &= \{\boldsymbol{X} \in \mathbb{S}^{n+1} | \exists \widehat{\boldsymbol{X}} \in \mathbb{S}^{n+1}_+ \text{ such that } \widehat{\boldsymbol{X}} \in \mathbb{S}^{n+1}(E, \boldsymbol{X}, ?)\} \\
\mathbb{S}^{n+1}(E, 0) &= \{\boldsymbol{X} \in \mathbb{S}^{n+1} | X_{ij} = X_{ji} = 0 \text{ for } (i,j) \notin E \cup D\} \\
\mathbb{S}^{n+1}_+(E, 0) &= \mathbb{S}^{n+1}(E, 0) \cap \mathbb{S}^{n+1}_+.
\end{aligned}
$$

Here, D is the index set that corresponds to the diagonal elements; $D = \{(i,i) : 1 \le i \le n+1\}$. We also define $\overline{\boldsymbol{X}}(E, C)$ as a matrix in $\mathbb{S}^{n+1}(E, \overline{\boldsymbol{X}}, ?) \cap \mathbb{S}^{n+1}((C \times C) \cap J, 0)$. Note that if E is the edge set of the chordal extension (that is, $E = \widehat{E}$) and C_l is a clique of $G(V, \widehat{E})$, then $\overline{\boldsymbol{X}}(\widehat{E}, C_l)$ is uniquely determined for a given $\overline{\boldsymbol{X}} \in \mathbb{S}^{n+1}$.

The fundamental theorem on the matrix completion by Grone et al. [15] can be described as follows.

Theorem 1. [15] *Fix* $\overline{\boldsymbol{X}} \in \mathbb{S}^{n+1}$. *For* $\boldsymbol{X} \in \mathbb{S}^{n+1}(\widehat{E}, \overline{\boldsymbol{X}}, ?)$, *it holds that* $\boldsymbol{X} \in \mathbb{S}^{n+1}_+(\widehat{E}, ?)$ *if and only if* $\overline{\boldsymbol{X}}(\widehat{E}, C_l) \succeq \boldsymbol{O}$ *for each* $C_l \in \Lambda$.

The main idea in [13] and [27] is to replace the positive semidefinite condition $\boldsymbol{X} \succeq \boldsymbol{O}$ with the positive semidefinite conditions for the submatrices $\boldsymbol{X}(\widehat{E}, C_1), \ldots, \boldsymbol{X}(\widehat{E}, C_p) \succeq \boldsymbol{O}$. More precisely, the following SDP (4) is solved instead of (2):

$$
\begin{aligned}
\text{minimize:} \quad & \sum_{l=1}^{p} \boldsymbol{Q}_0(\widehat{E}, C_l) \bullet \boldsymbol{X}(\widehat{E}, C_l) \\
\text{subject to:} \quad & \sum_{l=1}^{p} \boldsymbol{Q}_k(\widehat{E}, C_l) \bullet \boldsymbol{X}(\widehat{E}, C_l) \le 0 \ (k = 1, ..., m) \\
& [\boldsymbol{X}(\widehat{E}, C_u)]_{11} = 1 \text{ for } u \in \{u \in \{1, 2, \ldots, p\} | 1 \in C_u\} \\
& [\boldsymbol{X}(\widehat{E}, C_u)]_{ij} = [\boldsymbol{X}(\widehat{E}, C_v)]_{ij} \text{ for } (i, j, u, v) \in U \\
& \boldsymbol{X}(\widehat{E}, C_l) \succeq \boldsymbol{O} \ (l = 1, ..., p).
\end{aligned} \tag{4}
$$

For each k, $\boldsymbol{Q}_k$ is decomposed into $\boldsymbol{Q}_k(\widehat{E}, C_1), \ldots, \boldsymbol{Q}_k(\widehat{E}, C_p) \in \mathbb{S}^{n+1}$ such that $\boldsymbol{Q}_k = \sum_{l=1}^{p} \boldsymbol{Q}_k(\widehat{E}, C_l)$. It is always possible to decompose as above since $\widehat{E}$ is a chordal extension of the aggregate sparsity pattern graph. The set U defined by

$$
\begin{aligned}
U = \{(i, j, u, v) \quad | \quad & (i, j) \in (C_u \times C_u) \cap (C_v \times C_v)\} \setminus \{(1, 1)\}, \ i < j, \\
& (C_u \times C_u) \cap (C_v \times C_v) \ne \emptyset \ \text{ for } \ 1 \le u < v \le p\}.
\end{aligned}
$$

represents the overlaps among the cliques, thus the additional equality constraints $[\boldsymbol{X}(\widehat{E}, C_u)]_{11} = 1$ and $[\boldsymbol{X}(\widehat{E}, C_u)]_{ij} = [\boldsymbol{X}(\widehat{E}, C_v)]_{ij}$ should be introduced in (4) for the overlapped elements.

We use $\overline{\boldsymbol{X}}(\widehat{E}, C_1), \ldots, \overline{\boldsymbol{X}}(\widehat{E}, C_l)$ to denote a solution of (4). To simplify the discussion here, we assume $\overline{\boldsymbol{X}}(\widehat{E}, C_1), \ldots, \overline{\boldsymbol{X}}(\widehat{E}, C_l)$ are positive definite matrices. With the additional equality constraints above, we can uniquely determine the entire matrix $\overline{\boldsymbol{X}} \in \mathbb{S}^{n+1}$ such that

$$\overline{X}_{ij} = \begin{cases} [\overline{\boldsymbol{X}}(\widehat{E}, C_l)]_{ij} & \text{if} \quad (i,j) \in C_l \times C_l \text{ for some } l \in \{1, \ldots, p\} \\ 0 & \text{if} \quad (i,j) \notin (C_1 \times C_1) \cup \ldots \cup (C_p \times C_p). \end{cases}$$

Since the entire $\overline{\boldsymbol{X}}$ is not necessarily positive semidefinite, we complete $\overline{\boldsymbol{X}}$ to $\widehat{\boldsymbol{X}} \in \mathbb{S}_+(\widehat{E}, ?)$ by the completion procedure of [13]. The completion procedure is described in detail in [27] where the matrix is completed in the order of the maximal cliques $\{C_1, \ldots, C_p\}$. We include the following lemma on $\widehat{\boldsymbol{X}}$ for the subsequent discussion.

Lemma 2. [13]. *The completed matrix $\widehat{\boldsymbol{X}}$ computed by the completion procedure is the unique positive definite matrix that maximizes the determinant among all possible matrix completions of $\overline{\boldsymbol{X}}$, that is,*

$$\det \widehat{\boldsymbol{X}} = \max \left\{ \det \boldsymbol{X} \ \middle| \ \boldsymbol{X} \in \mathbb{S}_{++}^{n+1} \cap \mathbb{S}^{n+1}(\widehat{E}, \overline{\boldsymbol{X}}, ?) \right\}.$$

It follows from [13] that the completed matrix $\widehat{\boldsymbol{X}}$ is an optimal solution of the SDP relaxation problem (2).

We should mention that there exists trade-off in terms of the computational efficiency for solving (4). Replacing $\boldsymbol{X} \succeq \boldsymbol{O}$ with $\boldsymbol{X}(\widehat{E}, C_1), \ldots,$ $\boldsymbol{X}(\widehat{E}, C_p) \succeq \boldsymbol{O}$ reduces the computational cost required for $\boldsymbol{X} \succeq \boldsymbol{O}$, especially when the size of $\boldsymbol{X}$ is large. However, the additional equality constraints such as $[\boldsymbol{X}(\widehat{E}, C_u)]_{11} = 1$ and $[\boldsymbol{X}(\widehat{E}, C_u)]_{ij} = [\boldsymbol{X}(\widehat{E}, C_v)]_{ij}$ can be new computational burden. Moreover, the completion procedure needs to be performed. Thus, the conversion to (4) works well when the cliques $C_1, \ldots, C_p$ are small and the size of the set U for the overlapping elements is small.

3 The sparsity of the SOCP relaxation

In this section, we discuss how the aggregate sparsity in the SOCP relaxation (3) can be exploited and how it is different from the case in the SDP relax-

ation. We will also investigate the sparsity exploitation using dual problems.

3.1 A matrix completion in the SOCP relaxation

Similarly to the chordal sparsity in the SDP relaxation, we define the following notation for an edge set $E \subset J$:

$$
\begin{aligned}
\mathbb{T}_+^{n+1}(E, ?) &:= \{\boldsymbol{X} \in \mathbb{S}^{n+1} | \exists \widehat{\boldsymbol{X}} \in \mathbb{T}_+^{n+1} \text{ such that } \widehat{\boldsymbol{X}} \in \mathbb{S}^{n+1}(E, \boldsymbol{X}, ?)\} \\
\mathbb{T}_+^{n+1}(E, 0) &:= \mathbb{S}^{n+1}(E, 0) \cap \mathbb{T}_+^{n+1}.
\end{aligned}
$$

The set $\overline{\overline{\mathbb{T}}}_+(E)$ for an edge set E is defined by

$$
\overline{\overline{\mathbb{T}}}_+^{n+1}(E) := \{\boldsymbol{X} \in \mathbb{S}^{n+1} | \boldsymbol{X}^{ij} \succeq \boldsymbol{O} \text{ for } (i, j) \in E, X_{ii} \geq 0 \text{ for } i \in V_e\},
$$

where
$$
\begin{aligned}
V_e &:= \{i \in V | (i, j) \notin E \text{ for } \forall j \in \{i+1, \ldots, n+1\} \\
&\quad \text{and } (j, i) \notin E \text{ for } \forall j \in \{1, \ldots, i-1\}\}
\end{aligned}
$$

is the set of vertices that are not involved in E. Since $\boldsymbol{X}^{ij} \succeq \boldsymbol{O}$ guarantees the non-negativeness of X_{ii} and X_{jj} for $(i, j) \in E$, we know that all the diagonal elements of any matrix in $\overline{\overline{\mathbb{T}}}_+^{n+1}(E)$ are nonnegative, that is, $X_{ii} \geq 0$ holds for each $i \in V$ if $\boldsymbol{X} \in \overline{\overline{\mathbb{T}}}_+^{n+1}(E)$.

We now examine the equivalence between $\mathbb{T}_+^{n+1}(E, ?)$ and $\overline{\overline{\mathbb{T}}}_+^{n+1}(E)$. For a matrix $\boldsymbol{X} \in \overline{\overline{\mathbb{T}}}_+^{n+1}(E)$, let us consider a range $R_{ij}(\boldsymbol{X}) := [-\sqrt{X_{ii}X_{jj}}, \sqrt{X_{ii}X_{jj}}]$ for each $(i, j) \in J$ and introduce a set

$$
\widehat{\mathbb{T}}_+^{n+1}(E, \boldsymbol{X}) := \{\widehat{\boldsymbol{X}} \in \mathbb{S}^{n+1}(E, \boldsymbol{X}, ?) | \widehat{X}_{ij} \in R_{ij}(\boldsymbol{X}) \text{ for } (i, j) \notin E \cup D\}.
$$

Lemma 3. *It holds that* $\mathbb{T}_+^{n+1}(E, ?) = \overline{\overline{\mathbb{T}}}_+^{n+1}(E)$.

Proof. First, we show $\mathbb{T}_+^{n+1}(E, ?) \subset \overline{\overline{\mathbb{T}}}_+^{n+1}(E)$. Fix $\boldsymbol{X} \in \mathbb{T}_+^{n+1}(E, ?)$. Then, there exists $\widehat{\boldsymbol{X}} \in \mathbb{T}_+^{n+1}$ such that $\widehat{X}_{ij} = X_{ij}$ for $(i, j) \in E \cup D$. Since $\widehat{\boldsymbol{X}} \in \mathbb{T}_+^{n+1}$ and $E \subset J$, we know $\begin{bmatrix} \widehat{X}_{ii} & \widehat{X}_{ij} \\ \widehat{X}_{ij} & \widehat{X}_{jj} \end{bmatrix} \succeq \boldsymbol{O}$ for $(i, j) \in E$, thus $\widehat{X}_{ij} = X_{ij}$ for $(i, j) \in E \cup D$ leads to $\boldsymbol{X}^{ij} \succeq \boldsymbol{O}$. Furthermore, for each $i \in V \backslash \{n+1\}$, it holds that $(i, i+1) \in J$. Then, $\begin{bmatrix} \widehat{X}_{ii} & \widehat{X}_{i,i+1} \\ \widehat{X}_{i,i+1} & \widehat{X}_{i+1,i+1} \end{bmatrix} \succeq \boldsymbol{O}$ guarantees $\widehat{X}_{ii} \geq 0$

9

and $\widehat{X}_{i+1,i+1} \geq 0$ for each $i \in V \backslash \{n + 1\}$. This implies $X_{ii} \geq 0$ for each $i \in V_e$, since $\widehat{X}_{ii} = X_{ii}$ for $(i, i) \in D$ and $V_e \subset V$.

For $\mathbb{T}_+^{n+1}(E, ?) \supset \overline{\mathbb{T}}_+^{n+1}(E)$, we fix $\boldsymbol{X} \in \overline{\mathbb{T}}_+^{n+1}(E)$ and take any matrix $\widehat{\boldsymbol{X}}$ from $\widehat{\mathbb{T}}_+^{n+1}(E, \boldsymbol{X})$. Then, we can show $\widehat{\boldsymbol{X}} \in \mathbb{T}_+^{n+1}$. In particular, if $(i, j) \notin E \cup D$, then we have $\begin{bmatrix} \widehat{X}_{ii} & \widehat{X}_{ij} \\ \widehat{X}_{ij} & \widehat{X}_{jj} \end{bmatrix} = \begin{bmatrix} X_{ii} & \widehat{X}_{ij} \\ \widehat{X}_{ij} & X_{jj} \end{bmatrix} \succeq \boldsymbol{O}$, since $-\sqrt{X_{ii}X_{jj}} \leq \widehat{X}_{ij} \leq \sqrt{X_{ii}X_{jj}}$.
□

Remark 1. From the proof of Lemma 3, we observe that for a given matrix $\overline{\boldsymbol{X}} \in \overline{\mathbb{T}}_+^{n+1}(\overline{E})$ corresponding to the aggregate sparsity graph $G(V, \overline{E})$, $\overline{\boldsymbol{X}}$ can be completed to some matrix $\widehat{\boldsymbol{X}} \in \widehat{\mathbb{T}}_+^{n+1}(\overline{E}, \overline{\boldsymbol{X}})$ without changing the elements specified in $\overline{E}$. In addition, $\widehat{\mathbb{T}}_+^{n+1}(\overline{E}, \overline{\boldsymbol{X}})$ covers all possible completion matrices of $\overline{\boldsymbol{X}}$ in $\mathbb{T}_+^{n+1}$.

As a result of the observation in Remark 1, we can modify (3) as the following SOCP (5). Notice that $\mathbb{T}_+^{n+1}$ is replaced with $\overline{\mathbb{T}}_+^{n+1}(\overline{E})$.

$$
\begin{aligned}
\text{minimize:} \quad & \boldsymbol{Q}_0 \bullet \boldsymbol{X} \\
\text{subject to:} \quad & \boldsymbol{Q}_k \bullet \boldsymbol{X} \leq 0 \text{ for } 1 \leq k \leq m \\
& \boldsymbol{H}_0 \bullet \boldsymbol{X} = 1 \\
& \boldsymbol{X} \in \overline{\mathbb{T}}_+^{n+1}(\overline{E}).
\end{aligned}
\tag{5}
$$

Let $\overline{\boldsymbol{X}}$ be an optimal solution of (5). We also let ζ and $\overline{\zeta}$ be the optimal values of (3) and (5), respectively. Since $\mathbb{T}_+^{n+1} \subset \overline{\mathbb{T}}_+^{n+1}(\overline{E})$ from the relation $\overline{E} \subset J$, we have $\zeta \geq \overline{\zeta}$, that is, (5) is a further relaxation of (3) in general. However, we can show the equivalence between ζ and $\overline{\zeta}$.

Theorem 4. *For ζ and $\overline{\zeta}$, we have $\zeta = \overline{\zeta}$. In addition, suppose that $\boldsymbol{X}^*$ and $\overline{\boldsymbol{X}}$ are optimal solutions of (3) and (5), respectively. Then, $\boldsymbol{X}^*$ is an optimal solution of (5), and $\widehat{\mathbb{T}}_+^{n+1}(\overline{E}, \overline{\boldsymbol{X}})$ is included in the set of optimal solutions of (3).*

Proof. First, we show that any matrix $\widehat{\boldsymbol{X}} \in \widehat{\mathbb{T}}_+^{n+1}(\overline{E}, \overline{\boldsymbol{X}})$ is an optimal solution of (3). From Remark 1, it follows that $\boldsymbol{Q}_0 \bullet \widehat{\boldsymbol{X}} = \boldsymbol{Q}_0 \bullet \overline{\boldsymbol{X}} = \overline{\zeta}$, $\boldsymbol{Q}_k \bullet \widehat{\boldsymbol{X}} = \boldsymbol{Q}_k \bullet \overline{\boldsymbol{X}}$ for $k = 1, \ldots, m$, and $\boldsymbol{H}_0 \bullet \widehat{\boldsymbol{X}} = \boldsymbol{H}_0 \bullet \overline{\boldsymbol{X}}$. As discussed in

10

the proof of Lemma 3, we have $\widehat{\boldsymbol{X}} \in \mathbb{T}_+^{n+1}$, therefore, $\widehat{\boldsymbol{X}}$ is a feasible solution of (3). Since its objective value $\boldsymbol{Q}_0 \bullet \widehat{\boldsymbol{X}}$ is $\overline{\zeta}$, $\overline{\zeta} \geq \zeta$ holds. By combining this with $\zeta \geq \overline{\zeta}$, we know that $\zeta = \overline{\zeta}$, and this implies $\widehat{\boldsymbol{X}}$ is an optimal solution of (3).

Finally, $\boldsymbol{X}^*$ is a feasible solution of (5) from $\mathbb{T}_+^{n+1} \subset \overline{\mathbb{T}}_+^{n+1}(\overline{E})$, and its objective value is ζ. Therefore, $\zeta = \overline{\zeta}$ leads to the conclusion that $\boldsymbol{X}^*$ is also an optimal solution of (5). $\qquad \square$

Lemma 2 from [13] shows that the completed matrix by the completion procedure has the maximum determinant among all possible matricies. To discuss a similar maximum-determinant property in the framework of (3), we need to introduce the determinant for $\mathbb{T}_+^{n+1}$.

From the self-concordant function discussed in [29] for the theoretical analysis of interior-point methods, we see that the standard self-concordant barrier function at $\boldsymbol{X} \in \mathbb{S}_{++}^{n+1}$ takes the form of $-\log \det \boldsymbol{X}$. In $\mathbb{T}_+^{n+1}$, we have multiple positive semidefinite matrices $\boldsymbol{X}^{ij}$ for $(i,j) \in J$, thus, if interior-point methods are applied to (3), $\sum_{(i,j)\in J}(-\log \det \boldsymbol{X}^{ij}) = -\log \left(\Pi_{(i,j)\in J} \det \boldsymbol{X}^{ij}\right)$ can be used as a self-concordant barrier function. Using the analogy, we define the determinant for $\boldsymbol{X} \in \mathbb{T}_+^{n+1}$ by

$$\det_{\mathbb{T}} \boldsymbol{X} := \Pi_{(i,j)\in J} \det \boldsymbol{X}^{ij}. \qquad (6)$$

Note that $\boldsymbol{X}^{ij} \succeq \boldsymbol{O}$ is equivalent to a second-order cone constraint $\frac{X_{ii}+X_{jj}}{2} \geq \left\| \begin{pmatrix} \frac{X_{ii}-X_{jj}}{2} \\ X_{ij} \end{pmatrix} \right\|$. Let $\det_{\mathrm{SOCP}}$ denote the determinant of $\boldsymbol{v} = (v_0, v_1, v_2)^\top \in \mathbb{R}$ for a second-order cone $v_0 \geq \sqrt{v_1^2 + v_2^2}$ defined in [4]. Then, $\det_{\mathrm{SOCP}}(\boldsymbol{v}) = v_0^2 - v_1^2 - v_2^2$, and $\det_{\mathrm{SOCP}}\left(\left(\frac{X_{ii}+X_{jj}}{2}, \frac{X_{ii}-X_{jj}}{2}, X_{ij}\right)^\top\right) = \det \boldsymbol{X}^{ij}$ follows. Thus, the definition of $\det_{\mathbb{T}}$ in (6) is also consistent with $\det_{\mathrm{SOCP}}$ in [4].

For $\overline{\boldsymbol{X}} \in \overline{\mathbb{T}}_+^{n+1}(\overline{E})$, we note that the set $\mathbb{S}^{n+1}(\overline{E}, \overline{\boldsymbol{X}}, ?) \cap \mathbb{S}^{n+1}(\overline{E}, 0)$ consists of only one matrix, which will be denoted as $\overline{\boldsymbol{X}}^{\cap 0}$. Similarly to Lemma 2, we can show the maximum-determinant property of $\overline{\boldsymbol{X}}^{\cap 0}$.

Theorem 5. *The matrix $\overline{\boldsymbol{X}}^{\cap 0}$ has the maximum determinant among all possible matrix completion of $\overline{\boldsymbol{X}} \in \overline{\mathbb{T}}_+^{n+1}(\overline{E})$, i.e.,*

$$\det_{\mathbb{T}}\overline{\boldsymbol{X}}^{\cap 0} = \max\left\{\det_{\mathbb{T}}\widehat{\boldsymbol{X}} \;\middle|\; \widehat{\boldsymbol{X}} \in \widehat{\mathbb{T}}_+^{n+1}(\overline{E}, \overline{\boldsymbol{X}})\right\}.$$

Proof. For $\widehat{\boldsymbol{X}} \in \widehat{\mathbb{T}}_+^{n+1}(\overline{E}, \overline{\boldsymbol{X}})$, we have

$$
\begin{aligned}
\det_{\mathbb{T}}\widehat{\boldsymbol{X}} &= \Pi_{(i,j)\in J}(\widehat{X}_{ii}\widehat{X}_{jj} - \widehat{X}_{ij}^2) \\
&= \Pi_{(i,j)\in\overline{E}}(\widehat{X}_{ii}\widehat{X}_{jj} - \widehat{X}_{ij}^2)\Pi_{(i,j)\notin\overline{E}}(\widehat{X}_{ii}\widehat{X}_{jj} - \widehat{X}_{ij}^2) \\
&= \Pi_{(i,j)\in\overline{E}}(\overline{X}_{ii}\overline{X}_{jj} - \overline{X}_{ij}^2)\Pi_{(i,j)\notin\overline{E}}(\overline{X}_{ii}\overline{X}_{jj} - \widehat{X}_{ij}^2) \\
&\leq \Pi_{(i,j)\in\overline{E}}(\overline{X}_{ii}\overline{X}_{jj} - \overline{X}_{ij}^2)\Pi_{(i,j)\notin\overline{E}}(\overline{X}_{ii}\overline{X}_{jj}) \\
&= \Pi_{(i,j)\in\overline{E}}(\overline{X}_{ii}^{\cap 0}\overline{X}_{jj}^{\cap 0} - (\overline{X}_{ij}^{\cap 0})^2)\Pi_{(i,j)\notin\overline{E}}(\overline{X}_{ii}^{\cap 0}\overline{X}_{jj}^{\cap 0} - (\overline{X}_{ij}^{\cap 0})^2) \\
&= \det_{\mathbb{T}}\overline{\boldsymbol{X}}^{\cap 0}.
\end{aligned}
$$

For the third equality, we have used $\widehat{X}_{ij} = \overline{X}_{ij}$ for $(i,j) \in \overline{E} \cup D$. The fourth equality is derived from $\overline{\boldsymbol{X}}^{\cap 0} \in \mathbb{S}^{n+1}(\overline{E}, 0)$. Note that $\overline{X}_{ii}\overline{X}_{jj} - \widehat{X}_{ij}^2 \geq 0$ holds for $(i,j) \notin \overline{E}$ since $\widehat{X}_{ij} \in R_{ij}(\overline{\boldsymbol{X}})$. $\qquad\square$

Theorems 4 and 5 show that an optimal solution of (3) as $\overline{\boldsymbol{X}}^{\cap 0}$ can be obtained by solving (5) and substituting 0 in the elements in $J\backslash\overline{E}$. In view of computational time, we can expect that (5) is more efficient than (3), since the number of second-order constraints in (5) is less than that of (3). In Section 4, numerical results will be presented to verify the expected efficiency.

Compared with the matrix completion in SDP, the matrix completion in SOCP has two advantages: First, the constraints in (5) is determined by the aggregate sparsity pattern $\overline{E}$, therefore, the chordal extension $\widehat{E}$ is not necessary. Since $\widehat{E}$ has more edges than $\overline{E}$ if $\overline{E}$ is not chordal, it is a clear advantage for (5). Secondly, the completion procedure in SOCP is to substitute 0 in the elements in $J\backslash\overline{E}$. This procedure is much simpler than the procedure of SDP where we need to compute matrices recursively with the maximal cliques $\{C_1, \ldots, C_p\}$.

3.2 Dual problems

We investigate the relation between (3) and (5) with their dual problems. The dual of (3) is

$$\text{maximize: } \xi$$

$$\text{subject to: } \boldsymbol{Q}_0 + \sum_{k=1}^{m} \boldsymbol{Q}_k y_k - \boldsymbol{H}_0 \xi - \sum_{(i,j) \in J} \boldsymbol{W}^{ij} = \boldsymbol{O}$$

$$\boldsymbol{y} \in \mathbb{R}_+^m, \ \xi \in \mathbb{R}$$

$$\boldsymbol{W}^{ij} \in \mathbb{S}_+^{n+1,\{i,j\}} \text{ for } (i,j) \in J, \tag{7}$$

where

$$\mathbb{S}_+^{n+1,\{i,j\}} := \left\{ \boldsymbol{W} \in \mathbb{S}_+^{n+1} \mid W_{kl} = 0 \text{ for } (k,l) \notin \{(i,i),(i,j),(j,i),(j,j)\} \right\}.$$

Similarly, the dual of (5) is:

$$\text{maximize: } \xi$$

$$\text{subject to: } \boldsymbol{Q}_0 + \sum_{k=1}^{m} \boldsymbol{Q}_k y_k - \boldsymbol{H}_0 \xi - \sum_{(i,j) \in \overline{E}} \boldsymbol{W}^{ij} - \sum_{i \in V_e} w_i \boldsymbol{e}_i \boldsymbol{e}_i^T = \boldsymbol{O}$$

$$\boldsymbol{y} \in \mathbb{R}_+^m, \ \xi \in \mathbb{R}$$

$$\boldsymbol{W}^{ij} \in \mathbb{S}_+^{n+1,\{i,j\}} \text{ for } (i,j) \in \overline{E}$$

$$w_i \geq 0 \text{ for } i \in V_e, \tag{8}$$

where $\boldsymbol{e}_i$ is the ith unit vector in $\mathbb{R}^{n+1}$.

In Theorem 4, we have discussed the relation between the primal problems (3) and (5). We show the relation between the two dual problems in the following theorem.

Theorem 6. *Each feasible solution of (7) (or (8)) can be converted to a feasible solution of (8) (or (7), respectively) while maintaining the objective value.*

Proof. Suppose that $\widehat{\boldsymbol{y}}, \widehat{\xi}$ and $\widehat{\boldsymbol{W}}^{ij}$ for $(i,j) \in J$ is a feasible solution of (7). Let $\widehat{\boldsymbol{Q}} := \boldsymbol{Q}_0 + \sum_{k=1}^{m} \boldsymbol{Q}_k \widehat{y}_k - \boldsymbol{H}_0 \widehat{\xi}$, then it holds that $\widehat{\boldsymbol{Q}} \in \mathbb{S}^{n+1}(\overline{E}, 0)$. This indicates that if $(i,j) \in J \backslash \overline{E}$, then $\widehat{\boldsymbol{W}}^{ij}$ is a diagonal matrix. Therefore, we can find appropriate $\overline{\boldsymbol{W}}^{ij} \in \mathbb{S}_+^{n+1,\{i,j\}}$ for each $(i,j) \in \overline{E}$ such that $\widehat{\boldsymbol{Q}} -$

13

$\sum_{(i,j)\in\overline{E}}\overline{\boldsymbol{W}}^{ij}$ is a diagonal matrix and nonnegative diagonal elements appear only in V_e, by distributing the nonnegative diagonal elements of $\widehat{\boldsymbol{W}}^{ij}$ of $(i,j)\in J\backslash\overline{E}$ to some $\widehat{\boldsymbol{W}}^{ij}$ of $(i,j)\in\overline{E}$. If we denote the nonnegative diagonal elements by $\widehat{w}_i$ for $i\in V_e$, we know that the solution with $\widehat{\boldsymbol{y}},\widehat{\xi},\overline{\boldsymbol{W}}^{ij}$ for $(i,j)\in\overline{E}$ and $\widehat{w}_i$ for $i\in V_e$ is a feasible solution of (8).

The opposite direction can be derived from the fact that

$$\left\{\sum_{(i,j)\in\overline{E}}\boldsymbol{W}^{ij}+\sum_{i\in V_e}w_i\boldsymbol{e}_i\boldsymbol{e}_i^T\;\middle|\;\begin{array}{l}\boldsymbol{W}^{ij}\in\mathbb{S}_+^{n+1}\cap\mathbb{S}^{n+1,\{i,j\}}\text{ for }(i,j)\in\overline{E},\\ w_i\geq 0\text{ for }i\in V_e\end{array}\right\}$$

is a subset of

$$\left\{\sum_{(i,j)\in J}\boldsymbol{W}^{ij}+\sum_{i\in V_e}w_i\boldsymbol{e}_i\boldsymbol{e}_i^T\;\middle|\;\boldsymbol{W}^{ij}\in\mathbb{S}_+^{n+1,\{i,j\}}\text{ for }(i,j)\in J\right\}.$$

In fact, if $\overline{\boldsymbol{W}}^{ij}$ for $(i,j)\in\overline{E}$ and $\overline{w}_i$ for $i\in V_e$ are a feasible solution of (8), a feasible solution of (7) can be constructed by assigning the values as

$$\boldsymbol{W}^{ij}=\left\{\begin{array}{ll}\overline{\boldsymbol{W}}^{ij}+\overline{w}_i\boldsymbol{e}_i\boldsymbol{e}_i^T & \text{if }(i,j)\in\overline{E}\text{ and }i\in V_e\\ \overline{\boldsymbol{W}}^{ij} & \text{if }(i,j)\in\overline{E}\text{ and }i\notin V_e\\ \overline{w}_i\boldsymbol{e}_i\boldsymbol{e}_i^T & \text{if }(i,j)\notin\overline{E}\text{ and }i\in V_e\\ \boldsymbol{O} & \text{if }(i,j)\notin\overline{E}\text{ and }i\notin V_e,\end{array}\right.\tag{9}$$

and keeping the values of the other variables. $\qquad\square$

A direct consequence of Theorem 6 is that we can solve (8) instead of solving (7) and recover the optimal solution of (7) using (9).

4   Numerical experiments

Numerical experiments on exploiting the sparsity of the SOCP relaxation were conducted with two test problems, the lattice and pooling problem. For the test problems, the SOCP relaxation (3) is known to provide the same optimal value as that of the SDP relaxation (2) due to the structure of the data matrices. The aim of the numerical experiments is to observe how much computational time can be reduced by exploiting the aggregate

Table 1: Relaxation problems in the primal form (P)

F-SOCP (P)	the full SOCP relaxation (3) formulated as (P)
S-SOCP (P)	the sparse SOCP relaxation (3) formulated as (P)
F-SDP (P)	the full SDP relaxation (2) formulated as (P)
S-SDP (P)	the sparse SDP relaxation formulated as (P)

sparsity of the SOCP relaxation for the same optimal values by the SDP and SOCP relaxation.

The numerical experiments were performed using MATLAB R2018a on a MacBook Pro with an Intel Core i7 processor (2.8GHz) and 16 GB memory space. The relaxation problems of the lattice and pooling problem were solved by SeDuMi [32] and MOSEK version 8.1.0.72 [25], respectively. SeDuMi could not handle the pooling problem as it required too much memory. To test with SeDuMi and MOSEK, the SDP and SOCP relaxations of the test problems should be converted into the following input format:

$$(P) \quad \text{minimize:} \quad \boldsymbol{c}^\top \boldsymbol{x} \qquad (D) \quad \text{maximize:} \quad \boldsymbol{b}^\top \boldsymbol{y}$$
$$\text{subject to:} \quad \mathcal{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in \mathcal{K} \qquad \text{subject to:} \quad \boldsymbol{c} - \mathcal{A}^\top \boldsymbol{y} \in \mathcal{K}.$$

Here, $\mathcal{K}$ stands for a Cartesian product of a linear cones, second-order cones and positive semidefinite cones. $\mathcal{A}$ is a linear map and its adjoint operator is denoted with $\mathcal{A}^\top$. Note that the SDP relaxation problems ((2) and (4)) and the SOCP relaxation problems ((3) and (5)) can be formulated as either (P) or (D).

To denote which of (P) or (D) is used, we use the notation described in Table 1 for the relaxation problems in the primal form (P). Similarly, for the dual form (D), F-SOCP (D), S-SOCP (D), F-SDP (D), and S-SDP (D) are used. We mention that S-SDP (P) and S-SDP (D), which employ the chordal sparsity discussed in Section 2, can be obtained by executing SparseCoLO [12].

4.1 The lattice problem

We first describe how to generate a QCQP for the lattice problem with the size $n = n_L^2$ where n_L is a positive integer. Consider

$$\text{minimize:} \quad \boldsymbol{x}^\top \boldsymbol{P}_0 \boldsymbol{x}$$
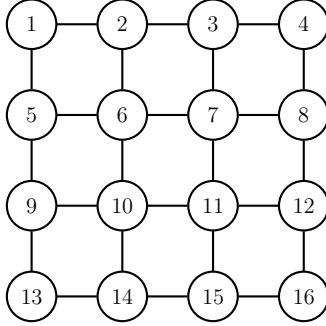$$\text{subject to:} \quad \boldsymbol{x}^\top \boldsymbol{P}_k \boldsymbol{x} + r_k \leq 0 \ (k = 1, ..., m). \tag{10}$$

Figure 1: A lattice graph for $n_L = 4$.

Here $\boldsymbol{x} \in \mathbb{R}^n$ is the decision variable. For the coefficient matrices $\boldsymbol{P}_0, \boldsymbol{P}_1, \ldots,$ $\boldsymbol{P}_m \in \mathbb{S}^n$, a lattice graph $G(V_L, E_L)$ is considered with the vertex set $V_L = \{1, 2, \ldots, n_L^2\}$ and the edge set

$$
\begin{aligned}
E_L \;=\; & \{((i_L - 1)n_L + j_L, (i_L - 1)n_L + (j_L + 1))|i_L = 1, 2, ...n_L, j_L = 1, 2, ..., n_L - 1\} \\
& \cup \{((i_L - 1)n_L + j_L, i_L n_L + j_L)|i_L = 1, 2, ..., n_L - 1, j_L = 1, 2, ..., n_L\}.
\end{aligned}
$$

Figure 1 illustrates the lattice graph $G(V_L, E_L)$ with $n_L = 4$. Test instances with the lattice graph were also used in [35].

For $k = 0, 2, \ldots, m$, $[\boldsymbol{P}_k]_{ij}$ was generated randomly in the interval $[-1, 0]$ for each $(i, j) \in E_L$, and the diagonal elements $[\boldsymbol{P}_k]_{ii}$ was generated randomly in the interval $[-1, 1]$ for each $i = 1, \ldots, n+1$. For $k = 1$, randomly generated number in the interval $(0, 1]$ was used as $[\boldsymbol{P}_1]_{ii}$ for $i = 1, \ldots, n+1$. The other elements in $\boldsymbol{P}_0, \ldots, \boldsymbol{P}_m$ were set to zeros. For $r_1, \ldots, r_k$, negative random values were used so that $\boldsymbol{x} = \boldsymbol{0}$ could be feasible for the lattice QCQP (10). The feasible region of (10) is bounded since $\boldsymbol{P}_1$ is a positive diagonal matrix. Note that the assumption in Theorems 3.4 and 3.5 of [19] is satisfied by the non-positivity of the off-diagonal elements, thus we can obtain the exact optimal value of the lattice QCQP (10) by the SDP relaxation (2) or the SOCP relaxation (3).

In Table 2, we compare F-SOCP (P) and S-SOCP (P) to see the effect of exploiting sparsity in the SOCP relaxation formulated as (P). The first and second columns show the number of constraints and variables, respectively. The third and fourth columns report the computational time in seconds and the objective values obtained.

We also observe in Table 2 that F-SOCP (P) and S-SOCP (P) attained the same objective value as shown in Theorem 4, but S-SOCP (P) is much

16

Table 2: Comparing F-SOCP (P) and S-SOCP (P)

Number of constraints (m)	Number of variables ($n = n_L^2$)	CPU time (s)		Objective value	
		F-SOCP (P)	S-SOCP (P)	F-SOCP (P)	S-SOCP (P)
15	64^2	13.78	0.20	-556.56	-556.56
20	81^2	88.91	0.12	-415.75	-415.75
30	100^2	315.61	0.20	-996.37	-996.37
40	121^2	1292.46	0.33	-2481.84	-2481.84
50	144^2	4594.44	0.50	-1157.12	-1157.12
120	169^2	15397.13	1.71	-3308.35	-3308.35

Table 3: Numerical results on F-SDP (P), S-SDP (P), and S-SOCP (P)

Number of constraints (m)	Number of variables ($n = n_L^2$)	CPU time (s)			Objective value		
		F-SDP (P)	S-SDP (P)	S-SOCP (P)	F-SDP (P)	S-SDP (P)	S-SOCP (P)
1000	81^2	4.79	6.77	6.85	-396.15	-396.15	-396.15
1000	100^2	6.22	8.94	9.08	-669.20	-669.20	-669.20
1500	121^2	19.12	26.79	27.89	-2412.98	-2412.98	-2412.98
1500	144^2	23.05	32.77	35.47	-1108.57	-1108.57	-1108.57
2000	169^2	51.86	65.82	83.68	-3043.22	-3043.22	-3043.22
2000	196^2	53.92	75.04	104.88	-5401.28	-5401.28	-5401.28
2500	225^2	106.07	148.99	210.25	-9921.85	-9921.85	-9921.85

more efficient for solving the SOCP relaxation problems than F-SOCP (P). The number of second-order cones for the constraint $\boldsymbol{X} \in \mathbb{T}_+^{n+1}$ in (3) is $\frac{n(n-1)}{2} = \frac{n_L^2(n_L^2-1)}{2}$, which is the fourth order of n_L. In contrast, the number of second-order cones for $\boldsymbol{X} \in \overline{\mathbb{T}}_+^{n+1}(\overline{E})$ in (5) is $2n_L(n_L - 1)$, which is the quadratic order of n_L. We see that the difference in the numbers of second-order cones resulted in the large difference in CPU time shown in Table 2.

Table 3 displays the numerical results on F-SDP (P), S-SDP (P) and S-SOCP (P) for larger n_L than Table 2, and it shows that F-SDP (P) is the most efficient. When we formulate the problem with $m = 2500$ into (P), the sizes of $\mathcal{A}$ in F-SDP (P), S-SDP (P), and S-SOCP (P) are 2500×53125, 3026×13322 and 3340×3985, respectively. In S-SDP (P), the average size of maximal cliques is 23.42 and the number of equality constraints to be added for the overlaps between cliques (the cardinality of U in (4)) is 526. The number of the added equality constraints becomes computational burden when solving S-SDP (P), though we can reduce the number of variables compared with F-SDP (P). Next, S-SOCP (P) includes many second-order cones. The dimension of each second-order cone in S-SOCP (P) is three. We mention that handling many cones of small size is known to be inefficient. Therefore, in the standard form of (P), S-SOCP (P) is less efficient than

Table 4: Computational time (in seconds) for the dual (D)

Number of constraints (m)	Number of variables ($n = n_L^2$)	CPU time (s)			
		F-SDP (D)	S-SDP (D)	F-SOCP (D)	S-SOCP (D)
9000	64^2	39.01	12.07	32.37	2.30
10000	81^2	179.98	21.77	80.60	3.70
15000	100^2	724.58	38.03	185.11	7.32
20000	121^2	2373.0	83.77	484.28	14.89
30000	144^2	6515.9	244.11	1323.8	31.99
60000	169^2	19868.23	537.48	3436.7	76.50

Table 5: Numerical comparison between F-SDP (P) and S-SOCP (D)

Number of constraints (m)	Number of variables ($n = n_L^2$)	CPU time (s)		Objective value	
		F-SDP (P)	S-SOCP (D)	F-SDP (P)	S-SOCP (D)
3000	81^2	120.60	1.32	-363.22	-363.22
5000	100^2	591.40	2.63	-616.87	-616.87
5000	121^2	663.15	3.79	-2318.78	-2318.78
10000	144^2	5029.6	9.15	-1091.03	-1091.03
10000	169^2	5412.5	11.81	-2901.12	-2901.12
15000	196^2	19033	23.45	-4204.64	-4204.64

F-SDP (P).

Now, we consider the dual (D). In Table 4, we compare four relaxations, F-SDP (D), S-SDP (D), F-SOCP (D) and S-SOCP (D). As seen in Tables 2 and 3, the objective values from the four problems are the same, so the objective values are not shown in Table 4. The computational advantage by exploiting the aggregate sparsity in S-SOCP over F-SOCP is not as large as Table 2. However, we can still observe the improved efficiency by exploiting the aggregate sparsity.

When the problem with $m = 60000$ is formulated as the dual (D), the sizes of $\mathcal{A}$ in F-SDP (D), S-SDP (D), F-SOCP (D) and S-SOCP (D) are 14365×88651, 3559×67204, 14365×102757 and 481×61105, respectively. It is faster to solve S-SDP (D) and S-SOCP (D), which include relatively small-sized $\mathcal{A}$, than F-SDP (D) and F-SOCP (D). We also see that F-SOCP (D) consumes less computational time than F-SDP (D), despite larger-sized $\mathcal{A}$ in F-SOCP (D) than that of F-SDP (D).

Finally, we discuss which relaxation method for the lattice QCQP is the most efficient. From Tables 3 and 4, we observe that F-SDP (P) and S-SOCP (D) outperform other methods in (P) and (D), respectively. Table 5

summarizes the computational time to compare F-SDP (P) and S-SOCP (D). In all cases, S-SOCP (D) is remarkably faster than F-SDP (P).

We have not included the computational time for the completion procedure for S-SDP in all tables. As discussed in Section 3.1, the SOCP relaxation (5) does not require a completion procedure, but it can generate the optimal solution of the lattice QCQP (10).

4.2 The pooling problem

In this subsection, we present numerical results on the relaxation problems from the pooling problem studied in Kimizuka et al. [21]. The pooling problem with time discretization is a mixed-integer nonconvex QCQP, and it is shown as NP-hard [3]. Kimizuka et al. [21] solved the SDP, SOCP and LP relaxation problems for the QCQP and applied a rescheduling method to the solution obtained from the relaxation problem to generate a good approximate solution for the pooling problem. They used the fact that all the diagonal elements of the data matrices $\boldsymbol{P}_0, \ldots, \boldsymbol{P}_m$ are always zeros in the QCQP (1) obtained from the pooling problem, and proved that the SDP, SOCP and LP relaxations of the QCQP provided the same optimal value. In [21], they generated the SOCP and LP relaxation problems using SPOTless [2].

For our numerical experiments, we used the 10 test instances from [21], and the description of each test instance can be found in [21]. In Table 6, we compare CPU time consumed by F-SDP (D), S-SDP (D), F-SOCP (D), and S-SOCP (D). F-SDP (D) and F-SOCP (D) correspond to the SDP and SOCP relaxation problems used in [21]. S-SDP (D) was obtained using SparseCoLO, while S-SOCP (D) was formulated using the aggregate sparsity based on (5). The primal standard form (P) of the pooling problem is not compared, as SPOTless [2] always generate full SDP or SOCP relaxations in (P), and extracting the original aggregate sparsity from the resulting SDP or SOCP problems was difficult.

In Table 6, we first observe from the results on Instances 1 and 3 that it was much more expensive to solve F-SDP (D) than the other three relaxations. Furthermore, the other instances could not be solved due to out of memory. By exploiting the chordal sparsity, S-SDP (D) successfully solved most instances except for the largest two instances where out of memory occurred. Compared to the SDP relaxations, the SOCP relaxations can be solved very efficiently. F-SOCP (D), which does not exploit any sparsity, is

Table 6: Numerical results on the pooling problem (OOM stands for out of memory.)

Instance	Number of variables (n)	CPU time (s)			
		F-SDP (D)	S-SDP (D)	F-SOCP (D)	S-SOCP (D)
1	138	768.24	0.82	0.68	0.51
2	298	OOM	1.00	1.33	0.62
3	176	5024.0	1.15	0.85	0.68
4	356	OOM	1.67	1.96	0.60
5	222	OOM	2.22	1.01	0.57
6	446	OOM	4.01	2.80	0.60
7	1228	OOM	19.98	21.66	0.86
8	1228	OOM	21.83	21.02	1.04
9	1334	OOM	OOM	28.44	4.84
10	1616	OOM	OOM	40.90	6.53

competitive with S-SDP (D) in terms of computational time. In addition, F-SOCP (D) did not take much memory to solve Instances 9 and 10.

For the performance of S-SOCP (D), we see that S-SOCP (D) is 3-6 times faster than F-SOCP (D). In particular, the number of second-order cones for Instance 10 in F-SOCP (D) was 1306536, while S-SOCP (D) had only 70464 second-order cones. This difference resulted in much shorter CPU time for S-SOCP (D).

5 Conclusion

We have presented a method to exploit the aggregate sparsity in the SOCP relaxation. From the numerical experiments in Section 4, we have observed that the proposed approach is very efficient in solving the SOCP relaxation. In addition, the proposed method can obtain the optimal solution with the simple matrix completion that attains the maximum determinant in the SOCP relaxation, as in the matrix completion for the SDP relaxation.

For future work, exploiting sparsity in the SDSOS relaxation [1] can be considered to obtain an approximate solution of polynomial optimization problems fast. Since the rows and columns of matrices involved in the relaxation problems for polynomial optimization problems usually have some

structure, the proposed approach in this paper can be applied to improve the computational efficiency. In addition, the hierarchy that employs SOCP for polynomial optimization problems in [22] will be also our interest.

References

[1] A. A. Ahmadi and A. Majumdar. DSOS and SDSOS optimization: LP and SOCP-based alternatives to sum of squares optimization. In *48th annual conference on information sciences and systems (CISS)*, pages 1–5. IEEE, 2014.

[2] A. A. Ahmadi and A. Majumdar. Software for DSOS and SDSOS optimization, 2014. `https://github.com/spot-toolbox/spotless` (Accessed on September 25th, 2019.).

[3] M. Alfaki. *Models and solution methods for the pooling problem*. PhD thesis, The University of Bergen, 2012.

[4] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming*, 95(1):3–51, 2003.

[5] K. M. Anstreicher. Recent advances in the solution of quadratic assignment problems. *Mathematical Programming*, 97(1):27–42, Jul 2003.

[6] E. Balas. Projection, lifting and extended formulation in integer and combinatorial optimization. *Annals of Operations Research*, 140(1):125, 2005.

[7] X. Bao, N. V. Sahinidis, and M. Tawarmalani. Semidefinite relaxations for quadratically constrained quadratic programming: A review and comparisons. *Mathematical Programming*, 129(1):129, May 2011.

[8] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Transactions on Sensor Networks*, 2(2):188–220, 2006.

[9] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[10] A. d'Aspremont and S. Boyd. Relaxations and randomized methods for nonconvex QCQPs. EE392o Class Notes, Stanford University, 2003.

[11] A. De Maio, Y. Huang, D. P. Palomar, S. Zhang, and A. Farina. Fractional QCQP with applications in ML steering direction estimation for radar detection. *IEEE Transactions on Signal Processing*, 59(1):172–185, 2010.

[12] K. Fujisawa, S. Kim, M. Kojima, Y. Okamoto, and M. Yamashita. User's manual for SparseCoLO: Conversion methods for sparse conic-form linear optimization problems. *Research Report B-453, Dept. of Math. and Comp. Sci. Japan, Tech. Rep.*, pages 152–8552, 2009.

[13] M. Fukuda, M. Kojima, K. Murota, and K. Nakata. Exploiting sparsity in semidefinite programming via matrix completion I: General framework. *SIAM Journal on Optimization*, 11(3):647–674, 2001.

[14] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[15] R. Grone, C. R. Johnson, E. M. Sá, and H. Wolkowicz. Positive definite completions of partial hermitian matrices. *Linear Algebra and its Applications*, 58:109–124, 1984.

[16] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[17] Z.-H. Huang, D. Sun, and G. Zhao. A smoothing newton-type algorithm of stronger convergence for the quadratically constrained convex quadratic programming. *Computational Optimization and Applications*, 35(2):199–237, 2006.

[18] S. Kim and M. Kojima. Second order cone programming relaxation of nonconvex quadratic optimization problems. *Optimization Methods and Software*, 15(3-4):201–224, 2001.

[19] S. Kim and M. Kojima. Exact solutions of some nonconvex quadratic optimization problems via SDP and SOCP relaxations. *Computational Optimization and Applications*, 26(2):143–154, 2003.

[20] S. Kim, M. Kojima, M. Mevissen, and M. Yamashita. Exploiting sparsity in linear and nonlinear matrix inequalities via positive semidefinite matrix completion. *Mathematical programming*, 129(1):33–68, 2011.

[21] M. Kimizuka, S. Kim, and M. Yamashita. Solving pooling problems with time discretization by LP and SOCP relaxations and rescheduling methods. *Journal of Global Optimization*, 75(3):631–654, 2019.

[22] X. Kuang, B. Ghaddar, J. Naoum-Sawaya, and L. F. Zuluaga. Alternative sdp and socp approximations for polynomial optimization. *EURO Journal on Computational Optimization*, 7(2):153–175, 2019.

[23] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0–1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

[24] R. P. Mason and A. Papachristodoulou. Chordal sparsity, decomposing SDPs and the Lyapunov equation. In *2014 American Control Conference*, pages 531–537. IEEE, 2014.

[25] Mosek ApS. The MOSEK optimization toolbox for MATLAB manual, 2015.

[26] T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of turán. *Canadian Journal of Mathematics*, 17:533–540, 1965.

[27] K. Nakata, K. Fujisawa, M. Fukuda, M. Kojima, and K. Murota. Exploiting sparsity in semidefinite programming via matrix completion II: Implementation and numerical results. *Mathematical Programming*, 95(2):303–327, 2003.

[28] A. Nemirovskii and K. Scheinberg. Extension of Karmarkar's algorithm onto convex quadratically constrained quadratic problems. *Mathematical Programming*, 72(3):273–289, 1996.

[29] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.

[30] T. Nishi. A semidefinite programming relaxation approach for the pooling problem. Master's thesis, Kyoto University, 2010.

[31] S. Safarina, S. Moriguchi, T. J. Mullin, and M. Yamashita. Conic relaxation approaches for equal deployment problems. *To appear in Discrete Applied Mathematics*, 2019.

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

[33] L. Vandenberghe and M. S. Andersen. *Chordal graphs and semidefinite optimization*, volume 1. Now Publishers, Inc., 2015.

[34] M. Yamashita, T. J. Mullin, and S. Safarina. An efficient second-order cone programming approach for optimal selection in tree breeding. *Optimization Letters*, 12(7):1683–1697, 2018.

[35] M. Yamashita and K. Nakata. Fast implementation for semidefinite programs with positive matrix completion. *Optimization Methods and Software*, 30(5):1030–1049, 2015.