Minimum rank positive semidefinite matrix completion with chordal sparsity pattern

Xin Jiang

1 Introduction

In semidefinite programming (SDP), one minimizes a linear objective function of a symmetric $n \times n$ matrix variable X subject to a set of linear equality constraints and a matrix inequality constraint:

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & X \succeq 0, \end{array} \tag{1}$$

where C, $A_1, \ldots, A_m$ are all symmetric $n \times n$ matrices. Here $X \succeq 0$ means that the symmetric matrix X is positive semidefinite. The semidefinite programming has been extensively studied since the 1990s. Subsequent researches in semidefinite programming have shown the wide variety of application in convex optimization. In this thesis we will be interested in its usefulness in semidefinite relaxation techniques for nonconvex quadratic optimization.

1.1 Semidefinite relaxation of quadratic optimization

Since the 1990s, semidefinite relaxation techniques have been widely studied and applied to many fields. It was first introduced in combinatorial optimization [GW95, Ali95], and then widely used in polynomial optimization [Par03, Las09, Las15]. It is also a well-studied, powerful approximation method for various quadratic optimization problems [Luo03] including waveform design in radar [MNH$^+$09, MNH$^+$08], phase unwrapping [JJZQJ09], sensor localization [BY04]. In most applications mentioned above, the problem is modeled as a nonconvex quadratically constrained quadratic program (QCQP), *i.e.*,

$$\begin{array}{ll} \text{minimize} & x^T C x + d^T x \\ \text{subject to} & x^T A_i x = b_i, \quad i = 1, \ldots, m, \end{array} \tag{2}$$

where the variable is $x \in \mathbf{R}^n$ and the given matrices C and A_i's are general real symmetric matrices. It is a nonconvex problem due to the indefiniteness of matrix C as well as the nonlinear equality constraints. As an example, the famous two-way partition problem is a nonconvex QCQP, *i.e.*,

$$\begin{array}{ll} \text{minimize} & x^T C x \\ \text{subject to} & x_i^2 = 1, \quad i = 1, \ldots, n, \end{array} \tag{3}$$

The problem (3) is known as an NP-hard problem, but it is the optimization model for many real-life applications like multi-in-multi-out (MIMO) detection. Hence, the semidefinite relaxation

is introduced to approximate the above QCQP (2). The fundamental step is to transform the quadratic form into an inner product of matrices, *i.e.*,

$$x^T C x = \mathbf{tr}(x^T C x) = \mathbf{tr}(C x x^T)$$
$$x^T A_i x = \mathbf{tr}(x^T A_i x) = \mathbf{tr}(A_i x x^T).$$

The **tr** operator means the trace of a matrix. Then by introduction of variable $X = xx^T$, we get an equivalent matrix optimization problem:

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & X \succeq 0, \quad \mathbf{rank}\ X = 1, \end{array} \tag{4}$$

where the variable $X \in \mathbf{R}^{n \times n}$ is symmetric and positive semidefinite. But the problem (4) is still difficult to solve, mainly due to the nonconvex rank constraint $\mathbf{rank}\ X = 1$. The only difference between the nonconvex problem (4) and the standard form SDP (5) is the rank constraint. Hence, the standard form SDP (1) can be attained by simply dropping the rank constraint, *i.e.*,

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & X \succeq 0. \end{array} \tag{5}$$

The resulting problem (5) is thus convex, and can be solved efficiently by many solvers [Bor99, Stu98, GB14]. However, the penalty is that the optimal solution $X^\star$ for (5) is not guaranteed to be optimal for (4). If $X^\star$ is a rank-one matrix, then we can construct $x^\star \in \mathbf{R}^n$ with $X^\star = x^\star (x^\star)^T$, which is optimal for (4). On the other hand, if the rank of $X^\star$ is larger than one, then the problem remains that how we can extract a vector $\tilde{x}$ from $X^\star$ to approximate the optimal solution for (4). In many optimization problems, there are a set of optimal solutions, and the interior point methods typically give an optimal solution with maximal rank. Hence, this relaxation technique often fails in reconstructing an optimal solution for (4).

1.2 Sparse semidefinite optimization

To solve a semidefinite programming problem, we can exploit the structure of the coefficients. Sometimes, the coefficient matrices C and A's are sparse, *i.e.*, a large portion of elements in the matrix are required to be zero. Furthermore, in many applications they share the same *sparsity pattern*, formally defined in Section 2.1. Some examples are semidefinite relaxations of combinatorial graph optimization problems [Ali95, GW95] and eigenvalue optimization problems for symmetric matrices associated with graphs [BDX04]. Semidefinite optimization or semidefinite relaxation techiques are also extensively used for sensor localization problems within large-scale networks [BY04, BLT$^+$06, BLWY06] and many machine learning applications [WS04, WSZS07, XSB06]. The graph structure of these applications is naturally inherited by the SDP [AVD10]. In addition, SDP formulation of problems involving matrix norms and robust quadratic optimization are found to share the same sparsity pattern too. In certain situations, SDP (5) can be transformed into the following sparse SDP [FKMN01, NFF$^+$03, Sun15, SAV13, VA14]

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & X \succeq_{\mathrm{c}} 0, \end{array} \tag{6}$$

where the matrix variable X is symmetric and sparse. The inequality in (6) means that the symmetric matrix X has a positive semidefinite completion, *i.e.*, we can construct a positive semidefinite matrix by replacing zero elements in X with other numbers. The equivalence between (5) and (6) can be easily established. If X is optimal in (6), then any positive semidefinite completion of X is optimal in (5). Conversely, if X is optimal for (5), then the projection of X on the sparse pattern is optimal in (6).

By formulating SDP (5) as a sparse semidefinite optimization problem (6), we can reduce the dimension of the problem from $n(n+1)/2$ (for $X \in \mathbf{S}^n$) to a number equal to the nonzero elements in the sparsity pattern. Although by transforming SDP (5) into sparse SDP (6), we change the types of inequality constraints used in the problem, it is reasonable to expect reduction of the computing complexity of interior-point methods for these sparse problems. We refer readers to Section 2.1 for detailed discussion.

After we solve the sparse SDP (6), the remaining problem is to find a positive semidefinite completion for the solution, which is also optimal for (5). As discussed in Section 1.1, among all positive semidefinite completions, we are most interested in the one with the lowest rank. In general, it is difficult to determine the minimum rank of positive semidefinite completions, not to say to construct a matrix with certain rank. However, it turns out that if the variables and coefficients share a *chordal* sparsity pattern, which is defined in Section 2.2, then we are able to determine the minimum rank of positive semidefinite completion [Dan92], as well as to construct a completion with the minimum rank [Sun15]. If the minimum rank is one, then we can get an optimal solution for the nonconvex SDP (4), as well as the nonconvex QCQP (2). Moreover, in many applications the coefficient matrices are proven to be chordal, or can be extended into a chordal sparsity pattern (see Section 2.4). Hence, the procedure of finding the minimum rank completion is extremely useful with chordal sparsity pattern, and we show an example in the optimal power flow problem in later chapters. The results show that by the minimum rank completion algorithm, one can substantially reduce the rank of the optimal solution for the optimal power flow problem.

1.3 Outline

The remainder of the thesis are organized as follows. Chapter 2 provides the necessary background for semidefinite optimization with sparsity pattern and preliminaries for understanding chordal sparsity. The algorithm for minimum rank positive semidefinite matrix completion with chordal sparsity pattern is introduced in Section 3. In Section 4, we introduce the optimal power flow model and see how the algorithm in Section 3 can be applied to the OPF problem.

1.4 Notation

The notation $\mathbf{S}^n$ is the set of symmetric matrices of order n. $\mathbf{S}^n_+ = \{X \in \mathbf{S}^n \mid X \succeq 0\}$ and $\mathbf{S}^n_{++} = \{X \in \mathbf{S}^n \mid X \succ 0\}$ are the sets of positive semidefinite and definite matrices of order n, respectively. The set $\mathbf{S}^n_{++}$ is the interior of $\mathbf{S}^n_+$, *i.e.*, $\mathbf{S}^n_{++} = \mathbf{int}\ \mathbf{S}^n_+$.

Similarly, $\mathbf{H}^n$ is the set of Hermitian matrices of order n. $\mathbf{H}^n_+ = \{X \in \mathbf{H}^n \mid X \succeq 0\}$ is the set of positive semidefinite and definite matrices of order n, with $\mathbf{H}^n_{++} = \{X \in \mathbf{H}^n \mid X \succ 0\}$ its interior. The trace notation $\mathbf{tr}(XY) = \sum_{i,j=1}^n X_{ij}Y_{ij}$ denotes the inner product of symmetric or Hermitian matrices of order n, according to the context.

The notation K^* denotes the dual cone of a cone K, *i.e.*,

$$K^* = \{Y \mid \mathbf{tr}(XY) \succeq 0 \text{ for all } X \in K\}.$$

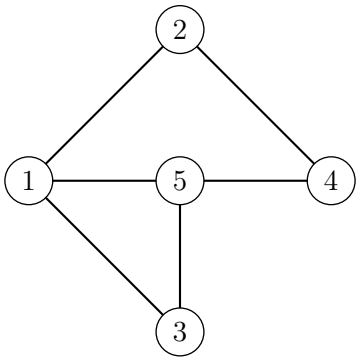


Figure 1: Sparsity pattern for matrix (7).

2 Semidefinite optimization with chordal sparsity pattern

Semidefinite programs (SDPs) have been a hot topic for over two decades [VB96, LMS+10], but exploiting sparsity in SDP still remains a great challenge in this field [FKMN01, Haz08]. In this section, we introduce the methods to represent sparsity pattern in matrices, and show the equivalence between the primal-dual pair of standard SDP and the primal-dual pair of a sparse matrix conic program. Next we introduce the concept of chordal sparsity. With the properties in chordal graphs, we show how to decompose the above sparse matrix conic programs into several small-size dense SDPs, which can be solved efficiently in existing SDP solvers.

2.1 Sparsity in semidefinite optimization

2.1.1 Symmetric sparsity pattern

An undirected graph is a pair $G = (V, E)$, with V a finite set and E a subset of $\{\{v, w\} \mid v, w \in V\}$. The elements of V are called *vertices* or *nodes* of the graph, the elements of E its edges. The notation $\{v, w\}$ is used to emphasize that there is no self-loop in edges, *i.e.*, $v \neq w$, and that the order of the vertices does not matter, *i.e.*, $\{v, w\} = \{w, v\}$. We use the structure of an undirected graph $G = (V, E)$ to describe the sparsity pattern of a symmetric matrix. Suppose the vertex set in graph G is $V = \{1, 2, \ldots, n\}$, and then the edge set is represented as

$$E \subseteq \{\{i, j\} \mid i, j \in V\}.$$

A symmetric matrix X has sparsity pattern E if its entries $X_{ij} = 0$ when $i \neq j$ and $\{i, j\} \notin E$. But the other entries in X, *i.e.*, the diagonal entries X_{ii} and off-diagonal elements X_{ij} with $\{i, j\} \in E$, may or may not be nonzero. We use the notation $\mathbf{S}^n_E$ to denote the set of $n \times n$ symmetric matrices with sparsity pattern E. Fig. 1 shows the corresponding graph structure of the sparsity pattern of matrix $X \in \mathbf{S}^5_E$:

$$X = \begin{bmatrix} X_{11} & X_{21} & X_{31} & 0 & X_{51} \\ X_{21} & X_{22} & 0 & X_{42} & 0 \\ X_{31} & 0 & X_{33} & 0 & X_{53} \\ 0 & X_{42} & 0 & X_{44} & X_{54} \\ X_{51} & 0 & X_{53} & X_{54} & X_{55} \end{bmatrix}. \tag{7}$$

We can use different sparsity patterns and sparse graphs to describe the same matrix. For any edge set with $E \subset E'$, the graph $G' = (V, E')$ can also be used to describe the sparsity pattern of X. The sparsity pattern E' is called an *embedding* or *extension* of sparsity pattern E.

We define the *dense principal submatrix* of a sparse matrix X as a principal submatrix of A with nonzero entries. A dense principal submatrix corresponds to a complete subgraph of G. For example, a dense principal submatrix of (7) is

$$\begin{bmatrix} X_{11} & X_{31} & X_{51} \\ X_{31} & X_{33} & X_{53} \\ X_{51} & X_{53} & X_{55} \end{bmatrix}. \tag{8}$$

The corresponding complete subgraph has vertex set $W = \{1, 3, 5\}$. But note that the above concepts of "nonzero" means "possibly nonzero". As in the above example, it means that there exists some matrices in $\mathbf{S}^n_E$ with nonzero entries in the nine positions in (8). The same idea applies to "dense principal submatrix". A *clique* W of a graph G is a subset of vertices $W \subseteq V$ that induces a maximal complete subgraph[1]. So a clique in a sparse graph is a maximal dense prinicipal submatrix in the corresponding sparse matrix.

2.1.2 Sparse matrix cones

In this section, we introduce two sparse matrix cones and show that they are dual cones of each other. Then in the following section we show how the standard form semidefinite programs (SDP) can be reformulated into semidefinite programs with sparsity pattern.

The positive semidefinite matrices with sparsity pattern E form a convex cone

$$\mathbf{S}^n_{E,+} = \mathbf{S}^n_E \cap \mathbf{S}^n_+ = \{X \in \mathbf{S}^n_E \mid X \succeq 0\}.$$

This cone is closed and convex since it is an intersection of a subspace and a closed convex cone. It is also pointed, *i.e.*, without any lines, since this case in which $X \succeq 0$ and $-X \succeq 0$ is $X = 0$. It also has a nonempty interior $\mathbf{S}^n_{E,++} = \mathbf{S}^n_E \cap \mathbf{S}^n_{++}$ where $\mathbf{S}^n_{++}$ is the set of all positive definite matrices. Hence, it is a proper cone.

We are also interested in those matrices in $\mathbf{S}^n_E$ that have a positive semidefinite completion. It turns out that the set

$$\mathbf{S}^n_{E,\mathrm{c}+} = \{\mathbf{proj}_E(X) \mid X \in \mathbf{S}^n_+\}$$

is also a convex proper cone. It is convex since it is the projection of a closed convex cone $\mathbf{S}^n_+$ on the subspace $\mathbf{S}^n_E$.

Next we prove that it is pointed. Assume $A \in \mathbf{S}^n_{E,\mathrm{c}+}$ and $-A \in \mathbf{S}^n_{E,\mathrm{c}+}$. Then there exists $X, Y \succeq 0$ such that $A = \mathbf{proj}_E(X) = -\mathbf{proj}_E(Y)$. Hence, we have $\mathbf{proj}_E(X + Y) = 0$. But $X + Y \succeq 0$. Therefore, we must have $X = Y = 0$.

To prove the closedness of the positive semidefinite completable cone, we need the following theorem.

Theorem 1 ([Roc70], theorem 9.1)**.** If C is a closed convex cone and M a linear mapping satisfying

$$\mathcal{N}(M) \cap C = \{0\},$$

[1]Some authors define a clique as any subset of nodes that induces a complete subgraph, maximal or not. In this thesis, we use the term *complete subgraph* for this purpose, and reserve the term clique to denote maximal complete subgraphs.

where $\mathcal{N}(M)$ indicates the nullspace of M, then $M(C)$ is closed.

The closedness of $\mathbf{S}^n_{E,\mathrm{c}+}$ is established if we apply this theorem to $C = \mathbf{S}^n_+$ and $M = \mathbf{proj}_E$. Hence, cone $\mathbf{S}^n_{E,\mathrm{c}+}$ is a proper cone.

Now we show that the positive semidefinite completable cone $\mathbf{S}^n_{E,\mathrm{c}+}$ and the positive semidefinite sparse matrix cone $\mathbf{S}^n_{E,+}$ are actually dual cones of each other in $\mathbf{S}^n_E$, under the inner product trace, *i.e.*, **tr**. First we show that the dual of $\mathbf{S}^n_{E,\mathrm{c}+}$ is $\mathbf{S}^n_{E,+}$.

$$\begin{aligned}(\mathbf{S}^n_{E,\mathrm{c}+})^* &= \{Y \in \mathbf{S}^n_E \mid \mathbf{tr}(XY) \geq 0,\ \forall X \in \mathbf{S}^n_{E,\mathrm{c}+}\} \\ &= \{Y \in \mathbf{S}^n_E \mid \mathbf{tr}(\mathbf{proj}_E(\hat{X})Y) \geq 0,\ \forall \hat{X} \in \mathbf{S}^n_+\} \\ &= \{Y \in \mathbf{S}^n_E \mid \mathbf{tr}(\hat{X}Y) \geq 0,\ \forall \hat{X} \succeq 0\} \\ &= \mathbf{S}^n_{E,+}.\end{aligned}$$

To prove the other direction, we borrow the result from page 121 in [Roc70] that the dual of the dual of a convex cone is the closure of the cone. Hence, we have

$$(\mathbf{S}^n_{E,+})^* = ((\mathbf{S}^n_{E,\mathrm{c}+})^*)^* = \mathbf{cl}\ \mathbf{S}^n_{E,\mathrm{c}+} = \mathbf{S}^n_{E,\mathrm{c}+}$$

since the proper cone $\mathbf{S}^n_{E,\mathrm{c}+}$ is closed itself. From the above proofs we can see that the pair of dual relationship holds for any sparsity pattern E.

2.1.3 Semidefinite programs with sparsity pattern

With the properties of matrix sparsity pattern as well as two sparse matrix cones, we are now ready to consider the semidefinite programs with sparsity pattern. In general, exploiting sparsity in SDPs remains a great challenge. One of the difficulties lies in the dense pattern of the variable $X \in \mathbf{S}^n$ in the standard form SDP

$$\begin{array}{ll}\text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_iX) = b_i, \quad i = 1, \ldots, m \\ & X \succeq 0.\end{array} \tag{9}$$

Even when the coefficients A_i, C share the same sparsity pattern, the sparse property cannot be generalized to the feasible set of the problem (9), not to say the optimal solution. Exploiting sparsity in the dual form could be more straightforward as the slack variable $S \in \mathbf{S}^n$ in the dual problem

$$\begin{array}{ll}\text{maximize} & b^Ty \\ \text{subject to} & \displaystyle\sum_{i=1}^m y_iA_i + S = C \\ & S \succeq 0\end{array} \tag{10}$$

shares the same (*aggregate*) sparsity pattern as the coefficients A_i and C. In most SDP solvers, however, solving the dual problem usually involves the computation of the inverse of S, which is dense in most cases.

Some papers [FKMN01, NFF+03, Bur03, SV04] transform the problems into several optimization problems in the subspace of sparse symmetric matrices. Specifically, if coefficient matrices A_i and

C share the same sparse pattern E, *i.e.*, A_i, $C \in \mathbf{S}^n_E$, then the standard primal-dual SDP pair (9 and 10) is equivalent to the following pair of problems

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & X \succeq_{\mathrm{c}} 0, \end{array} \tag{11}$$

with primal variable $X \in \mathbf{S}^n_E$, and

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to} & \displaystyle\sum_{i=1}^{m} y_i A_i + S = C \\ & S \succeq 0, \end{array} \tag{12}$$

with dual variables $y \in \mathbf{R}^m$ and $S \in \mathbf{S}^n_E$. The inequality in the primal (11) $X \succeq_{\mathrm{c}} 0$ means that the symmetric matrix X with sparsity pattern E has a positive semidefinite completion, *i.e.*, X is the projection of a positive semidefinite matrix on the cone $\mathbf{S}^n_E$

$$X \succeq_{\mathrm{c}} 0 \quad \Longleftrightarrow \quad X \in \mathbf{S}^n_{E,\mathrm{c}+}.$$

Also note that the slack variable S in (12) is in the positive semidefinite cone $\mathbf{S}^n_{E,+}$ that we defined in the previous subsubsection.

It is clear to see the equivalence between (9-10) and (11-12). If X is optimal in (11), then any positive semidefinite completion is optimal in (9). Conversely, if X is optimal in (9), then $\mathbf{proj}_E(X)$ its projection on $\mathbf{S}^n_E$, is optimal in (11). By adding the constraint of sparsity pattern, we reduce the dimension of the problem from $n(n+1)/2$ to the number of lower-triangular zeros in the pattern E. Furthermore, if the sparsity pattern is *chordal*, some more efficient methods have been introduced to deal with the primal-dual pair of the problems (11-12), resulting in a more efficient and reliable solver.

2.2 Chordal sparsity

2.2.1 Basic graph theory

Before we formally define chordal graphs, we will briefly review some basic concepts in graph theory and give our notations (see *e.g.*, [Die12] or [Gol04] for more details). For an undirected graph $G = (V, E)$, two vertices v and w are *adjacent* if $\{v, w\} \in E$. A *path* between v and $w \neq v$ is a sequence of distinct vertices $(v_0 = v, v_1, \cdots, v_k = w)$ with $\{v_i, v_{i+1}\} \in E$ for $i = 0, \ldots, k-1$. A *cycle* is a path with no repeated vertices other than the ending vertices (*i.e.*, $v_0 = v_k$). A *chord* of a cycle is an edge that joins two non-consecutive vertices of the cycle.

Two vertices are said to be *connected* if there exists at least one path that has the two vertices as its end points. A graph is *connected* if any two vertices inside the graph are connected. A maximal connected subgraph of G is a *component* of G.

The *neighborhood* or *adjacency set* $\mathrm{adj}(v)$ of a vertex v is the set of vertices adjacent to it:

$$\mathrm{adj}(v) = \{w \mid \{v, w\} \in E\}.$$

The *degree* of vertex v is the number of vertices adjacent to it, *i.e.*, $\deg(v) = |\mathrm{adj}(v)|$.

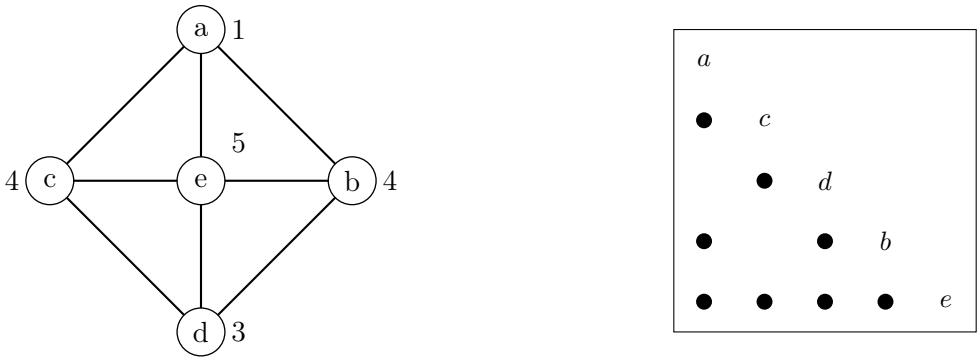


Figure 2: Ordered undirected graph.

Ordered undirected graphs

An *ordering* of an undirected graph $G = (V, E)$ is a numbering of its vertices, or equivalent, a bijection from $\{1, 2, \ldots, n\}$ to V if $|V| = n$. We may also use the notation $\sigma : \{1, 2, \ldots, n\} \to V$ to denote a sequence of vertices $\sigma = (\sigma(1), \ldots, \sigma(n))$. We refer to $\sigma^{-1}(v)$ as the *index* of vertex v. An *ordered graph*, denoted as $G_\sigma = (V, E, \sigma)$, is an undirected graph $G = (V, E)$ with an ordering σ. Then we use the generalized inequality notation $v \prec_\sigma w$ for $\sigma^{-1}(v) < \sigma^{-1}(w)$ and $v \preceq_\sigma w$ for $\sigma^{-1}(v) \leq \sigma^{-1}(w)$. The subscripts in $\prec_\sigma$ and $\preceq_\sigma$ are sometimes omitted if the ordering σ is clear from the context. To visualize an ordered graph G_σ, we use a lower-triangular array (or table) with the vertex names on the diagonal and a dot in the entry (i, j) $(i > j)$ if the vertices $\sigma(i)$ and $\sigma(j)$ are adjacent. An example is shown in Fig. 2. Two types of monotone neighborhoods are defined in ordered graphs:

$$\begin{aligned} \mathrm{adj}^+ &= \{w \in \mathrm{adj}(v) \mid w \succ v\}, \\ \mathrm{adj}^- &= \{w \in \mathrm{adj}(v) \mid v \prec v\}. \end{aligned}$$

The first set is called the *higher neighborhood* or *higher adjacency set* of v while the second set the *lower neighborhood* or *lower adjacency set*. Correspondingly, the *higher degree* and *lower degree* are defined by

$$\deg^+ = |\mathrm{adj}^+(v)|, \qquad \deg^- = |\mathrm{adj}^-(v)|.$$

It is easy to identify the higher and lower adjacency set in the matrix representation form of an ordered graph. The indices of the elements of $\mathrm{adj}^-(v)$ are the column indices of the entries in row $\sigma^{-1}(v)$ of the matrix; the indices of the elements of $\mathrm{adj}^+(v)$ are the row indices of the entries in column $\sigma^{-1}(v)$. In Fig. 2, $\mathrm{adj}^-(d) = \{c\}$ and $\mathrm{adj}^+(d) = \{b, c\}$.

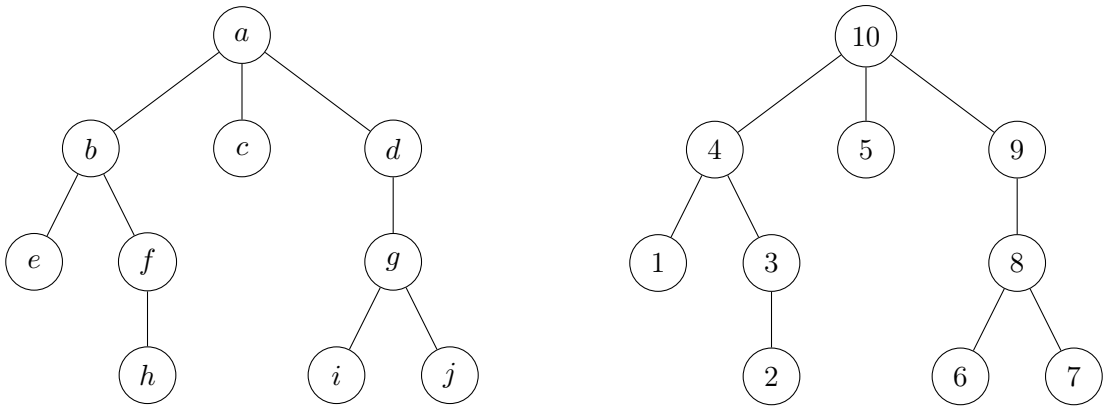
We use the notation $\mathrm{col}(v)$ and $\mathrm{row}(v)$ for the *closed* monotone neighborhoods

$$\mathrm{col}(v) = \{v\} \cup \mathrm{adj}^+(v), \qquad \mathrm{row}(v) = \{v\} \cup \mathrm{adj}^-(v).$$

Trees

An undirected graph is a *tree* if it is connected and does not contain any cycles. A *rooted tree* is a tree with a special vertex designated as the root of the tree, as in Fig. 3a.

An important property in trees is that a unique path exists between any two vertices of a tree. If v is a non-root vertex and w is the first vertex on the path from v to the root, then w is the



(a) Rooted tree. (b) A postordering of (a)

Figure 3: Tree and one postordering.

parent of v and v is called a *child* of w. The set of children of v is denoted $\mathrm{ch}(v)$. A childless vertex is called a *leaf* of the tree. We define the *parent function* $p : V \to V$ to denote this relationship. We assume the parent of the root is itself. And parents of degree k, with k a nonnegative integer, are defined recursively as

$$p^0(v) = v, \qquad p^{k+1}(v) = p(p^k(v)).$$

In Fig. 3a, we have $p(i) = g$, $p^2(i) = d$, and $p^3(i) = a$. Furthermore, if $w = p^k(v)$ for some $k \geq 0$ then w is an *ancestor* of v and v is a *descendant* of w. The ancestor or descendant is *proper* if $w \neq v$.

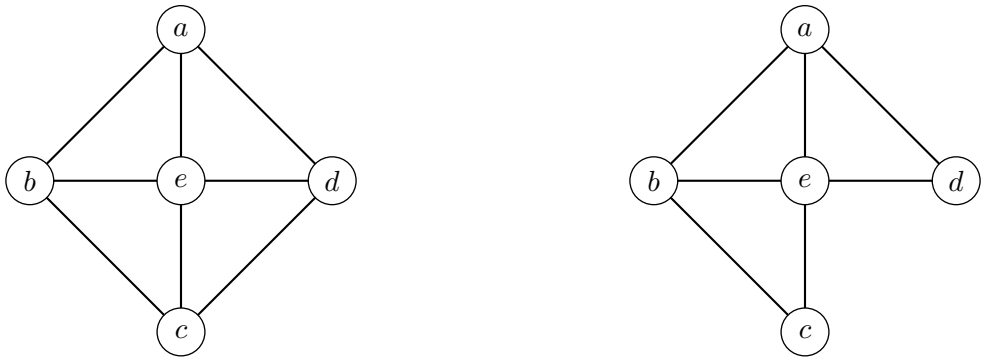
An ordering σ of a rooted tree is a *topological* ordering if $v \prec_\sigma p(v)$ for all non-root vertices v. This means the root vertex has an index n if $|V| = n$, and every other vertex has an index lower than its parent. A *postordering* is a topological ordering in which the descendants of a vertex are given consecutive numbers. A postordering can be generated by assigning the indices $n, n-1, \ldots, 1$ in decreasing order during a depth-first search traversal of the tree. Fig. 3b shows a possible postordering of Fig. 3a.

2.2.2 Chordal graphs

An undirected graph G is *chordal* if every cycle of length at least four has a chord.[2] A sparse pattern X is called chordal if its sparsity graph $G(X)$ is chordal.

An immediate conclusion from the definition is that subgraphs $G(W) = (W, E(W))$ of a chordal graph $G = (V, E)$ are chordal.

Some trivial examples of chordal graphs are the complete graphs, trees and forests (undirected graphs without any cycles). Here we see two other examples of chordal graphs, *i.e.*, k-trees and interval graphs.



(a) Non-chordal graph (b) Chordal graph

Figure 4: Examples of chordal and non-chordal graphs.

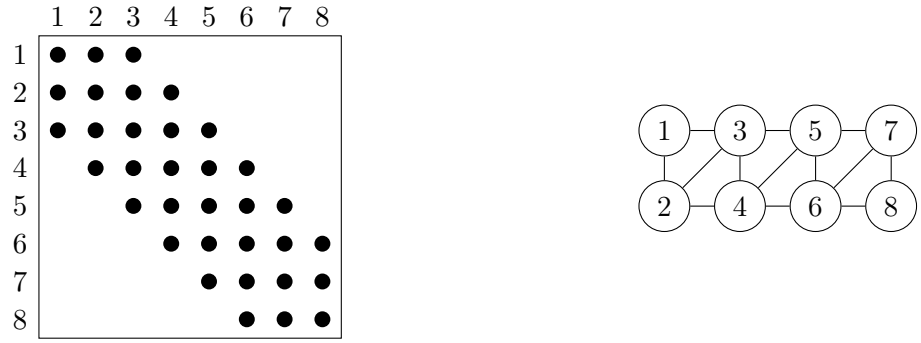


Figure 5: Band sparsity pattern and sparsity graph. It is a 2-tree.

k-trees

Obviously, trees are chordal graphs since there exists no cycles inside trees. The k-tree can be viewed as a generalization of a tree, and it can be defined recursively as follows [Ros70, Ros74]. A complete graph on k vertices is a k-tree. A k-tree on $n+1$ ($n \geq k$) vertices can be constructed from a k-tree on n vertices by adding a vertex adjacent to exactly k vertices that induce a complete subgraph. The band matrix in Fig. 5 is an example of 2-tree, and in general, the number k is half the bandwidth. An example of 2-tree is shown in Fig. 6, which corresponds to a block-arrow matrix with bandwidth 2. In general, the number k is the block-width in block-arrow matrix. When $k = 1$ in this case, the corresponding sparse graph is the so-called star graph, which is a tree with one root and n leaves, if it has $n + 1$ nodes.

2.2.3 Clique trees

A clique tree of a graph $G = (V, E)$ is a tree which has the cliques of G as its vertices. A clique tree T has the *induced subtree property* if for every $v \in V$, the cliques that contain v form a subtree (connected subgraph) R_v of T. An example is shown in Fig. 7[3]. It is shown that chordal graphs are

[2]In history, chordal graphs have been called *rigid-circuit graphs* [Dir61, FG65, Bun74], *triangulated graphs*[Ber86], and *acyclic graphs* [Lau96].

[3]The examples of clique tree and rooted clique tree are from [VA14].

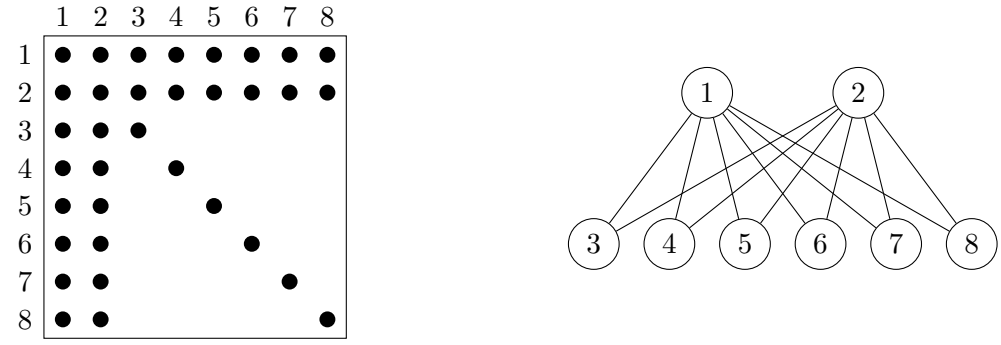


Figure 6: Block-arrow sparsity pattern and sparsity graph. It is a 2-tree.

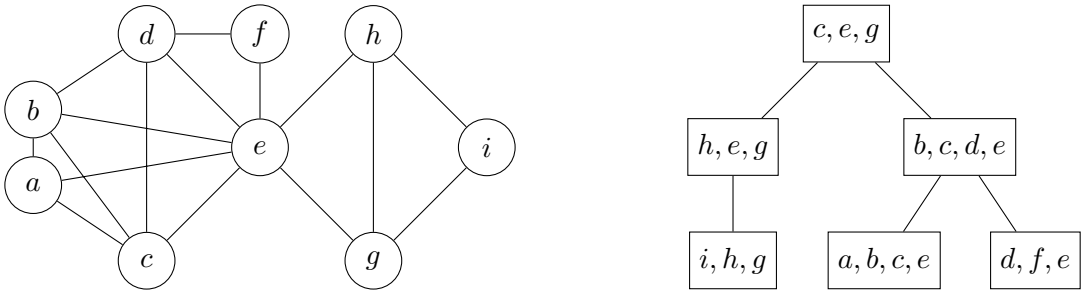


Figure 7: Clique tree.

exactly the graphs for which a clique tree with the induced subtree property exists [Bun74, Gav74]. Let T be a clique tree with the above induced subtree property, and we can arbitrarily pick a clique as the root of T to form a rooted clique tree. Let the function $p_{\mathrm{c}}(W)$ denote the parent of clique W. Then every non-root clique W can be partitioned into *clique separators* and *clique residuals*, *i.e.*,

$$\mathrm{sep}(W) = W \cap p_{\mathrm{c}}(W), \qquad \mathrm{res}(W) = W \backslash \mathrm{sep}(W).$$

We define $\mathrm{sep}(W) = \emptyset$ for the root clique W since it does not have a parent. From the definition of $\mathrm{sep}(W)$ and $\mathrm{res}(W)$ we can see that every vertex $v \in V$ belongs to exactly one clique residual $\mathrm{res}(W)$. For any vertex $v \in V$, the induced subtree R_v is rooted at $\mathrm{res}(W)$ with $v \in W$, and the other vertices are the cliques that contain vertex v as separators. Proofs of the above two properties can be found in the survey paper [VA14].

Fig. 8 shows an example of rooted clique tree of Fig. 7 with the clique $\{c, e, g\}$ is chosen as the root. The top half-block is the separator $\mathrm{sep}(W)$ while the bottom half is the residual $\mathrm{res}(W)$. For the clique $W = \{a, b, c, e\}$, its clique separator is $\mathrm{sep}(W) = \{b, c, e\}$ and clique residual $\mathrm{res}(W) = \{a\}$. Fig. 9 is the clique trees of band pattern and block arrow pattern in Fig. 5 and 6. Fig. 10 [VA14] is a useful chordal pattern with two overlapping diagonal blocks and its corresponding clique tree.

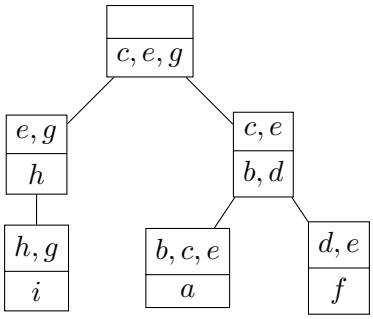


Figure 8: Rooted clique tree of the graphs in Fig. 7.

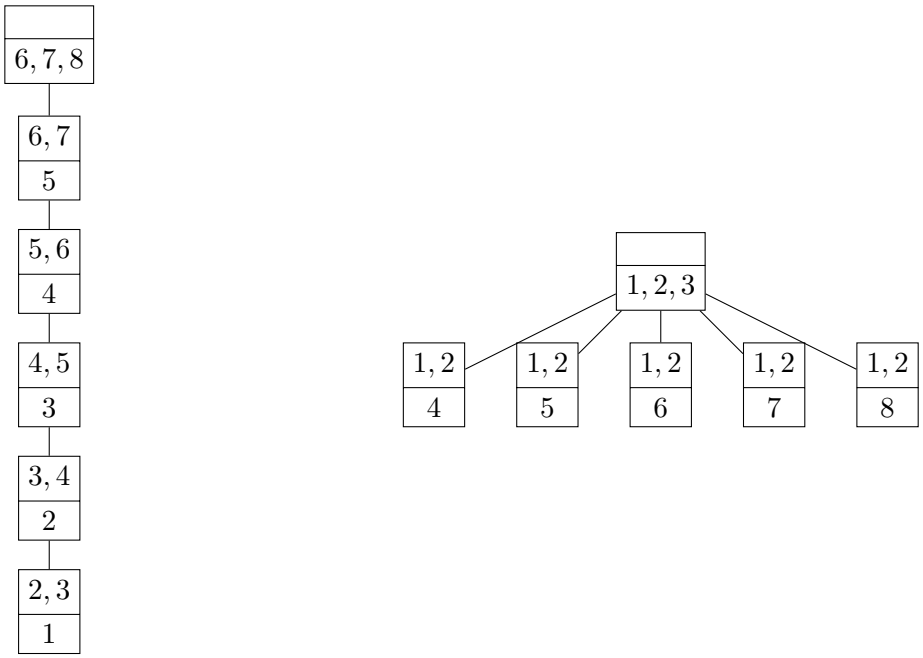


Figure 9: Rooted clique tree of Fig. 5 and 6.

2.3 Perfect elimination ordering

2.3.1 Filled graph and perfect elimination ordering

An ordered undirected graph $G_\sigma = (V, E, \sigma)$ is *filled* or *monotone transitive* if higher neighborhood of every vertex is complete, *i.e.*,

$$v, w \in \mathrm{adj}^+(u) \quad \Longrightarrow \quad \{v, w\} \in E.$$

It is easy to see from the definition that the graph $G = (V, E)$ is chordal. Suppose there exists a cycle with length greater than three in G. Consider the vertex u in this cycle with least order $\sigma^{-1}(u)$. We can always find two higher neighbors v, w of vertex u. Then by the definition of monotone transitivity, there must exist a chord between v and w. Hence, the vertex set $\{u, v, w\}$ forms a cycle of three.

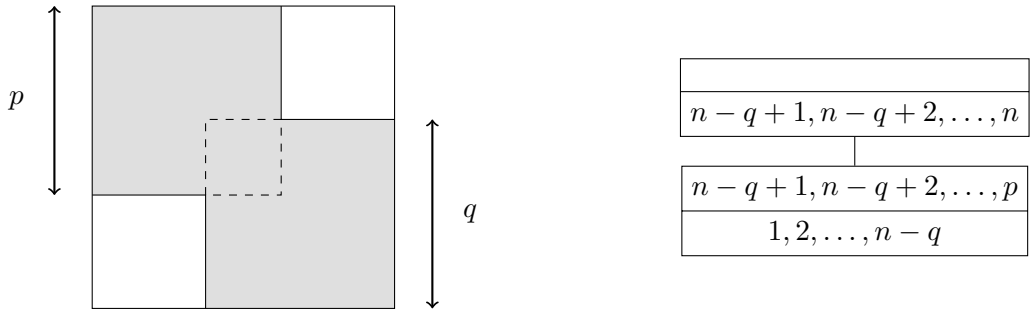


Figure 10: Chordal sparsity pattern with two overlapping diagonal blocks and associated clique tree.

An ordering σ of an undirected graph $G = (V, E)$ is called a *perfect elimination ordering* of G if the ordered graph $G_\sigma = (V, W, \sigma)$ is filled. The paper [FG65] has shown that a graph is chordal if and only if it has a perfect elimination ordering. Thus chordal graphs are sometimes called *perfect elimination graphs*. Rose, Tarjan, and Lueker [RTL76] first published a linear-time algorithm for constructing perfect elimination orderings. Later, Tarjan and Yannakakis [TY84] developed the well-known *maximum cardinality search (MCS)* algorithm to find a perfect elimination ordering for chordal graphs. With these algorithms, we can easily test whether a graph G is chordal by looking for a perfect elimination ordering of G. If the resulting ordering is a perfect elimination ordering, then G is chordal.

2.3.2 Elimination tree

Given a connected undirected graph $G = (V, E)$ with a perfect elimination ordering σ, we can derive an elimination tree of G. Without loss of generality we can assume G is connected, and we will use the assumption of connectivity throughout the section. The elimination tree of G_σ is a rooted tree with vertices V and root $\sigma(n)$. For vertex $v \in V$, its parent $p(v)$ in the elimination tree is the least ordered vertex in its higher neighborhood $\mathrm{adj}^+(v)$, *i.e.*,

$$p(v) = \mathrm{argmin}\{\sigma^{-1}(u) \mid u \in \mathrm{adj}^+(v)\}.$$

It is a valid definition since the set $\mathrm{adj}^+(v)$ is non-empty as long as v is not the root of the elimination tree. The proof of the above proposition can be found in the survey paper [VA14]. But note that we cannot reconstruct the structure of G by its elimination tree. Fig. 11 shows the sparse pattern with a perfect elimination ordering of Fig. 7 as well as associated elimination tree.

From monotone transitivity, the subgraph with vertices $\mathrm{col}(v) = \{v\} \cup \mathrm{adj}^+(v)$ is complete, and we see that higher neighbors of vertex v in G, *i.e.*, elements of $\mathrm{adj}^+(v)$, are ancestors of v in the elimination tree. In particular,

$$\mathrm{adj}^+(v) \subseteq \mathrm{col}(p(v)) = \{p(v)\} \cup \mathrm{adj}^+(p(v)) \tag{13}$$

for any non-root vertex v. Furthermore, if we apply (13) repeatedly until we reach the root of the elimination tree, we get

$$\mathrm{adj}^+(v) \subseteq \{p(v), p^2(v), \ldots, p^{\mathrm{lev}(v)}(v)\}, \tag{14}$$

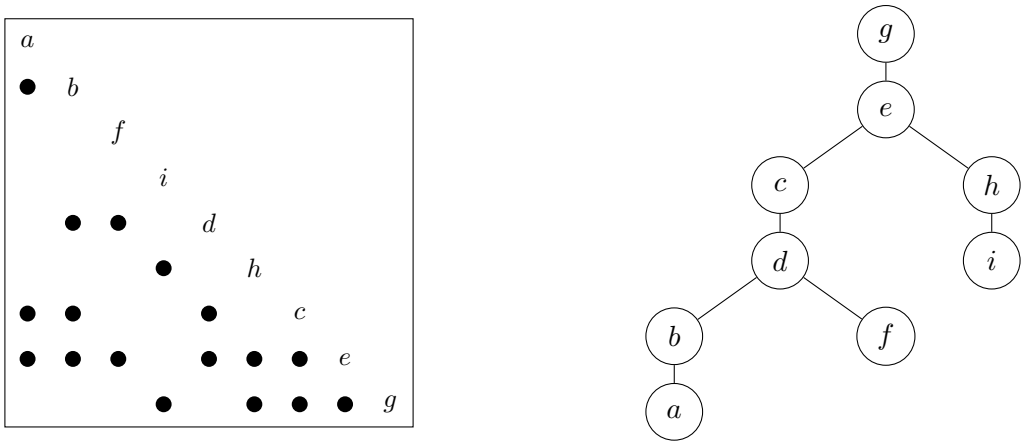


Figure 11: Sparse pattern with perfect elimination ordering and elimination tree of Fig. 7.

where $\mathrm{lev}(v)$ is the length of path from v to root, and $p^i(v) = p(p^{i-1}(v))$ is the parent of v of degree i. Or generally, for $k = 1, \ldots, \mathrm{lev}(v)$, we have

$$\mathrm{adj}^+(v) \subseteq \{p(v), p^2(v), \ldots, p^k(v)\} \cup \mathrm{adj}^+(p^k(v)). \tag{15}$$

We can also expand the elimination tree in a way similar to building the clique tree. In this case, the separator is $\mathrm{adj}^+(v)$ while the node v itself works as the residual. Actually, for every given v, vertices in $\mathrm{row}(v) = \{u \mid v \in \mathrm{col}(u)\}$ form a subtree of elimination tree. And this *row subtree* is proven to have the induced subtree property. As an example, Fig. 12 shows the expanded elimination tree of Fig. 11 as well as the row subtree of vertex e.

2.3.3 Maximal supernodal elimination tree

Since the higher degree of a vertex is $\deg^+(v) = |\mathrm{adj}^+(v)|$, from (15) we get

$$\deg^+(v) \leq k + \deg^+(p^k(v)), \quad k = 1, \ldots, \mathrm{lev}(v). \tag{16}$$

Two immediate properties are as follows: if the inequality (16) holds with equality for $k = j$, then it holds with equality for $1 \leq k \leq j$, as shown Fig. 13 with solid line denoting equality and $u = p^j(v)$. Conversely, if the inequality (16) is strict for $k = j + 1$, then it is strict for $j + 1 \leq k \leq \mathrm{lev}(v)$, as shown in Fig. 13 with dashed line denoting strict equality and $w = p^{j+1}(v)$.

For simplicity, we first assume that the vertex w has only one child u and $\deg(u) < \deg(w) + 1$, as shown in Fig. 13. Consider the clique W that contains the node w. Then w has the lowest order in clique W by our assumption. By monotone transitivity, $W \subseteq \mathrm{col}(w)$, and since W is a maximal complete subgraph, $W = \mathrm{col}(w)$. In this case, the vertex w is called the *representative vertex* of the clique $W = \mathrm{col}(w)$. The following theorem [PS90] shows that our original assumption that u is the only child of w can be generalized to all children of w have the inequality (16) holds with strict inequality:

Theorem 2. Let T be the elimination tree of a connected filled graph. A vertex v is a representative vertex if and only if

$$\deg^+(u) < \deg^+(w) + 1 \quad \forall u \in \mathrm{ch}(w), \tag{17}$$

where $\mathrm{ch}(v)$ denotes the set of children of v in T.

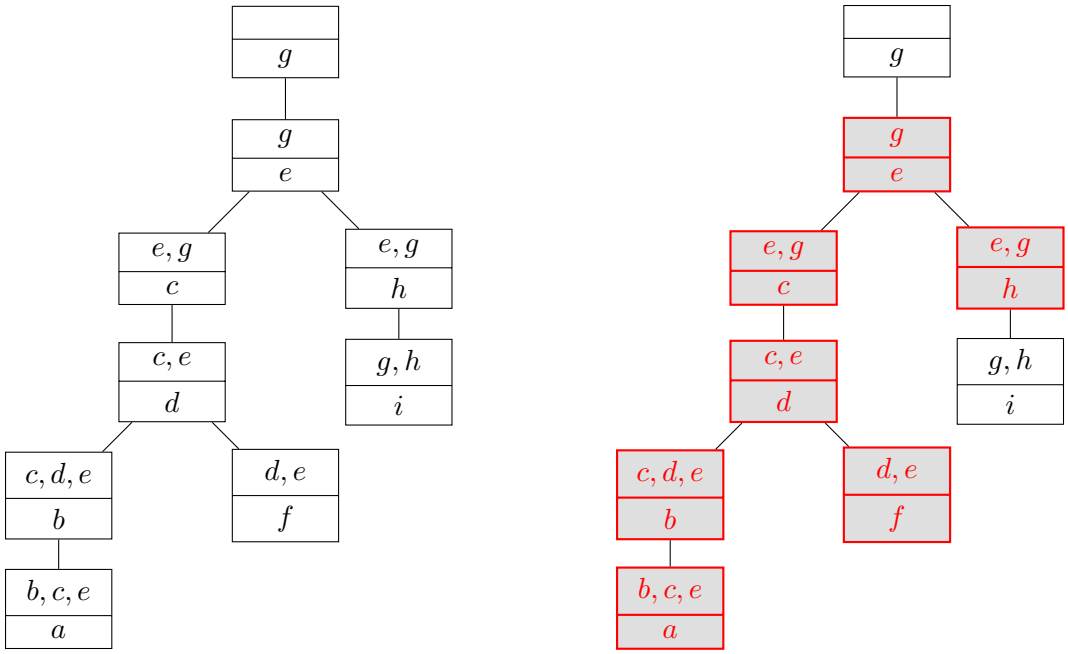


Figure 12: *Left.* Expanded elimination tree of Fig. 11. *Right.* The shaded vertices form the row subtree of vertex e with $\text{row}(e) = \{a, b, c, d, e, f, h\}$.

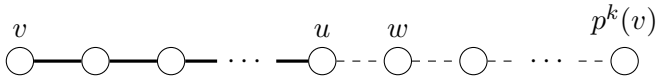


Figure 13: Illustration of inequality (16). The nodes represent v and its ancestors. Solid line means the inequality (16) holds with equality while dashed line means the inequality (16) is strict.

Thus we conclude that every clique can be represented as $W = \text{col}(w)$ where w is the vertex with the lowest ordering in clique W [FG65, Gav72, LPP89]. We can find a partition of vertex set V by cliques of G, or equivalently, by the representative vertices. Let V^c be the set of representative vertices. For every $v \in V^c$, the clique that contains node v can be represented as

$$\text{snd}(v) = \{v, p(v), \ldots, p^{n_v}(v)\} \tag{18}$$

where $n_v = |\text{snd}(v)| - 1$ with

$$\deg^+(v) = \deg^+(p^{n_v}(v)) + n_v. \tag{19}$$

The set $\text{snd}(v)$ is called the *maximal supernode* and the first vertex v is called the *representative vertex* of the supernode $\text{snd}(v)$ [VA14][4]. The partition of V is called the *maximal supernode partition*. Note that in general the maximal supernode partition is not unique, since any non-representative vertex u may have more than one child w with $\deg(w) = \deg(u) - 1$.

A *maximal supernodal elimination tree* T^c can be built according to a maximal supernode partition. The vertex set of T^c is the set the supernodes $\{\text{snd}(v) \mid v \in V^c\}$. The root of T^c is the supernode that contains the root of elimination tree $\sigma(n)$. For any other non-root supernode

[4] In [LPP89, PS90] they use the term $\text{new}(v)$.

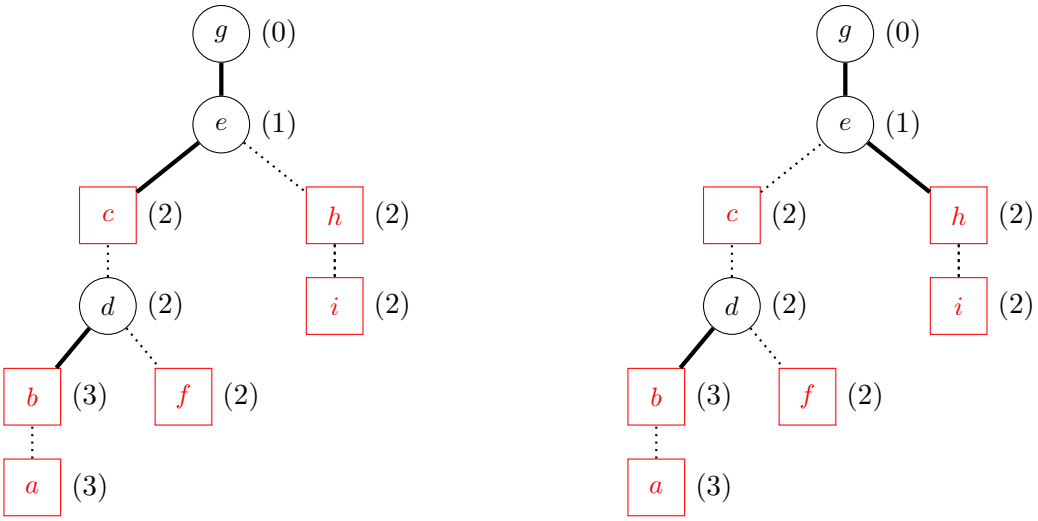


Figure 14: Two possible maximal supernode partition of Fig. 11. Representative vertices are shown in rectangles. For every representative node v, maximal supernode $\mathrm{snd}(v)$ is drawn in solid path starting at v. The label next to every node is the number of higher neighbors $\deg^+(v)$.

$\mathrm{snd}(v) = \{v, p(v), \ldots, p^{n_v}(v)\}$, we define the *first ancestor* [LPP89, PS90] as $a(v) = p^{n_v+1}(v)$, *i.e.*, the lowest ancestor of v that does not in the supernode $\mathrm{snd}(v)$. Then the parent of $\mathrm{snd}(v)$ in T^c is denoted as $\mathrm{snd}(q(v))$ with $a(v) \in \mathrm{snd}(q(v))$, *i.e.*, $q(v)$ is the representative vertex of the parent of $\mathrm{snd}(v)$ in T^c. With these notations, we have the following theorem [VA14]:

Theorem 3. Let T^c be a supernodal elimination tree of a filled graph $G_\sigma = (V, E, \sigma)$, based on a partition of V in maximal supernodes $\mathrm{snd}(v)$. Then for all non-root vertices $v \in V^c$,

$$\mathrm{col}(v) \backslash \mathrm{snd}(v) \subset \mathrm{col}(q(v)), \tag{20}$$

where $q(v)$ is the parent of v in T^c. For any given $u \in V$, the set $\{v \in V^c \mid u \in \mathrm{col}(v)\}$ is a subtree of T^c, with as root the vertex v that contains u in $\mathrm{snd}(v)$.

Lewis, Peyton, and Pothen [LPP89] have published an algorithm to generate a supernodal elimination tree T^c, the first ancestors $\mathrm{snd}(v)$, and the parent function $q(v)$. Later Pothen and Sun [PS90] have developed a simplified version of the algorithm with input of the elimination tree and the higher degrees of all vertices. Fig. 15 shows the corresponding supernodal elimination trees of the two maximal supernode partitions in Fig. 11.

We can build a clique tree given a supernodal elimination tree as well as the associated maximal supernode partition [VA14]. The clique separators and residuals are

$$\mathrm{res}(\mathrm{col}(v)) = \mathrm{snd}(v), \quad \mathrm{sep}(\mathrm{col}(v)) = \mathrm{col}(v) \backslash \mathrm{snd}(v),$$

and the parent function is defined as

$$p_c(\mathrm{col}(v)) = \mathrm{col}(q(v)).$$

The clique trees for the supernodal elimination trees in Fig. 15 are shown in Fig. 16.

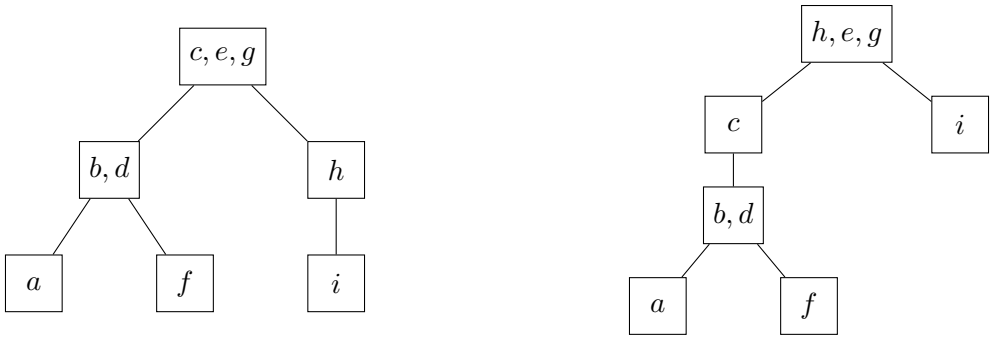


Figure 15: The supernodal elimination trees for the maximal supernode partition in Fig. 11.

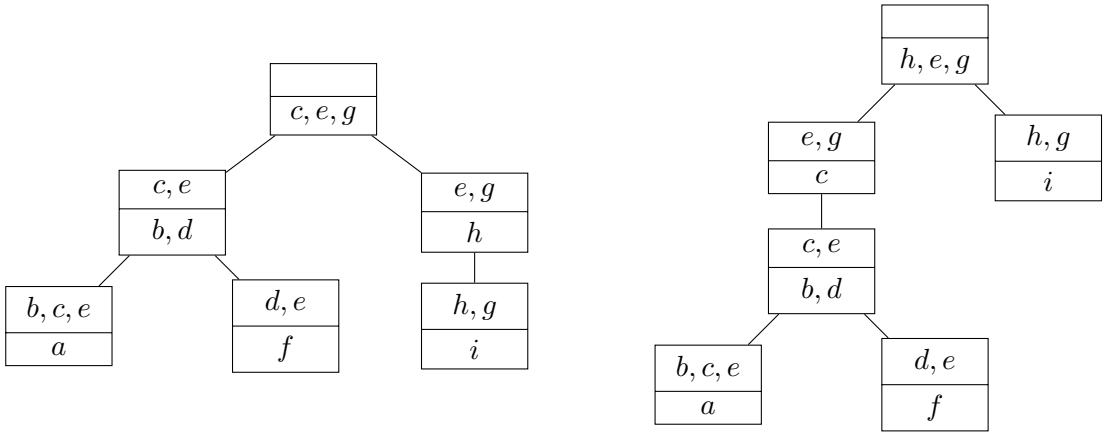


Figure 16: Two clique trees built from the maximal supernode partition 14. The first element of each separator is the first ancestor $a(v)$.

2.3.4 Postordered maximal supernodal elimination tree

Without loss of generality, we assume the order $\sigma = (1, 2, \ldots, n)$ is a perfect elimination ordering. The following theorem [VA14, Liu90] states that for any given supernodal elimination tree, we can always find a *topological postordering*:

Theorem 4. For a given supernode partition $\{\mathrm{snd}(v) \mid v \in V^{\mathrm{c}}\}$, we can always find an ordering σ with the following two properties.

- The elements of each supernode $\mathrm{snd}(v)$ are numbered consecutively: if $\sigma^{-1}(v) = i$ and $n_v = |\mathrm{snd}(v) - 1|$, then
$$\mathrm{snd}(v) = \{\sigma(i), \sigma(i+1), \ldots, \sigma(i+n_v)\}.$$
- The ordering σ defines a topological ordering of the representative vertices in the supernodal elimination tree $T^{\mathrm{c}} : v \prec_\sigma q(v)$ if $v \in V^{\mathrm{c}}$ and $q(v)$ is the parent of v in T^{c}.

Hence, we can assume the maximal supernode partition is in a topological postordering, *i.e.*, vertices in $\mathrm{snd}(v)$ are numbered consecutively and a leaf in the maximal supernodal elimination tree contains a set of lower-ordered vertices than its parent does. To illustrate, suppose supernode

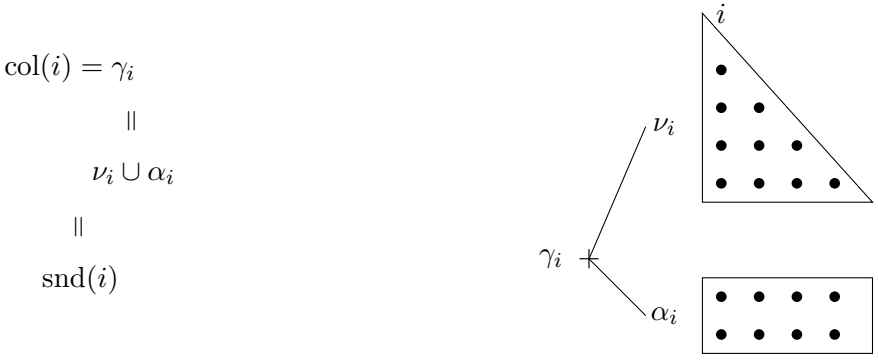


Figure 17: Illustration of index set in postordered maximal supernodal elimination tree.

$\mathrm{snd}(i)$ has $n_i + 1$ elements and its parent in the supernode elimination tree T^{c} is the supernode $\mathrm{snd}(q(i))$, then we have

$$\mathrm{snd}(i) = \{i, i+1, \ldots, i+n_i\} \quad \text{and} \quad i < q(i).$$

We refer to a maximal supernode elimination tree with certain topological postordering σ as *postordered maximal supernodal elimination tree*. Given a postordered supernodal elimination tree with a topological postordering σ, we define the following index sets [VA14]:

- For $i = 1, \ldots, n$, define the index set $\gamma_i = \mathrm{col}(i)$, *i.e.*, the index set γ_i contains the row indices of lower-triangular nonzero elements in column i.
- For $i \in V^{\mathrm{c}}$, define the index set $\nu_i = \mathrm{snd}(i)$, *i.e.*, $\nu_i = \{i, i+1, \ldots, i+n_i\}$ if $n_i = |\mathrm{snd}(i)| - 1$.
- For $i \in V^{\mathrm{c}}$, define the index set $\alpha_i = \mathrm{col}(i) \backslash \mathrm{snd}(i) = \gamma_i \backslash \nu_i$. The index set α_i is empty if i is the root of the maximal supernodal elimination tree.

Fig. 17 illustrates the above definitions of index sets. Fig. 18 shows a maximal supernode partition of a chordal sparsity pattern with a topological postordering. In this case,

$$\gamma_3 = \{4, 5, 8\}, \quad \nu_3 = \{4\}, \quad \alpha_3 = \{5, 8\}, \quad \text{and} \quad \eta_3 = \{6, 7, 9\}.$$

2.4 Chordal embedding and decomposition

2.4.1 Cholesky factorization

A fundamental result in linear algebra states that for any symmetric positive definite matrix X and every permutation P_σ with an order $\sigma = (\sigma(1), \ldots, \sigma(n))$, there exists a unique factorization, known as *Cholesky factorization*,

$$P_\sigma X P_\sigma^T = L D L^T, \tag{21}$$

where L is unit lower triangular, D positive diagonal. If X is positive semidefinite, then D is nonnegative diagonal, and the factorization is not unique for every permutation P_σ.

In addition, if the positive definite matrix X has a sparsity pattern E, we can find a relationship between the sparsity pattern of X and $L + L^T$. The paper [Ros70] proves that if $X \in \mathbf{S}_E^n$, then

$$P_\sigma^T (L + L^T) P_\sigma \in \mathbf{S}_{E'}^n$$

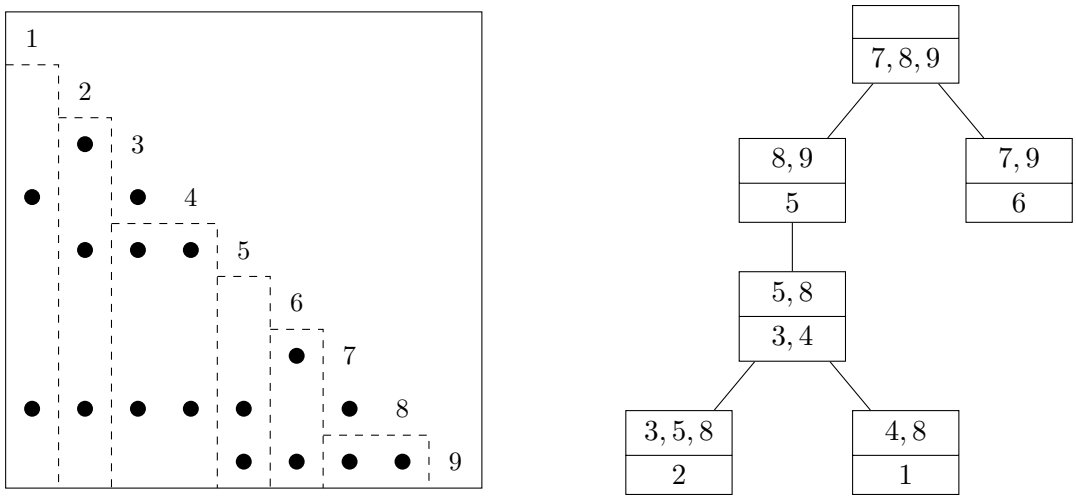


Figure 18: Fig. 11 with a topological postordering and a corresponding maximal supernodal elimination tree. The dashed lines separate the supernodes in the maximal supernodal elimination tree.

where $E' = E^*_\sigma$ is the edge set of the filled ordered sparsity graph $G_\sigma(V, E, \sigma)$.

Furthermore, Rose [Ros70] found the relationship between chordal graphs and matrix elimination. Namely, if E is a chordal pattern and σ is a corresponding perfect elimination ordering, then we have $E^*_\sigma = E$, which means we get a 'zero-fill' Cholesky factorization. And the original matrix X and its Cholesky factor $L + L^T$ share the same sparsity pattern, which is stated in the theorem below.

Theorem 5. A sparsity pattern E is chordal if and only if every positive definite matrix in $\mathbf{S}^n_E$ has a Cholesky factorization with

$$P^T(L + L^T)P \in \mathbf{S}^n_E. \tag{22}$$

2.4.2 Chordal embedding

Chordal embedding, or *triangulation*, describes the idea of transforming a non-chordal graph $G = (V, E)$ into a chordal graph $G' = (V, E')$ by adding edges, *i.e.*, $E \subseteq E'$. With all the properties of chordal sparsity mentioned above, it is not difficult to imagine why chordal embedding is useful. However, this approach could not be used in non-chordal graphs. Sometimes we also add edges to chordal graphs in order to achieve improvements in computational efficiency as well as algorithm stability. For example, Fukuda et al. [FKMN01] describes the trade-offs between number of cliques and clique size, and claims to achieve a balance by the conversion method.

In practice, there are several factors in chordal embedding that are interesting to researchers. For example, the number of fill-ins, *i.e.*, added edges, is a great concern in some problems. And actually the number of added edges depends heavily on the ordering of the graphs. In other words, if we change the order of a graph, we may arrive in different embedded graph, even with the same algorithm. Also, researches have been done to find minimum fill-ins. It is interesting because a large number of added edges increases the size of storage as well as the computation time. But

unfortunately, minimum fill-in is claimed to be an NP-complete problem [RTL76, Yan81], and is not always desirable due to algorithm complexity.

An alternative criterion is called *minimal* chordal embedding [Oht76, Ros74]. It is called minimal if the removal of any added edge results in a non-chordal graph. In practice, the minimal chordal embedding algorithm is a useful method to test chordality. If we plug in a chordal graph, the algorithm outputs a perfect elimination ordering with zero fill-in. For more discussion on minimal triangulation techniques, one may refer to the survey [Heg06].

2.4.3 Chordal decomposition of sparse matrix cones

Before we show the two important decomposition theorems of sparse matrix cones, we introduce the concept of *index set*. Index set indicates an ordered sequence of distinct integers from $\{1, \ldots, n\}$, denoted as $\beta = (\beta(1), \ldots, \beta(r))(r \leq n)$. Then the corresponding *index matrix* $P_\beta \in \mathbf{R}^{r \times n}$ is defined by

$$(P_\beta)_{ij} = \begin{cases} 1 & j = \beta(i) \\ 0 & \text{otherwise.} \end{cases}$$

A special case is the one with $r = n$, which is called a *permutation matrix*. An immediate result of the definition is

$$P_\beta x = x_\beta = (x_{\beta(1)}, \cdots, x_{\beta(r)}),$$

where x_β is called a permuted subvector of x. And then if we multiply an $n \times n$ matrix with P_β on the left and P_β^T on the right, we simply extract an $r \times r$ principal submatrix with rows and columns indexed by β, and apply a symmetric reordering to it, *i.e.*,

$$P_\beta X P_\beta^T = X_{\beta\beta} = \begin{bmatrix} X_{\beta(1)\beta(1)} & X_{\beta(1)\beta(2)} & \cdots & X_{\beta(1)\beta(r)} \\ X_{\beta(2)\beta(1)} & X_{\beta(2)\beta(2)} & \cdots & X_{\beta(2)\beta(r)} \\ \vdots & \vdots & & \vdots \\ X_{\beta(r)\beta(1)} & X_{\beta(r)\beta(2)} & \cdots & X_{\beta(r)\beta(r)} \end{bmatrix}.$$

Similarly, we have

$$(P_\beta^T y)_i = \begin{cases} y_i & i = \beta(j) \\ 0 & i \notin \{\beta(1), \ldots, \beta(r)\}. \end{cases}$$

And then if we multiply an $r \times r$ matrix Y with P_β^T on the left and P_β on the right, we obtain an $n \times n$ matrix X with $X_{\beta\beta} = Y$ and zeros in other entries, *i.e.*,

$$(P_\beta^T Y P_\beta)_{st} = \begin{cases} Y_{ij} & (s,t) = (\beta(i), \beta(j)) \\ 0 & s, t \notin \{\beta(1), \ldots, \beta(r)\}. \end{cases}$$

Now we are ready to introduce the decomposition theorems of sparse matrix cones.

Theorem 6. ([GT84, AHMR88, Kak10]) Let E be a chordal sparsity pattern of order n. Then $X \in \mathbf{S}^n_{E,+}$ if and only if it can be expressed as

$$X = \sum_{\text{cliques } \gamma_j} P_{\gamma_j}^T H_j P_{\gamma_j} \tag{23}$$

with positive semidefinite matrices H_j and index matrix P_{γ_j}.

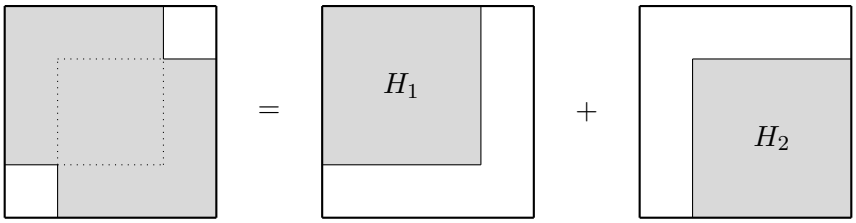


Figure 19: A chordal sparsity pattern with two overlapping dense principal submatrices.

Fig. 19 illustrates the decomposition theorem 6. Theorem 6 states that the positive semidefinite cone $\mathbf{S}^n_{E,+}$ can be written as the sum of closed convex cones, *i.e.*,

$$\mathbf{S}^n_{E,+} = \sum_{\text{cliques } \gamma_j} \mathcal{K}_j \quad \text{where} \quad \mathcal{K}_j = \{P_{\gamma_j}^T H_j P_{\gamma_j} \mid H_j \succeq 0\}.$$

The cones $\mathcal{K}_j$ have simple but dense structures. Also, it can be shown that each cone $\mathcal{K}_j$ is closed, convex and pointed. But since it has empty interior, it is not a proper cone. The only exception is the complete graph, in which there is only one clique.

If the sparsity pattern E is non-chordal, the decomposition (23) may still exist. But for every sparse matrix with pattern E, we can always find examples that cannot be decomposed in the form of (23). One possible method is introduced in the paper [GT84].

The next theorem establishes the relationship between a positive semidefinite completable matrix and its positive semidefinite principal submatrices.

Theorem 7. ([GJSW84]) Let E be a chordal sparsity pattern of order n. Then $X \in \mathbf{S}^n_{E,\mathrm{c}+}$ if and only if

$$X_{\gamma_j\gamma_j} \succeq 0 \tag{24}$$

for all cliques γ_j in the sparsity pattern E. Furthermore, $X \in \mathbf{int}\,\mathbf{proj}_E(\mathbf{S}^n_+) = \mathbf{proj}_E(\mathbf{S}^n_{++})$ if and only if $X_{\gamma_j\gamma_j} \succ 0$.

It is also worth noting that if E is non-chordal, the condition (24) is necessary but not sufficient. A simple counterexample is provided in the survey paper [VA14].

2.5 Clique decomposition of SDPs

With Theorem 6 and 7, we can decompose the positive semidefinite cone $\mathbf{S}^n_{E,+}$ and the positive semidefinite completable cone $\mathbf{S}^n_{E,\mathrm{c}+}$ into several small-size but dense cones. It is first introduced in [FKMN01, NFF$^+$03] as the *conversion methods* or *clique decomposition methods*. Suppose the sparsity pattern E has cliques $\gamma_1, \ldots, \gamma_l$. We introduce a separate variable $\tilde{X}_j$ for each principal

submatrix $X_{\gamma_j\gamma_j}$. Then the primal sparse SDP (11) can be written as

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{l} \mathbf{tr}(\tilde{C}_j\tilde{X}_j) \\ \text{subject to} & \displaystyle\sum_{j=1}^{l} \mathbf{tr}(\tilde{A}_{ij}\tilde{X}_j) = b_i, \quad i = 1,\ldots,m \\ & \tilde{X}_j = X_{\gamma_j\gamma_j}, \quad j = 1,\ldots,l \\ & \tilde{X}_j \succeq 0, \quad j = 1,\ldots,l, \end{array}$$

with primal variables $\tilde{X}_j = X_{\gamma_j\gamma_j} = P_{\gamma_j} X P_{\gamma_j}^T \in \mathbf{S}^{|\gamma_j|}$, $j = 1,\ldots,l$, and $X \in \mathbf{S}^n$. The coefficients $\tilde{C}_j$ and $\tilde{A}_{ij}$ are matrices that satisfy

$$\sum_{j=1}^{l} \mathbf{tr}(\tilde{C}_j X_{\gamma_j\gamma_j}) = \mathbf{tr}(CX)$$

and

$$\sum_{j=1}^{l} \mathbf{tr}(\tilde{A}_{ij}\tilde{X}_j) = \mathbf{tr}(A_i X), \quad i = 1,\ldots,m,$$

for all $X \in \mathbf{S}_E^n$. The second set of constraints states that the overlapping submatrices $X_{\gamma_j\gamma_j}$ must be consistent. If we denote ξ_k as the index set of intersection of cliques γ_k and its parent clique in the clique tree. Thus it can be rewritten as

$$P_{\xi_k}(P_{\gamma_j}^T \tilde{X}_j P_{\gamma_j} - P_{\gamma_k}^T \tilde{X}_k P_{\gamma_k}) P_{\xi_k}^T = 0, \quad k \in \mathrm{ch}(j),\ j = 1,\ldots,l.$$

Hence the problem becomes

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{l} \mathbf{tr}(\tilde{C}_j\tilde{X}_j) \\ \text{subject to} & \displaystyle\sum_{j=1}^{l} \mathbf{tr}(\tilde{A}_{ij}\tilde{X}_j) = b_i, \quad i = 1,\ldots,m \\ & P_{\xi_k}(P_{\gamma_j}^T \tilde{X}_j P_{\gamma_j} - P_{\gamma_k}^T \tilde{X}_k P_{\gamma_k}) P_{\xi_k}^T = 0, \quad k \in \mathrm{ch}(j),\ j = 1,\ldots,l \\ & \tilde{X}_j \succeq 0, \quad j = 1,\ldots,l. \end{array} \tag{25}$$

And the dual sparse SDP (12) can be written as

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to} & \displaystyle\sum_{i=1}^{m} y_i A_i + \sum_{j=1}^{l} P_{\gamma_j}^T H_j P_{\gamma_j} = C \\ & H_j \succeq 0, \quad j = 1,\ldots,l, \end{array} \tag{26}$$

with dual variables $y \in \mathbf{R}^m$ and $H_j \in \mathbf{S}^{|\gamma_j|}$. Note that P_{γ_j} is the index matrix.

As an example, suppose the chordal sparsity pattern E is the one in Fig. 18. The cliques are $\gamma_1 = (1,4,8)$, $\gamma_2 = (2,3,5,8)$, $\gamma_3 = (3,4,5,8)$, $\gamma_4 = (5,8,9)$, $\gamma_5 = (6,7,9)$, $\gamma_6 = (7,8,9)$. The principal submatrices are

$$X_{\gamma_1\gamma_1} = \begin{bmatrix} X_{11} & X_{14} & X_{18} \\ X_{41} & X_{44} & X_{48} \\ X_{81} & X_{84} & X_{88} \end{bmatrix} \qquad X_{\gamma_2\gamma_2} = \begin{bmatrix} X_{22} & X_{23} & X_{25} & X_{28} \\ X_{32} & X_{33} & X_{35} & X_{38} \\ X_{52} & X_{53} & X_{55} & X_{58} \\ X_{82} & X_{83} & X_{85} & X_{88} \end{bmatrix}$$

$$X_{\gamma_3\gamma_3} = \begin{bmatrix} X_{33} & X_{34} & X_{35} & X_{38} \\ X_{43} & X_{44} & X_{45} & X_{48} \\ X_{53} & X_{54} & X_{55} & X_{58} \\ X_{83} & X_{84} & X_{85} & X_{88} \end{bmatrix} \qquad X_{\gamma_4\gamma_4} = \begin{bmatrix} X_{55} & X_{58} & X_{59} \\ X_{85} & X_{88} & X_{89} \\ X_{95} & X_{98} & X_{99} \end{bmatrix}$$

$$X_{\gamma_5\gamma_5} = \begin{bmatrix} X_{66} & X_{67} & X_{69} \\ X_{76} & X_{77} & X_{79} \\ X_{96} & X_{97} & X_{99} \end{bmatrix} \qquad X_{\gamma_6\gamma_6} = \begin{bmatrix} X_{77} & X_{78} & X_{79} \\ X_{87} & X_{88} & X_{89} \\ X_{97} & X_{98} & X_{99} \end{bmatrix}.$$

Then separate matrices $\tilde{X}_1$, $\tilde{X}_2$, $\tilde{X}_3$, and $\tilde{X}_4$ are introduced and the consistency constraints are

$$\begin{bmatrix} \tilde{X}_{1,22} & \tilde{X}_{1,23} \\ \tilde{X}_{1,32} & \tilde{X}_{1,33} \end{bmatrix} = \begin{bmatrix} \tilde{X}_{3,22} & \tilde{X}_{3,24} \\ \tilde{X}_{3,42} & \tilde{X}_{3,44} \end{bmatrix}$$

$$\begin{bmatrix} \tilde{X}_{2,22} & \tilde{X}_{2,23} & \tilde{X}_{2,24} \\ \tilde{X}_{2,32} & \tilde{X}_{2,33} & \tilde{X}_{2,34} \\ \tilde{X}_{2,42} & \tilde{X}_{2,43} & \tilde{X}_{2,44} \end{bmatrix} = \begin{bmatrix} \tilde{X}_{3,11} & \tilde{X}_{3,13} & \tilde{X}_{3,14} \\ \tilde{X}_{3,31} & \tilde{X}_{3,33} & \tilde{X}_{3,34} \\ \tilde{X}_{3,41} & \tilde{X}_{3,43} & \tilde{X}_{3,44} \end{bmatrix}$$

$$\begin{bmatrix} \tilde{X}_{3,33} & \tilde{X}_{3,34} \\ \tilde{X}_{3,43} & \tilde{X}_{3,44} \end{bmatrix} = \begin{bmatrix} \tilde{X}_{4,11} & \tilde{X}_{4,12} \\ \tilde{X}_{4,21} & \tilde{X}_{4,22} \end{bmatrix}$$

$$\begin{bmatrix} \tilde{X}_{4,22} & \tilde{X}_{4,23} \\ \tilde{X}_{4,32} & \tilde{X}_{4,33} \end{bmatrix} = \begin{bmatrix} \tilde{X}_{6,22} & \tilde{X}_{6,23} \\ \tilde{X}_{6,32} & \tilde{X}_{6,33} \end{bmatrix}$$

$$\begin{bmatrix} \tilde{X}_{5,22} & \tilde{X}_{5,23} \\ \tilde{X}_{5,32} & \tilde{X}_{5,33} \end{bmatrix} = \begin{bmatrix} \tilde{X}_{6,11} & \tilde{X}_{6,13} \\ \tilde{X}_{6,31} & \tilde{X}_{6,33} \end{bmatrix}.$$

The main advantage of the conversion method is that it dramatically decreases the size of the matrix variables, which in turn reduces the computation cost. But as a trade-off, the conversion method introduces a large number of equality constraints in addition to the m equality constraints in the original problem. If the reformulated problem is solved by an interior-point method, the increased computing complexity in each iteration may offset the benefit of the increased sparsity. The above conversion methods can be implemented in MATLAB package `SparseCoLo` [FKK+09] and in Python library `Chompack` [DV09]. Also it is interesting to know that the conversion method can be used in combination with any semidefinite optimization algorithm.

In the packages mentioned above, interior-point methods are used to exploit the chordal sparsity of SDPs. One approach is to apply interior-point methods to the standard SDP (9) and exploit aggregate sparsity during implementation. It was first introduced in [FKMN01, NFF+03]. The paper [KHAR15] implements the primal-dual interior-point method to solve distributed SDPs with a tree structure. The other approach is to treat (11) as a conic LP with respect to the positive semidefinite completable cone $\mathbf{S}^n_{E,\mathrm{c}+}$. Then the interior-point method is implemented to the nonsymmetric conic

optimization. The survey paper [VA14] briefly summarizes the procedure, and several applications can be found in [Cha09, Nes12, SY15, KHA15].

Also several first-order algorithms and splitting algorithms [Sun15, ZG14, DZG12, LZT11] make use of decomposed SDPs (25 - 26) to solve the standard SDP (9). The paper [ZFP$^+$16] applies ADMM to the above results of conversion methods, which generates nice results for large sparse problems in SDPLIB [Bor99]. Z. Lu et al [LNM07] uses proximal method to solve a convex-concave saddle-point problem, which is equivalent to a decomposed well-structured large-scale semidefinite problem.

2.6 Summary

At the end of this section, we briefly summarize the assumptions and properties in chordal sparsity, and give the notations which will be used in the following sections.

- The graph $G_\sigma = (V, E, \sigma)$ is a connected chordal graph with $|V| = n$. The ordering $\sigma = (1, \ldots, n)$ is a perfect elimination ordering, *i.e.*, under such ordering, the higher neighborhood of every vertex is complete.

- An elimination tree of graph G_σ (Fig. 11) is a rooted tree with vertex set V and root $\sigma(n)$. The parent $p(v)$ of any vertex $v \in V$ in the elimination tree is the least ordered vertex in the higher neighborhood $\mathrm{adj}^+(v)$.

- Every clique in the elimination tree can be represented as $W = \mathrm{col}(w) = \mathrm{adj}^+(w) \cup \{w\}$ where the representative vertex w is the least ordered vertex in clique W. Thus we call the maximal supernode with representative vertex $w \in V^{\mathrm{c}}$ as

$$W = \mathrm{snd}(w) = \{w, p(w), \ldots, p^{|\mathrm{snd}(w)|-1}(w)\},$$

  and the partition of V formed accordingly is called maximal supernode partition, as shown in Fig. 14.

- The cliques in graph G_σ can be arranged in a clique tree or a maximal supernodal elimination tree T^{c}, as shown in Fig. 15. The vertex set is $\{\mathrm{snd}(v) \mid v \in V^{\mathrm{c}}\}$ and the root is $\mathrm{snd}(u)$ with $\sigma(n) \in \mathrm{snd}(u)$. Assume the ordering $\sigma = (1, \ldots, n)$ of the above supernodal elimination tree is a topological ordering. We can always construct a clique tree with the induced subtree property, *i.e.*, for every $v \in V$, the cliques that contain v form a subtree of T, as shown in Fig. 7. Assume T^{c} has the induced subtree property.

- Vertices in $\mathrm{snd}(v)$ are numbered consecutively, *i.e.*,

$$\mathrm{snd}(v) = \{i, i+1, \ldots, i+n_i\}$$

  if $\sigma(v) = i$ and $|\mathrm{snd}(v)| = n_i + 1$.

Fig. 18 shows an example of maximal supernodal elimination tree with topological ordering and consecutively numbered vertices in every supernode. The index sets in the above postordered maximal supernodal elimination tree are defined as

$$\gamma_i = \mathrm{col}(i), \quad \nu_i = \mathrm{snd}(i), \quad \alpha_i = \mathrm{col}(i) \backslash \mathrm{snd}(i) = \gamma_i \backslash \nu_i$$

SDP category	Primal/dual variables	Algorithms
Standard SDP (9-10)	$X \in \mathbf{S}^n_+$ $y \in \mathbf{R}^m, S \in \mathbf{S}^n_+$	Interior-point methods exploiting sparsity during implementation [FKMN01, NFF$^+$03]
Sparse SDP (11-12)	$X \in \mathbf{S}^n_{E,\mathrm{c}+}$ $y \in \mathbf{R}^m, S \in \mathbf{S}^n_{E,+}$	Interior-point methods applied to nonsymmetric conic program [KHA15]
Decomposed SDP (25-26)	$\tilde{X}_j \in \mathbf{S}^{\lvert\gamma_j\rvert}_+$, $j = 1, \ldots, l$ $y \in \mathbf{R}^m, H_j \in \mathbf{S}^{\lvert\gamma_j\rvert}_+$	First-order algorithms and splitting algorithms [Sun15]

Table 1: List of semidefinite programs with optimal solution and algorithms.

for any representative vertex $i \in V^{\mathrm{c}}$.

Also in this section, we discuss different ways to solve semidefinite programs with aggregate sparsity pattern. Table 1 summarizes the three primal-dual pairs of SDPs we are interested in, and gives notations for the primal and dual variables. Assume the coefficient matrices A_i and C share the aggregate sparsity pattern E. From an optimal solution $\tilde{X}^\star_j$, $j = 1, \ldots, l$ of (25), we can construct an optimal solution X° of (11) with

$$\tilde{X}^\star_j = P_{\gamma_j} X^\circ P^T_{\gamma_j}, \quad j = 1, \ldots, l.$$

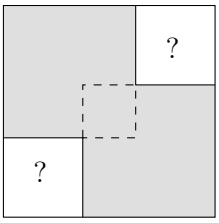
And any positive semidefinite completion $X^\bullet$ of X° is optimal for the standard SDP (9), *i.e.*,

$$X^\circ = \mathbf{proj}_E(X^\bullet).$$

Also Table 1 briefly summarizes the common algorithms applied to the corresponding SDPs.

3 Minimum rank positive semidefinite matrix completion with chordal sparsity pattern

In the last chapter, we saw that any positive semidefinite completion of an optimal solution of the sparse SDP (11) is optimal for the standard SDP (9). Although there are several matrix completion algorithms which can accomplish this procedure [GJSW84, GHJT99, VA14], we are usually interested in the positive semidefinite completion with the minimum rank. In general, it is difficult to even determine the minimum rank among all possible positive semidefinite completions, not to say to construct such a solution. However, if the positive semidefinite completable matrix has a chordal sparsity pattern, we can not only determine the minimum rank [Dan92], but we can also construct a matrix completion with minimum rank [Sun15]. To show this, we start with a matrix completion problem with a simple two-block chordal sparsity pattern. We introduce the algorithm to construct the minimum rank positive semidefinite matrix completion with chordal sparsity pattern, which is also a constructive proof of Dancis' theorem [Dan92]. This is followed by a detailed example, trying to explain the algorithm step by step. We then test the algorithm with randomly-generated sparse SDPs. At the end of the chapter, we discuss how this algorithm can be applied to the optimal solution of sparse SDP (11) or decomposed SDP (25). Application to the optimal power flow (OPF) problem is discussed in the next chapter.



$$A = \begin{bmatrix} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{bmatrix}$$

Figure 20: Two-block completion problem.

3.1 A two-block completion problem

We first consider a positive semidefinite matrix completion problem with a simple two-block chordal sparsity pattern as shown in Fig. 20. By Theorem 7, a positive semidefinite completion of matrix A exists if and only if

$$A_1 = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \succeq 0, \qquad A_2 = \begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix} \succeq 0.$$

We will show how we compute a PSD completion with rank

$$r = \max\{\mathbf{rank}(A_1), \mathbf{rank}(A_2)\}.$$

First, by singular value decomposition, we can find matrices U, V, $\tilde{V}$, W with column dimension r such that

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} U \\ V \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix}^T, \quad \begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} \tilde{V} \\ W \end{bmatrix} \begin{bmatrix} \tilde{V} \\ W \end{bmatrix}^T.$$

Then we have

$$VV^T = \tilde{V}\tilde{V}^T,$$

which in term means $V = \tilde{V}Q$ with Q orthogonal. To see this, suppose the matrices V and $\tilde{V}$ have singular value decomposition

$$V = P\Sigma Q_1^T, \quad \tilde{V} = P\Sigma Q_2^T.$$

Hence, we have $V = \tilde{V}Q$ with an orthogonal $r \times r$ matrix $Q = Q_2 Q_1^T$. Thus we can write A_1 in the following form

$$A_1 = \begin{bmatrix} U \\ V \end{bmatrix} Q^T Q \begin{bmatrix} U \\ V \end{bmatrix}^T = \begin{bmatrix} UQ^T \\ \tilde{V} \end{bmatrix} \begin{bmatrix} UQ^T \\ \tilde{V} \end{bmatrix}^T,$$

and we find a matrix Y with column dimension r, *i.e.*,

$$Y = \begin{bmatrix} UQ^T \\ \tilde{V} \\ W \end{bmatrix}$$

such that the matrix YY^T is a positive semidefinite completion of A with rank r. To see this, we have

$$YY^T = \begin{bmatrix} UQ^T \\ \tilde{V} \\ W \end{bmatrix} \begin{bmatrix} UQ^T \\ \tilde{V} \\ W \end{bmatrix}^T = \begin{bmatrix} A_{11} & A_{12} & UQ^T W^T \\ A_{21} & A_{22} & A_{23} \\ WQU^T & A_{32} & A_{33} \end{bmatrix}.$$

3.2 Minimum rank completion with chordal sparsity pattern

Now we are ready to introduce the minimum rank positive semidefinite matrix completion with chordal sparsity pattern. Dancis [Dan92] has shown that a matrix $A \in \mathbf{S}^n_{E,\mathrm{c}+}$ has a positive semidefinite completion with rank equal to

$$r^\bullet = \max_{i \in V^{\mathrm{c}}} \mathbf{rank}(A_{\gamma_i\gamma_i}). \tag{27}$$

As a constructive proof of Dancis' theorem, Algorithm 1 computes a matrix Y of size $n \times r^\bullet$ with $\mathbf{proj}_E(YY^T) = A$.

Algorithm 1 Minimum rank completion for chordal sparsity pattern.

Require: A matrix $A \in \mathbf{S}^n_{E,\mathrm{c}+}$, where $G = (V, E)$ is a chordal sparsity pattern with perfect elimination ordering $\sigma = (1, 2, \ldots, n)$, and a postordered maximal supernodal elimination tree for G_σ.

Ensure: A matrix $Y \in \mathbf{R}^{n\times m}$ with m equal to (27) and $\mathbf{proj}_E(YY^T) = A$.

- Compute $m = \max_{i\in V^{\mathrm{c}}} \mathbf{rank}(A_{\gamma_i\gamma_i})$.
- Enumerate the supernodes $j \in V^{\mathrm{c}}$ in inverse topological order.
  - For each supernode $j \in V^{\mathrm{c}}$, compute a factorization

$$A_{\gamma_j\gamma_j} = \begin{bmatrix} A_{\nu_j\nu_j} & A_{\nu_j\alpha_j} \\ A_{\alpha_j\nu_j} & A_{\alpha_j\alpha_j} \end{bmatrix} = \begin{bmatrix} U_j \\ V_j \end{bmatrix} \begin{bmatrix} U_j \\ V_j \end{bmatrix}^T, \tag{28}$$

    with U_j of size $|\nu_j| \times m$ and V_j of size $|\alpha_j| \times m$.
  - If j is the root of the supernodal elimination tree, let $Y_{\nu_j} = U_j$.
  - Otherwise compute an $m \times m$ orthogonal matrix Q such that $Y_{\alpha_j} = V_jQ^T$, and set $Y_{\nu_j} = U_jQ^T$.

To show correctness of the algorithm, we verify that YY^T is a completion, *i.e.*,

$$Y_{\gamma_j}Y_{\gamma_j}^T = A_{\gamma_j\gamma_j} \tag{29}$$

for all $j \in V^{\mathrm{c}}$.

At each step in the recursion over the tree, we add a new block Y_{ν_j}. Note that the blocks Y_{ν_i} for the supernodes that precede ν_i in the inverse topological ordering are left unchanged. It is therefore sufficient to verify that after supernode ν_j has been processed, the identity (29) holds, *i.e.*,

$$\begin{bmatrix} Y_{\nu_j} \\ Y_{\alpha_j} \end{bmatrix} \begin{bmatrix} Y_{\nu_j} \\ Y_{\alpha_j} \end{bmatrix}^T = \begin{bmatrix} A_{\nu_j\nu_j} & A_{\nu_j\alpha_j} \\ A_{\alpha_j\nu_j} & A_{\alpha_j\alpha_j} \end{bmatrix}.$$

If j is the root supernode, then

$$Y_{\gamma_j}Y_{\gamma_j}^T = U_jU_j^T = A_{\nu_j\nu_j} = A_{\gamma_j\gamma_j}.$$

Suppose (29) holds for all supernodes ν_i that are ancestors of supernodes j in the supernodal elimination tree. Then

$$A_{\alpha_j\alpha_j} = Y_{\alpha_j} Y_{\alpha_j}^T = V_j V_j^T \tag{30}$$

because α_j is a subset of the parent supernode. This means that there exists an orthogonal matrix Q such that $V_j = Y_{\alpha_j} Q$. By choosing $Y_{\nu_j} = U_j Q^T$, we obtain

$$\begin{aligned}
\begin{bmatrix} Y_{\nu_j} \\ Y_{\alpha_j} \end{bmatrix} \begin{bmatrix} Y_{\nu_j} \\ Y_{\alpha_j} \end{bmatrix}^T &= \begin{bmatrix} U_j U_j^T & U_j Q^T Y_{\alpha_j}^T \\ Y_\alpha Q U_j^T & Y_{\alpha_j} Y_{\alpha_j}^T \end{bmatrix} \\
&= \begin{bmatrix} U_j U_j^T & U_j V_j^T \\ V_j U_j^T & Y_{\alpha_j} Y_{\alpha_j}^T \end{bmatrix} \\
&= \begin{bmatrix} A_{\nu_j\nu_j} & A_{\nu_j\alpha_j} \\ A_{\alpha_j\nu_j} & A_{\alpha_j\alpha_j} \end{bmatrix}.
\end{aligned}$$

In general, the orthogonal matrix Q needed in the recursion can be computed in a similar way as shown in Section 3.1. We first compute a singular value decomposition (SVD)

$$V_j = P \Sigma Q_1^T,$$

with P and Q_1 orthogonal, and Σ rectangular. It follows from (30) that Y_{α_j} has the same rank, singular values, and left singular vectors as V_j. Therefore, it has an SVD

$$Y_{\alpha_j} = P \Sigma Q_2^T$$

with Q_2 orthogonal, and $P^T Y_{\alpha_j}$ can be written as

$$P^T Y_{\alpha_j} = \Sigma Q_2^T = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Q_{21}^T \\ Q_{22}^T \end{bmatrix},$$

where Σ_1 is the positive diagonal subblock of Σ. This determines

$$Q_{21}^T = \begin{bmatrix} \Sigma_1^{-1} & 0 \end{bmatrix} P^T Y_{\alpha_j}.$$

The block Q_{22}^T is any matrix with orthonormal rows that makes Q_2 orthogonal. The matrix $Q = Q_2 Q_1^T$ satisfies

$$Y_{\alpha_j} Q = P \Sigma Q_2^T Q_2 Q_1^T = P \Sigma Q_1^T = V_j.$$

Example. In the remaining part of the subsection, we show a concrete example to illustrate the algorithm. We use the chordal sparsity pattern E and the supernodal elimination tree as shown in Fig. 18. Suppose the matrix A to be completed is

$$A = \begin{bmatrix}
923 & ? & ? & 464 & ? & ? & ? & 1215 & ? \\
? & 1721 & 1494 & ? & 1182 & ? & ? & 1638 & ? \\
? & 1494 & 841 & 726 & 1016 & ? & ? & 1432 & ? \\
464 & ? & 1182 & 418 & 580 & ? & ? & 770 & ? \\
? & 1182 & 1061 & 580 & 921 & ? & ? & 1209 & 959 \\
? & ? & ? & ? & ? & 677 & 421 & ? & 746 \\
? & ? & ? & ? & ? & 421 & 270 & 661 & 499 \\
1215 & 1638 & 1432 & 770 & 1209 & ? & 661 & 1773 & 1195 \\
? & ? & ? & ? & 959 & 746 & 499 & 1195 & 1021
\end{bmatrix},$$

where the question marks denote the values that needed to be determined.

In this case, The set of representative vertices is $V^{\mathrm{c}} = \{1, 2, 3, 5, 6, 7\}$. The resulting rank is

$$m = \max_{i \in V^{\mathrm{c}}} \mathbf{rank}(A_{\gamma_i \gamma_i}) = 3.$$

The root of supernodal elimination tree is $\nu_7 = \{7, 8, 9\}$. By the Cholesky factorization of $A_{\gamma_7 \gamma_7}$, we have

$$Y_{\{7,8,9\}} = \begin{bmatrix} 16.4316 & 0 & 0 \\ 40.2272 & 12.4408 & 0 \\ 30.3682 & -2.1402 & 9.7053 \end{bmatrix},$$

where $Y_{\{7,8,9\}}$ is the 7th, 8th, and 9th rows of matrix Y. Then for $j = 6$, we have $\nu_6 = \{6\}$ and $\alpha_6 = \{7, 9\}$. Thus consider the Cholesky factorization

$$A_{\gamma_6 \gamma_6} = \begin{bmatrix} U_6 \\ V_6 \end{bmatrix} \begin{bmatrix} U_6 \\ V_6 \end{bmatrix}^T,$$

with $U_6 \in \mathbf{R}^{1 \times 3}$, $V_6 \in \mathbf{R}^{2 \times 3}$, and SVD $V_6 = P \Sigma Q_{61}^T$ and $Y_{\{7,9\}} = P \Sigma Q_{62}^T$. Hence, let

$$Q_6 = Q_{62} Q_{61}^T = \begin{bmatrix} 0.9847 & 0.1742 & 0 \\ 0.1462 & -0.8263 & 0.5438 \\ -0.0948 & 0.5355 & 0.8392 \end{bmatrix},$$

and

$$Y_{\{6\}} = U_6 Q_6^T = \begin{bmatrix} 25.6212 & 3.8044 & -2.4655 \end{bmatrix}.$$

Next for $j = 5$, we have $\nu_5 = \{5\}$ and $\alpha_5 = \{8, 9\}$. Similarly, the Cholesky factorization for $A_{\gamma_5 \gamma_5}$ is given by

$$A_{\gamma_5 \gamma_5} = \begin{bmatrix} U_5 \\ V_5 \end{bmatrix} \begin{bmatrix} U_5 \\ V_5 \end{bmatrix}^T,$$

with $U_5 \in \mathbf{R}^{1 \times 3}$, $V_5 \in \mathbf{R}^{2 \times 3}$, and SVD $V_5 = P \Sigma Q_{51}^T$ and $Y_{\{8,9\}} = P \Sigma Q_{52}^T$. Hence, let

$$Q_5 = Q_{52} Q_{51}^T = \begin{bmatrix} 0.9786 & 0.0911 & -0.1846 \\ 0.0380 & 0.8015 & 0.5968 \\ 0.2024 & -0.5910 & 0.7809 \end{bmatrix},$$

and

$$Y_{\{5\}} = U_5 Q_5^T = \begin{bmatrix} 29.6981 & 1.1517 & 6.1396 \end{bmatrix}.$$

Similarly, we have the Cholesky factorization

$$A_{\gamma_3 \gamma_3} = \begin{bmatrix} U_3 \\ V_3 \end{bmatrix} \begin{bmatrix} U_3 \\ V_3 \end{bmatrix}^T,$$

with $U_3 \in \mathbf{R}^{2 \times 3}$, $V_3 \in \mathbf{R}^{2 \times 3}$, and SVD $V_3 = P \Sigma Q_{31}^T$ and $Y_{\{5,8\}} = P \Sigma Q_{32}^T$. Hence, let

$$Q_3 = Q_{32} Q_{31}^T = \begin{bmatrix} 0.9786 & 0.0911 & -0.1846 \\ 0.0380 & 0.8015 & 0.5968 \\ 0.20237 & -0.5910 & 0.7809 \end{bmatrix},$$

and

$$Y_{\{3,4\}} = U_3 Q_3^T = \begin{bmatrix} 35.5002 & 0.3156 & -6.2957 \\ 20.0955 & -3.0853 & -2.1571 \end{bmatrix}.$$

Next, we factorize $A_{\gamma_2\gamma_2}$ as

$$A_{\gamma_2\gamma_2} = \begin{bmatrix} U_2 \\ V_2 \end{bmatrix} \begin{bmatrix} U_2 \\ V_2 \end{bmatrix}^T,$$

with $U_3 \in \mathbf{R}^{1\times 3}$, $V_3 \in \mathbf{R}^{3\times 3}$, and SVD $V_2 = P\Sigma Q_{21}^T$ and $Y_{\{3,5,8\}} = P\Sigma Q_{22}^T$. Hence, let

$$Q_2 = Q_{22} Q_{21}^T = \begin{bmatrix} 0.9896 & -0.0795 & 0.1197 \\ -0.0262 & 0.7192 & 0.6943 \\ -0.1413 & -0.6902 & 0.7096 \end{bmatrix},$$

and

$$Y_{\{2\}} = U_2 Q_2^T = \begin{bmatrix} 41.0544 & -1.0853 & -5.8617 \end{bmatrix}.$$

Finally, we have

$$A_{\gamma_1\gamma_1} = \begin{bmatrix} U_1 \\ V_1 \end{bmatrix} \begin{bmatrix} U_1 \\ V_1 \end{bmatrix}^T,$$

with $U_3 \in \mathbf{R}^{1\times 3}$, $V_3 \in \mathbf{R}^{2\times 3}$, and SVD $V_2 = P\Sigma Q_{11}^T$ and $Y_{\{4,8\}} = P\Sigma Q_{12}^T$. Hence, let

$$Q_1 = Q_{12} Q_{11}^T = \begin{bmatrix} 0.8267 & 0.5495 & 0.1207 \\ 0.5414 & -0.8354 & 0.0950 \\ -0.1530 & 0.0132 & 0.9881 \end{bmatrix},$$

and

$$Y_{\{1\}} = U_1 Q_1^T = \begin{bmatrix} 25.1164 & 16.4490 & -4.6474 \end{bmatrix}.$$

Therefore, the resulting Y is

$$Y = \begin{bmatrix} 25.1164 & 16.4490 & -4.6474 \\ 41.0544 & -1.0853 & -5.8617 \\ 35.5002 & 0.3156 & -6.2957 \\ 20.0955 & -3.0853 & -2.1571 \\ 29.6981 & 1.1517 & 6.13962 \\ 25.6212 & 3.8044 & -2.4655 \\ 16.4316 & 0 & 0 \\ 40.2272 & 12.4408 & 0 \\ 30.3682 & -2.1402 & 9.7053 \end{bmatrix},$$

with $\mathbf{rank}(Y) = 3$, and the completed matrix is given by

$$YY^T = \begin{bmatrix} 923 & 1040 & 926 & 464 & 736 & 718 & 413 & 1215 & 682 \\ 1040 & 1721 & 1494 & 841 & 1182 & 1062 & 675 & 1638 & 1192 \\ 926 & 1494 & 841 & 726 & 1016 & 926 & 583 & 1432 & 1016 \\ 464 & 841 & 1182 & 418 & 580 & 508 & 330 & 770 & 596 \\ 736 & 1182 & 1061 & 580 & 921 & 750 & 488 & 1209 & 959 \\ 718 & 1062 & 926 & 508 & 750 & 677 & 421 & 1078 & 746 \\ 413 & 675 & 583 & 330 & 488 & 421 & 270 & 661 & 499 \\ 1215 & 1638 & 1432 & 770 & 1209 & 1078 & 661 & 1773 & 1195 \\ 682 & 1192 & 1016 & 596 & 959 & 746 & 499 & 1195 & 1021 \end{bmatrix}$$

with $YY^T \succeq 0$ and $\mathbf{proj}_E(YY^T) = A$.

3.3 Test cases with randomly-generated sparse SDP

In this subsection we would like to test our algorithm with a randomly-generalized sparse SDP. Consider the following semidefinite program

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & X \succeq 0 \end{array}$$

and its dual problem

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to} & \displaystyle\sum_{i=1}^{m} y_i A_i + S = C, \quad i = 1, \ldots, m \\ & S \succeq 0. \end{array}$$

The procedure of generating such an SDP with a given chordal sparsity pattern E is as follows.

1. Generate a random strictly positive definite matrix $\hat{X} \in \mathbf{S}^n_{++}$ as a feasible point of the SDP.
2. Generate a number of m random matrices A_i, $i = 1, \ldots, m$ with sparsity pattern E.
3. Compute $b_i = \mathbf{tr}(A_i \hat{X})$.
4. Generate a number of m random scalars y_i, $i = 1, \ldots, m$.
5. Generate a random positive definite matrix $\hat{S} \in \mathbf{S}^n_{++}$ with sparsity pattern E.
6. Compute $C = \sum_{i=1}^m y_i A_i + \hat{S}$.

One possible method to generate a random positive definite matrix is to generate a random matrix $\hat{L}$ and compute $\hat{X} = \hat{L}\hat{L}^T$. Alternatively, we can generate a random symmetric $\tilde{X}$ and compute its eigenvalue decomposition $\tilde{X} = Q\tilde{\Lambda}Q^T$. It is possible that some diagonal entries in $\tilde{\Lambda}$ may be negative. Then we replace these negative eigenvalues with some random small positive values. The resulting new diagonal matrix $\hat{\Lambda}$ is used to compute the positive definite matrix $\hat{X} = Q\hat{\Lambda}Q^T$.

We have tested a total number of 250 examples with different chordal sparsity patterns E, *i.e.*, band pattern and block-arrow pattern, different matrix size $n = 100, 150, 200, 300, 500$, number of constraints $m = 20, 30, 50, 100, 150$, and bandwidth $w = 2, 3, 5, 7, 10$. The rank of the optimal solutions given by CVX [GB14] in MATLAB is always n, which is consistent to the claim that standard interior-point algorithms for solving SDPs will always return a solution whose rank is maximal among all optimal solutions. And the minimum rank completion algorithm makes a posterior operation on the above optimal solutions and gives another optimal solution of rank w in both cases.

3.4 Minimum rank completion for sparse/decomposed SDPs

In the last chapter, we discuss three equivalent primal-dual pairs of semidefinite programs, *i.e.*, standard SDP pair (9-10), sparse SDP pair (11-12), and decomposed SDP pair (25-26). It is easy to construct an optimal solution for the sparse SDP (11) from the optimal solution $X_j^\star$, $j = 1, \ldots, l$, for the decomposed SDP (25), *i.e.*, the matrix $X^\circ \in \mathbf{S}^n_{E,\mathrm{c}+}$ is optimal for the sparse SDP (11) with $X_j^\star = P_{\gamma_j} X^\circ P_{\gamma_j}^T$. We also know that any positive semidefinite completion of X° is an optimal solution for the standard SDP (9), *i.e.*, $X^\bullet \in \mathbf{S}^n_+$ is optimal for (9) if

$$X^\circ = \mathbf{proj}_E(X^\bullet).$$

There are many interesting matrix completion algorithms that can accomplish this procedure. For example, maximum determinant completion discussed in [GJSW84, GHJT99] returns a positive semidefinite matrix with the maximum determinant among all the positive semidefinite completions. But maximum determinant completion does not exist when the completable matrix X° is on the boundary of $\mathbf{S}^n_{E,\mathrm{c}+}$ [VA14]. The paper [VA14] proposes an algorithm to compute a dense positive semidefinite completion for any sparse matrix with chordal pattern. Among this, a low-rank positive semidefinite completion would be useful in many applications. In the ad hoc wireless sensor network localization problem [BY04, BLT$^+$06], a rank 2 optimal solution gives the exact location of sensors. Low-rank solutions are also interesting in unsupervised learning of image manifolds [WPS05, WS06] because low-rank solutions help detect low dimensional structure in high dimensional data sets of images. Moreover, a low-rank solution helps prove the exactness of semidefinite relaxation of optimal power flow (OPF) problem [GLTL12, LL12, Low14a, Low14b].

For a semidefinite program with aggregate chordal pattern E, a low-rank solution $X^\bullet$ for the standard SDP (9) can be constructed by Algorithm 1 with

$$X^\circ = \mathbf{proj}_E(X^\bullet),$$

and rank

$$r^\bullet = \mathbf{rank}(X^\bullet) = \max_{\text{cliques } \gamma_i} \mathbf{rank}(X^\circ_{\gamma_i\gamma_i}),$$

where X° can either be solved by interior-point methods for the sparse SDP (11) or constructed from the optimal solution for the decomposed SDP (25). This posterior operation can help us find optimal solutions with valid physical meanings. For example, in the sensor localization problem, physical locations of sensors can be determined from a rank 2 solution. In the next chapter, we apply this posterior rounding idea to the OPF problem, and examine in what situations we can determine physically meaningful voltages and powers from a low-rank completion $X^\bullet$.

4 Rounding of semidefinite relaxation of optimal power flow problem

4.1 Introduction

The AC optimal power flow (OPF) problem is to find a cost-optimal operating point of a power system that consists of a set of power busses interconnected with a network of transmission lines, where each power bus (or substation) has one or more generators and/or a load. Typical objective functions are generation cost or transmission loss, which is some function of current, voltage, and

power, and the constraints consist of a set of nonlinear power flow constraints and bounds on voltage magnitudes, transmission line flows, and power generation. Many different problem formulations exist [HG91, MAEH99, MEHA99], but OPF problems are generally difficult nonconvex optimization problems *i.e.*, a nonconvex quadratic program with quadratic constraints (QCQP).

Since Carpentier [MEHA62] first introduced the OPF problem in 1962, numerous solution techniques have been proposed to solve this complicated nonconvex optimization problem, including sequential quadratic programming, interior-point methods, genetic algorithms and evolutionary programming [QDB09, ZMST11]. Recently, researchers have applied semidefinite relaxation (SDR) techniques to various OPF problem formulations [BWFW08, LL10, LL12]. The resulting relaxation problem is a conic optimization problem, *i.e.*, a convex problem, that can be solved in polynomial time with an interior-point method. The topic of semidefinite relaxation has attracted close attention because, unlike previous nonconvex formulations, the solution provides information on the feasibility of the original problem. We say that the SDR is "exact" if it provides the global optimal solution to the original nonconvex problem. Otherwise, the solution for the SDR formulation is a lower bound on the optimal value. Conditions that guarantee exactness of the SDR have been studied extensively in [BGLC11, LL10, LTZ14].

However, the computational cost of the above SDR formulation grows rapidly with the size of the power networks, *i.e.*, the size of the OPF problem. Hence, to solve the SDR of a large-scale OPF problem is considered to be impractical. The principal bottlenecks are a dense symmetric matrix primal variable, which corresponds to the number of busses in the power network, as well as a large-scale positive definite system of equations that defines the update direction in each iteration in the interior-point method. On the other hand, the SDR formulation has the property of sparsity as well as low-rank structure. The property of sparsity reflects the structure of power network, *i.e.*, the fact that each power bus is connected to a small portion of adjacent power busses. And surprisingly the dual variables inherits the sparsity pattern [LL12]. With this idea, Lavaei and Low propose to solve the dual problem of SDR by SeDuMi [Stu98], a general-purpose primal-dual conic solver. Although the paper [LL12] exploits the sparsity of the dual problem, the dense pattern in the primal variable still remains. Also it is interesting to know that even if we avoid the dense primal variable by using some sparse SDP solvers such as SMCP [ADV10] and DSDP [BY04] to solve the dual problem, the bottleneck in the update in each iteration still remains as a big problem in the large-scale OPF problem.

Moreover, thanks to the advances in sparse semidefinite programming, Jabr [Jab12] applied the conversion methods [FKMN01] to the SDR formulation of the OPF problem. In this context, the conversion method reformulates a sparse SDP as an equivalent decomposed SDP with additional equality constraints imposing consistency, as discussed in Section 2. In general, the primal variables in the decomposed SDP correspond to the cliques of the chordal embedding of the power network graph structure. Hence, the benefit of reformulating the sparse SDP as a decomposed SDP depends very much on the underlying network graph structure and its chordal embedding; see *e.g.*, [LZT11].

In the OPF problem, sometimes we are not only interested in the global optimal value, it is also practically meaningful to know the optimal solution for the SDR problem, as it indicates a practical allocation of voltages and powers. In general, it is difficult to find a low-rank positive semidefinite completion for the optimal solution of the decomposed SDP solved by the conversion method. In other words, it may be impractical to construct a practically meaningful solution for the original OPF problem. But if the conversion method is applied to a power network with chordal sparsity pattern or its chordal embedding, Algorithm 1 can be applied to the optimal solution to construct

a low-rank positive semidefinite completion.

In this section, we first introduce the power system model and our formulation of the OPF problem as well as its SDR. This is followed by numerical experiments.

4.2 Problem formulation and semidefinite relaxation

We start this section by describing the power system model and our formulation of OPF problem. We then propose a reformulation of the problem and derive the corresponding SDR.

4.2.1 Power flow model

The power system model consists of a network of power busses (nodes). We denote the set of power busses as $\mathcal{N}$ with $|\mathcal{N}| = n$, and we denote complex voltage at node i as v_i and v for the column vector. The voltage magnitude must satisfy the constraints

$$v_i^{\min} \leq |v_i| \leq v_i^{\max}, \quad i \in \mathcal{N}. \tag{31}$$

Note that this constraint is nonconvex because of the lower bound.

The set of transmission lines (edges) is a subset of $\mathcal{L} \subseteq \mathcal{N} \times \mathcal{N}$, *i.e.*, $(i,j) \in \mathcal{L}$ if node i and node j are connected with transmission line. Transmission lines may not be symmetric and we model the network graph as a directed graph, *i.e.*, (i,j) is not the same as (j,i). We denote $i \sim j$ if $(i,j) \in \mathcal{L}$ or $(j,i) \in \mathcal{L}$. The real and reactive power flows to node i from node j are denoted as p_{ij} and q_{ij}. Note that it is not the same as the flows from node i to node j, which are written as p_{ji} and q_{ji}.

The complex power flow magnitude must remain below the transmission line's capacity, *i.e.*,

$$p_{ij}^2 + q_{ij}^2 \leq S_{ij}^2, \tag{32}$$

where S_{ij} is a real positive constant.

The real and reactive power into or out of node i are the sums of the flows through the transmission lines connected to node i:

$$\sum_{j \in \mathcal{N}} p_{ij} = p_i \qquad \text{and} \qquad \sum_{j \in \mathcal{N}} q_{ij} = q_i. \tag{33}$$

They are subject to the constraints

$$p_i^{\min} \leq p_i \leq p_i^{\max} \qquad \text{and} \qquad q_i^{\min} \leq q_i \leq q_i^{\max}. \tag{34}$$

For each transmission line $(i,j) \in \mathcal{L}$, let $y_{ij} = g_{ij} - \hat{\imath} b_{ij}$ denote its admittance and $z_{ij} = r_{ij} + \hat{\imath} x_{ij}$ denote its impedance, then we have $y_{ij} z_{ij} = 1$. Hence, we can derive the relationship between power flows and voltages:

$$p_{ij} + \hat{\imath} q_{ij} = v_i (v_i^H - v_j^H) y_{ij}^H \quad \text{for } (i,j) \in \mathcal{L}, \tag{35}$$

where H denotes the complex conjugate. Note that this quadratic constraint is the main source of nonconvexity of the OPF problem. The above constraints (31 – 35) form the feasible region of the nonconvex quadratically constrained power flow problem, denoted as $\mathcal{F}_1$.

The objective function is the power loss. The power loss on line $(i,j) \in \mathcal{L}$ is $g_{ij}|v_i - v_j|^2$ where g_{ij} is the conductance of transmission line. Hence, the power loss in the network is

$$\sum_{(i,j) \in \mathcal{L}} g_{ij} |v_i - v_j|^2.$$

Combining the constraints and objective function, the OPF problem can be written as

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{(i,j)\in\mathcal{L}} g_{ij}|v_i - v_j|^2 & \\ \text{subject to} & p_{ij} + \hat{\imath} q_{ij} = v_i(v_i^H - v_j^H)y_{ij}^H, & (i,j)\in\mathcal{L} \\ & p_{ij}^2 + q_{ij}^2 \le S_{ij}^2, & (i,j)\in\mathcal{L} \\ & \displaystyle\sum_{j\in\mathcal{N}} p_{ij} = p_i, & i\in\mathcal{N} \\ & \displaystyle\sum_{j\in\mathcal{N}} q_{ij} = q_i, & i\in\mathcal{N} \\ & p_i^{\min} \le p_i \le p_i^{\max}, & i\in\mathcal{N} \\ & q_i^{\min} \le q_i \le q_i^{\max}, & i\in\mathcal{N} \\ & v_i^{\min} \le |v_i| \le v_i^{\max}, & i\in\mathcal{N}, \end{array} \tag{36}$$

where the variables are real and reactive power flows p_{ij}, q_{ij} over transmission line (i,j), power flows p_i and q_i into or out of node i, and the voltage v_i at node i. This is a nonconvex quadratic programming with quadratic constraints.

4.2.2 Semidefinite relaxation

The challenge in solving the OPF problem (36) comes from the nonconvex quadratic equality constraint as well as the inequality constraints $|v_i| \ge v_i^{\min}$. To overcome this challenge, we enlarge the feasible set of the OPF problem to a convex set. To see this, define $X = vv^H$, *i.e.*,

$$X_{ij} = v_i v_j^H, \quad i \sim j \text{ or } i = j.$$

Hence, the OPF problem (36) can be rewritten as

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{(i,j)\in\mathcal{L}} g_{ij}|v_i - v_j|^2 & \\ \text{subject to} & p_{ij} + \hat{\imath} q_{ij} = (X_{ii} - X_{ij})y_{ij}^H, & (i,j)\in\mathcal{L} \\ & p_{ij}^2 + q_{ij}^2 \le S_{ij}^2, & (i,j)\in\mathcal{L} \\ & \displaystyle\sum_{j\in\mathcal{N}} p_{ij} = p_i, & i\in\mathcal{N} \\ & \displaystyle\sum_{j\in\mathcal{N}} q_{ij} = q_i, & i\in\mathcal{N} \\ & p_i^{\min} \le p_i \le p_i^{\max}, & i\in\mathcal{N} \\ & q_i^{\min} \le q_i \le q_i^{\max}, & i\in\mathcal{N} \\ & (v_i^{\min})^2 \le X_{ii} \le (v_i^{\max})^2, & i\in\mathcal{N} \\ & \textbf{rank } X = 1. & \end{array} \tag{37}$$

The only nonconvexity results from the rank constraint **rank** $X = 1$. We apply the SDR by simply replacing it with a positive semidefinite cone constraint, yielding a convex SDR problem with a

convex feasible region $\mathcal{F}_2$, *i.e.*,

$$\begin{array}{lll}
\text{minimize} & g_{ij}(X_{ii}+X_{jj}-2X_{ij}) & \\
\text{subject to} & p_{ij}+\hat{\imath}q_{ij}=(X_{ii}-X_{ij})y_{ij}^H, & (i,j)\in\mathcal{L} \\
& p_{ij}^2+q_{ij}^2\le S_{ij}^2, & (i,j)\in\mathcal{L} \\
& \displaystyle\sum_{j\in\mathcal{N}} p_{ij}=p_i, & i\in\mathcal{N} \\
& \displaystyle\sum_{j\in\mathcal{N}} q_{ij}=q_i, & i\in\mathcal{N} \\
& p_i^{\min}\le p_i\le p_i^{\max}, & i\in\mathcal{N} \\
& q_i^{\min}\le q_i\le q_i^{\max}, & i\in\mathcal{N} \\
& (v_i^{\min})^2\le X_{ii}\le (v_i^{\max})^2, & i\in\mathcal{N} \\
& X\succeq 0. &
\end{array} \tag{38}$$

It is easy to see that $\mathcal{F}_1\subseteq\mathcal{F}_2$, and hence the solution $X^\star$ provides a lower bound on the optimal value of (36). Furthermore, if $\mathbf{rank}(X^\star)=1$, the SDR is said to be "exact", *i.e.*, we can obtain a global optimal solution for (36) by computing a rank-1 factorization $X^\star=\tilde{v}\tilde{v}^H$. Conditions for exactness are studied extensively around the world [Low14b, LL12, GLTL12]. The following theorem [FCLC11] provides a sufficient condition for the exactness of SDR (38), and the results in [ZT13, BGCL15] slightly improve the condition.

Theorem 8. Suppose the objective function is just a function of real power, $f(p)$, and is convex and strictly increasing. If the network is radial (with tree topology) and $p_i^{\min}=q_i^{\min}=-S_{ij}=-\infty$ for all $i\in\mathcal{N}$, the SDP relaxation (38) is exact.

If we follow the convention that p_i is negative for loads, the assumptions $p_i^{\min}=q_i^{\min}=-\infty$ can be interpreted in tree networks as allowing loads to be "over-satisfied" [Low14a, Low14b].

If $\mathbf{rank}(X^\star)>1$, then the solution for SDR (38) provides a lower bound for the original OPF problem (36). In this case, the optimal relaxed solution is not in the feasible region $\mathcal{F}_1$, but can still provide an approximation of optimal powers and voltage magnitudes. A convex heuristic is to find an approximation by replacing all the eigenvalues of the optimal solution for (38) with zero except the largest one λ_1, *i.e.*,

$$\hat{X}=\lambda_1 u_1 u_1^H,$$

where u_1 is the principal eigenvector associated with λ_1. If the rank of the optimal relaxed solution is not so much larger than one and the largest eigenvalue is substantially larger than the others, then the convex heuristic gives a solution close to the ideal outcome, and the principal eigenvector serves as a good approximation to the complex voltages [Tay15].

4.2.3 The conversion method and the posterior rounding operation

The conversion method discussed in section 2 can be applied to the SDR problem (38). Since the conversion only affects the positive semidefinite matrix variable, we will simplify our notation by

only considering the SDP of the form

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\ & X \in \mathbf{H}^n_{E,\mathrm{c}+} \end{array} \tag{39}$$

where $A_i \in \mathbf{H}^n_E$, $i = 1, \ldots, m$. The sparsity pattern E is associated with the underlying network graph, and is assumed to be connected chordal with cliques $\gamma_1, \ldots, \gamma_l$. We define $\mathbf{H}^n_E$ as the set of Hermitian matrices of order n and with sparsity pattern E, and $\mathbf{H}^n_{E,\mathrm{c}+}$ as the set of positive semidefinite completable matrices in $\mathbf{H}^n_E$. Hence we can rewrite (39) in a similar form as (25), *i.e.*,

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{j=1}^{l} \mathbf{tr}(\tilde{C}_j \tilde{X}_j) \\ \text{subject to} & \displaystyle\sum_{j=1}^{l} \mathbf{tr}(\tilde{A}_{ij} \tilde{X}_j) = b_i, \quad i = 1, \ldots, m \\ & P_{\xi_k}(P^T_{\gamma_j} \tilde{X}_j P_{\gamma_j} - P^T_{\gamma_k} \tilde{X}_k P_{\gamma_k}) P^T_{\xi_k} = 0, \quad k \in \mathrm{ch}(j),\ j = 1, \ldots, l \\ & \tilde{X}_j \succeq 0, \quad j = 1, \ldots, l \end{array} \tag{40}$$

with variables $\tilde{X}_j \in \mathbf{H}^{|\gamma_j|}$, $j = 1, \ldots, l$. The coefficients $\tilde{C}_j$ and $\tilde{A}_{ij}$ are matrices that satisfy

$$\sum_{j=1}^{l} \mathbf{tr}(\tilde{C}_j X_{\gamma_j \gamma_j}) = \mathbf{tr}(CX)$$

and

$$\sum_{j=1}^{l} \mathbf{tr}(\tilde{A}_{ij} \tilde{X}_j) = \mathbf{tr}(A_i X), \quad i = 1, \ldots, m,$$

for all $X \in \mathbf{H}^n_E$. Note that P is an index matrix defined in Section 2.4.3 and ξ_k is the index set of intersection of cliques γ_k and its parent clique in the clique tree. The variables $\tilde{X}_j \in \mathbf{H}^{|\gamma_j|}$ are decomposed from the primal variable $X \in \mathbf{H}^n_{E,\mathrm{c}+}$, *i.e.*,

$$\tilde{X}_j = P_{\gamma_j} X P^T_{\gamma_j}, \quad j = 1, \ldots, l, \tag{41}$$

and the second set of equality constraints require the overlapping submatrices $\tilde{X}_j$ must be consistent.

Distributed algorithms [LZT11] are proposed to solve the above decomposed SDP relaxation of the OPF problem. But what still remains as a problem is how to construct a globally optimal solution or an approximation of the original OPF problem (36). As discussed in section 3, given an optimal solution $\tilde{X}^\star_j$ in (40), we can construct an optimal solution X° for (38) with

$$\tilde{X}^\star_j = P_{\gamma_j} X^\circ P^T_{\gamma_j}, \quad j = 1, \ldots, l.$$

And furthermore, if the network graph has a chordal sparsity pattern E, we can construct a low-rank positive semidefinite completion $X^\bullet$ of X° by Algorithm 1 with

$$X^\circ = \mathbf{proj}_E(X^\bullet),$$

and rank

$$r^\bullet = \mathbf{rank}(X^\bullet) = \max_{\text{cliques } \gamma_j} \mathbf{rank}(X^\circ_{\gamma_j\gamma_j}) = \max_{j=1,\ldots,l} \mathbf{rank}(\tilde{X}^\star_j). \tag{42}$$

This process of finding a minimum rank positive semidefinite completion for the optimal solution of SDR (38) is called posterior rounding operation. If the completed matrix $X^\bullet$ has rank 1, we can reconstruct a complex column vector $\tilde{v} \in \mathrm{C}^n$ with $X^\bullet = \tilde{v}\tilde{v}^H$ to indicate the optimal voltages in the original power network. Even if $\mathbf{rank}(X^\bullet) > 1$, the principal eigenvector u_1 of $X^\bullet$ can be used as a (good) approximation of the globally optimal voltage vector, where $X^\bullet = \sum_{i=1}^n \lambda_i u_i u_i^H$ is the eigenvalue decomposition and the eigenvalues λ_i is in a nonincreasing order.

4.3 Numerical experiments

We have implemented and tested the SDR (38) as well as the minimum rank positive semidefinite completion. The experiments are based on the benchmark problems from the MATPOWER package ([ZMST11]), and the experiments are carried out in Matlab R2015b on a laptop with an Intel Core i5 dual-core 1.8 GHz CPU and 8 GB RAM. We used SeDuMi 1.3 with tolerance 10^{-7} to solve the SDR (38) of the OPF problem. Note that although we explicitly build the complex-valued SDR (38) and apply the conversion method to this formulation, we cast the converted problem as a real-valued problem before passing it to SeDuMi. We use the Chompack [DV09] package in Python to implement the algorithm (1).

Following the convention in [MHLD13, AHV14], we eliminate transmission line flow constraints in (32) that are not active at the local optimal solution provided by MATPOWER. In other words, we delete the constraints in (32) for those lines working below their maximum capacity when the power network follow the optimal solution provided by SeDuMi. The reason of doing so is that SeDuMi does not provide a useful solution when we include all the flow constraints (32), especially when the number of power busses is around several thousands. Table 2 lists the test cases along with relevant problem dimensions. The value $n = |\mathcal{N}|$ indicates the number of power busses in the network while the value $|\mathcal{L}|$ is the number of transmission lines. The number of generators are listed in the last column of the table.

For each test case, we solve the SDR problem in the form of (40) by conversion methods, and denote the optimal solution as $X^\star_j$, $j = 1, \ldots, l$. We reconstruct the full matrix X° as an optimal solution for (38). Then we apply Algorithm 1 to construct a minimum rank positive semidefinite completion $X^\bullet$. We also solve the SDR problem (38) by SeDuMi for an optimal solution $X^\star$. We investigate the number of power buses (nodes), the numerical rank of the optimal solution for the SDR problem (38), *i.e.*, $\mathbf{rank}(X^\star)$, and the minimum rank of its positive semidefinite completion as stated in (42), *i.e.*,

$$\mathbf{rank}(X^\bullet) = \max_{\text{cliques } \gamma_j} \mathbf{rank}(X^\circ_{\gamma_j\gamma_j}) = \max_{j=1,\ldots,l} \mathbf{rank}(X^\star_j).$$

The ratio between the largest and the other eigenvalues can be used as an indicator of the numerical rank of the solution. In words, if any eigenvalue λ_i is below a certain ratio of the largest eigenvalue $\lambda_{\max}$, we take it as zero when computing the numerical rank of a matrix. Especially, the solution has numerical rank 1 if the ratio between the largest and the second largest eigenvalue is sufficiently large. Hence, for both matrix outputs, *i.e.*, $X^\star$ and $X^\bullet$, we have implemented the tests with the tolerance ϵ of $10^{-4}\sqrt{n}$, $10^{-5}\sqrt{n}$, and $10^{-6}\sqrt{n}$. To be specific, those eigenvalues that smaller than $\epsilon\lambda_{\max}$ are considered as zero when we compute the numerical rank. Based on the above

Case	$n = \lvert\mathcal{N}\rvert$	$\lvert\mathcal{L}\rvert$	$\lvert\mathcal{G}\rvert$
IEEE-118	118	186	0
IEEE-300	300	409	0
2383wp	2383	2896	92
2736sp	2736	3269	118
2737sp	2737	269	165
2746wop	2746	3307	346
2746wp	2746	3279	352
3012wp	3012	3572	7
3120sp	3120	3693	9
3375wp	3375	3693	25
89pegase	89	210	12
1354pegase	1354	1991	260
2869pegase	2869	4582	510
1888rte	1888	2531	297
1951rte	1951	2596	391
2848rte	2848	3776	547
2868rte	2868	3808	599
6468rte	6468	9000	1295
6470rte	6470	9005	1330
6495rte	6495	9019	1372
6515rte	6515	9037	1388

Table 2: Test cases and problem dimensions.

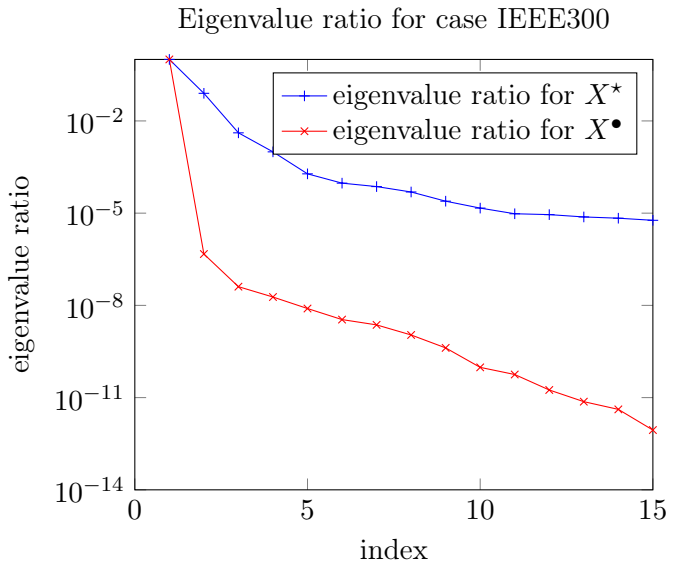


Figure 21: Eigenvalue ratio for case IEEE300

setup, the results are shown in Table 3. Note that the problem solve by SeDuMi is real-valued but we transform the result back into complex form, *i.e.*, $X^\star, X^\bullet \in \mathbf{H}^n_+$. Overall, the minimum rank completion algorithm has achieved a lower rank result compared with the interior point method SeDuMi provides. In some cases, the improvement is significant and the result can be considered to be exact within certain tolerance. However, it is also interesting to notice that the numerical rank depends on the tolerance we use. For example, in the case 2736sp, the rank is considered as 1 when the tolerance is $10^{-5}\sqrt{n}$ while it is 8 when we tighten the threshold. Also, in the last three cases when the number of busses is around 6k, the rank of the completed matrix is larger than the maximum clique size, which is impossible in theory. It occurs due to the numerical error inside MATLAB software where the eigenvalues are zero theoretically but MATLAB returns an extremely small positive values. These eigenvalues may be treated as nonzero under some small tolerance.

In our definition, the ratio between every eigenvalue and the largest eigenvalue is the criterion to decide the numerical rank. Thus it is interesting to see how this ratio changes in both cases, *i.e.*, $X^\star$ and $X^\bullet$. Figure 21 shows the eigenvalue ratio for the case IEEE-300. As we can see from the figure, the rest of the eigenvalues are comparably small compared with the largest eigenvalue of $X^\bullet$, while in the matrix $X^\star$, the eigenvalues are comparably concentrated.

Furthermore, different solvers may affect the above results. The SeDuMi solver used above is aimed for optimization over symmetric cones. However, as discussed in Section 2.5, the interior-point method, several first-order methods and splitting algorithms can also be used to exploit the sparsity inside SDPs. Table 4 shows the rank results of both matrix outputs, *i.e.*, $X^\star$ and $X^\bullet$, computed by three different solvers, SeDuMi, SDPT3 and MOSEK. Note that the numerical rank is calculated in the method mentioned above with ratio tolerance $\epsilon = 10^{-5}\sqrt{n}$. Results are slightly different in the three solvers, but they have similar outputs within a small variation. We can also see that the posterior rounding operation generates good results in all three solvers, but again the ranks of the completed matrix do not vary too much in different solvers.

			$\epsilon = 10^{-4}\sqrt{n}$		$\epsilon = 10^{-5}\sqrt{n}$		$\epsilon = 10^{-6}\sqrt{n}$	
Case	n	max. clique	$\mathbf{rank}(X^\star)$	$\mathbf{rank}(X^\bullet)$	$\mathbf{rank}(X^\star)$	$\mathbf{rank}(X^\bullet)$	$\mathbf{rank}(X^\star)$	$\mathbf{rank}(X^\bullet)$
IEEE-118	118	20	1	1	1	1	1	1
IEEE-300	300	17	5	1	5	1	36	11
2383wp	2383	31	13	1	17	1	19	3
2736sp	2736	30	1	1	1	1	14	8
2737sop	2737	29	1	1	43	1	87	9
2746wop	2746	30	1	1	32	1	76	11
2746wp	2746	31	1	1	1	1	268	17
3012wp	3012	32	281	5	346	13	578	17
3120sp	3120	32	445	32	572	32	761	32
3375wp	3375	33	442	19	451	19	518	33
89pegase	89	12	7	1	17	5	19	6
1354pegase	1354	19	97	3	111	7	124	19
2869pegase	2869	29	101	13	181	15	199	19
1888rte	1888	16	197	1	251	1	271	3
1951rte	1951	28	23	1	71	1	135	5
2848rte	2848	35	87	1	133	1	210	3
2868rte	2868	31	133	7	255	16	301	21
6468rte	6468	33	214	7	356	11	456	87
6470rte	6470	34	315	5	413	12	913	91
6495rte	6495	35	362	21	407	37	491	51
6515rte	6515	35	328	2	402	9	461	11

Table 3: Numerical rank results with different tolerance.

			MOSEK 8		SeDuMi v1.05		SDPT3 v4.0	
Case	n	max. clique	$\mathbf{rank}(X^\star)$	$\mathbf{rank}(X^\bullet)$	$\mathbf{rank}(X^\star)$	$\mathbf{rank}(X^\bullet)$	$\mathbf{rank}(X^\star)$	$\mathbf{rank}(X^\bullet)$
IEEE-118	118	20	1	1	1	1	1	1
IEEE-300	300	10	5	1	5	1	5	1
2383wp	2383	31	17	1	17	1	17	1
2736sp	2736	30	1	1	1	1	1	1
2737sop	2737	29	44	1	43	1	43	1
2746wop	2746	30	32	1	32	1	32	1
2746wp	2746	31	1	1	1	1	1	1
3012wp	3012	32	346	13	346	13	337	17
3120sp	3120	32	514	27	572	32	519	27
3375wp	3375	33	451	19	451	19	454	21
89pegase	89	12	19	5	17	5	17	5
1354pegase	1354	19	123	7	111	7	93	8
2869pegase	2869	29	183	14	181	15	167	13
1888rte	1888	16	175	15	251	1	175	15
1951rte	1951	28	71	1	71	1	70	1
2848rte	2848	35	142	1	133	1	133	1
2868rte	2868	31	255	16	255	16	223	13
6468rte	6468	33	356	11	356	11	751	13
6470rte	6470	34	413	12	413	12	409	13
6495rte	6495	35	399	35	407	37	401	34
6515rte	6515	35	412	9	402	9	378	11

Table 4: Numerical rank results with different solvers ($\epsilon = 10^{-5}\sqrt{n}$).

5 Conclusion

The paper studies the minimum rank positive semidefinite matrix completion with chordal sparsity pattern, and applies the completion algorithm to the SDP relaxation of the optimal power flow problem.

In general, it is difficult to determine the lowest rank of positive semidefinite completions for a given sparse completable matrix. However, we see that when the sparsity pattern is chordal, we can easily determine the minimum rank, as well as construct a positive semidefinite completion with the minimum rank.

We apply the algorithm to the SDP relaxation of the optimal power flow problem. We first solve the relaxation problem by exploiting the sparsity pattern of the underlying power network. Then we find a minimum rank positive semidefinite completion for the optimal solution. We see that the completion result has a lower rank compared with the solution for standard SDP formulation of the optimal power flow problem.

References

[ADV10] M. S. Andersen, J. Dahl, and L. Vandenberghe. Implementation of nonsymmetric interior-point methods for linear optimization over sparse matrix cones. *Mathematical Programming Computation*, 2(3):167–201, 2010.

[AHMR88] J. Agler, W. Helton, S. McCullough, and L. Rodman. Positive semidefinite matrices with a given sparsity pattern. *Linear Algebra and its Applications*, 107:101 – 149, 1988.

[AHV14] M. S. Andersen, A. Hansson, and L. Vandenberghe. Reduced-complexity semidefinite relaxations of optimal power flow problems. *IEEE Transactions on Power Systems*, 29(4):1855–1863, July 2014.

[Ali95] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[AVD10] M. S. Andersen, L. Vandenberghe, and J. Dahl. Linear matrix inequalities with chordal sparsity patterns and applications to robust quadratic optimization. In *2010 IEEE International Symposium on Computer-Aided Control System Design*, pages 7–12, Sept 2010.

[BDX04] S. Boyd, P. Diaconis, and L. Xiao. Fastest mixing Markov chain on a graph. *SIAM Rev.*, 46(4):667–689, April 2004.

[Ber86] C. Berge. New classes of perfect graphs. *Discrete Applied Mathematics*, 15(2):145 – 154, 1986.

[BGCL15] S. Bose, D. F. Gayme, K. M. Chandy, and S. H. Low. Quadratically constrained quadratic programs on acyclic graphs with application to power flow. *IEEE Transactions on Control of Network Systems*, 2(3):278–287, Sept 2015.

[BGLC11] S. Bose, D. F. Gayme, S. Low, and K. M. Chandy. Optimal power flow over tree networks. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1342–1348, Sept 2011.

[BLT+06] P. Biswas, T. C. Liang, K. C. Toh, Y. Ye, and T. C. Wang. Semidefinite programming approaches for sensor network localization with noisy distance measurements. *IEEE Transactions on Automation Science and Engineering*, 3(4):360–371, Oct 2006.

[BLWY06] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Trans. Sen. Netw.*, 2(2):188–220, May 2006.

[Bor99] B. Borchers. SDPLIB 1.2, a library of semidefinite programming test problems. *Optimization Methods and Software*, 11(1-4):683–690, 1999.

[Bun74] P. Buneman. A characterisation of rigid circuit graphs. *Discrete Mathematics*, 9(3):205 – 212, 1974.

[Bur03] S. Burer. Semidefinite programming in the space of partial positive semidefinite matrices. *SIAM Journal on Optimization*, 14(1):139–172, 2003.

[BWFW08] X. Bai, H. Wei, K. Fujisawa, and Y. Wang. Semidefinite programming for optimal power flow problems. *International Journal of Electrical Power & Energy Systems*, 30(6–7):383 – 392, 2008.

[BY04] P. Biswas and Y. Ye. Semidefinite programming for ad-hoc wireless sensor network localization. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, IPSN '04, pages 46–54, New York, NY, USA, 2004. ACM.

[Cha09] P. R. Chares. *Cones and Interior-Point Algorithms for Structured Convex Optimization involving Powers and Exponentials*. PhD thesis, Uni. Catholique de Louvain, 2009.

[Dan92] J. Dancis. Positive semidefinite completions of partial Hermitian matrices. *Linear Algebra and Applications*, 175:91–114, 1992.

[Die12] R. Diestel. *Graph Theory*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[Dir61] G. A. Dirac. On rigid circuit graphs. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 25(1):71–76, 1961.

[DV09] J. Dahl and L. Vandenberghe. Chompack: chordal matrix package, 2009.

[DZG12] E. Dall'Anese, H. Zhu, and G. B. Giannakis. Distributed optimal power flow for smart microgrids. *ArXiv e-prints*, November 2012.

[FCLC11] M. Farivar, C. R. Clarke, S. H. Low, and K. M. Chandy. Inverter VAR control for distribution systems with renewables. In *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 457–462, Oct 2011.

[FG65] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, 1965.

[FKK+09] 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.

[FKMN01] 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.

[Gav72] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, 1972.

[Gav74] F. Gavril. The intersection graphs of subtrees in trees are exactly the chordal graphs. *Journal of Combinatorial Theory, Series B*, 16(1):47 – 56, 1974.

[GB14] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[GHJT99] W. Glunt, T. L. Hayden, C. R. Johnson, and P. Tarazaga. Positive definite completions and determinant maximization. *Linear Algebra and its Applications*, 288:1–10, 1999.

[GJSW84] 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.

[GLTL12] L. Gan, N. Li, U. Topcu, and S. H. Low. Exact convex relaxation of optimal power flow in tree networks. *ArXiv e-prints*, August 2012.

[Gol04] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., Amsterdam, The Netherlands, The Netherlands, 2004.

[GT84] A. Griewank and P. L. Toint. On the existence of convex decompositions of partially separable functions. *Mathematical Programming*, 28(1):25–49, 1984.

[GW95] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, Nov 1995.

[Haz08] E. Hazan. Sparse approximate solutions to semidefinite programs. In E. Laber, C. Bornstein, L. T. Nogueira, and L. Faria, editors, *LATIN 2008: Theoretical Informatics: 8th Latin American Symposium, Búzios, Brazil, April 7-11, 2008. Proceedings*, pages 306–316, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[HG91] M. Huneault and F. D. Galiana. A survey of the optimal power flow literature. *IEEE Transactions on Power Systems*, 6(2):762–770, May 1991.

[Jab12] R. A. Jabr. Exploiting sparsity in SDP relaxations of the OPF problem. *IEEE Transactions on Power Systems*, 27(2):1138–1139, May 2012.

[JJZQJ09] Jin-Jun, Zhi-Quan, and M. Jiang. Two-dimensional phase unwrapping using semidefinite relaxation. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1105–1108, April 2009.

[Kak10] N. Kakimura. A direct proof for the matrix decomposition of chordal-structured positive semidefinite matrices. *Linear Algebra and its Applications*, 433(4):819 – 823, 2010.

[KHA15] S. Khoshfetrat Pakazad, A. Hansson, and M. S. Andersen. Distributed primal-dual interior-point methods for solving loosely coupled problems using message passing. *ArXiv e-prints*, February 2015.

[KHAR15] S. Khoshfetrat Pakazad, A. Hansson, M. S. Andersen, and A. Rantzer. Distributed semidefinite programming with application to large-scale system analysis. *ArXiv e-prints*, April 2015.

[Las09] J. B. Lasserre. *Moments, Positive Polynomials and Their Applications*. Imperial College Press Optimization Series. World Scientific, Singapore, 2009.

[Las15] J. B. Lasserre. *An Introduction to Polynomial and Semi-Algebraic Optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2015.

[Lau96] S. L. Lauritzen. *Graphical Models*. Oxford Statistical Science Series. Clarendon Press, 1996.

[Liu90] J. W. H. Liu. The role of elimination trees in sparse factorization. *SIAM Journal on Matrix Analysis and Applications*, 11(1):134–172, 1990.

[LL10] J. Lavaei and S. H. Low. Convexification of optimal power flow problem. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 223–232, Sept 2010.

[LL12] J. Lavaei and S. H. Low. Zero duality gap in optimal power flow problem. *IEEE Transactions on Power Systems*, 27(1):92–107, Feb 2012.

[LMS+10] Z. Q. Luo, W. K. Ma, A. M. C. So, Y. Ye, and S. Zhang. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine*, 27(3):20–34, May 2010.

[LNM07] Z. Lu, A. Nemirovski, and R. D. C. Monteiro. Large-scale semidefinite programming via a saddle point mirror-prox algorithm. *Mathematical Programming*, 109(2):211–237, 2007.

[Low14a] S. H. Low. Convex relaxation of optimal power flow, part I: formulations and equivalence. *ArXiv e-prints*, May 2014.

[Low14b] S. H. Low. Convex relaxation of optimal power flow, part II: exactness. *ArXiv e-prints*, May 2014.

[LPP89] J. G. Lewis, B. W. Peyton, and A. Pothen. A fast algorithm for reordering sparse matrices for parallel factorization. *SIAM Journal on Scientific and Statistical Computing*, 10(6):1146–1173, 1989.

[LTZ14] J. Lavaei, D. Tse, and B. Zhang. Geometry of power flows and optimization in distribution networks. *IEEE Transactions on Power Systems*, 29(2):572–583, March 2014.

[Luo03] Z.-Q. Luo. Applications of convex optimization in signal processing and digital communication. *Mathematical Programming*, 97(1):177–207, 2003.

[LZT11] A. Y. S. Lam, B. Zhang, and D. Tse. Distributed algorithms for optimal power flow problem. *ArXiv e-prints*, September 2011.

[MAEH99] J. A. Momoh, R. Adapa, and M. E. El-Hawary. A review of selected optimal power flow literature to 1993. I. nonlinear and quadratic programming approaches. *IEEE Transactions on Power Systems*, 14(1):96–104, Feb 1999.

[MEHA62] J. A. Momoh, M. E. El-Hawary, and R. Adapa. Contribution to the economic dispatch problem. *Bulletin de la Societe Francoise des Electriciens*, 8(3):431–447, 1962.

[MEHA99] J. A. Momoh, M. E. El-Hawary, and R. Adapa. A review of selected optimal power flow literature to 1993. II. Newton, linear programming and interior point methods. *IEEE Transactions on Power Systems*, 14(1):105–111, Feb 1999.

[MHLD13] D. K. Molzahn, J. T. Holzer, B. C. Lesieutre, and C. L. DeMarco. Implementation of a large-scale optimal power flow solver based on semidefinite programming. *IEEE Transactions on Power Systems*, 28(4):3987–3998, Nov 2013.

[MNH+08] A. De Maio, S. De Nicola, Y. Huang, S. Zhang, and A. Farina. Code design to optimize radar detection performance under accuracy and similarity constraints. *IEEE Transactions on Signal Processing*, 56(11):5618–5629, Nov 2008.

[MNH+09] A. De Maio, S. De Nicola, Y. Huang, Z. Q. Luo, and S. Zhang. Design of phase codes for radar performance optimization with a similarity constraint. *IEEE Transactions on Signal Processing*, 57(2):610–621, Feb 2009.

[Nes12] Y. Nesterov. Towards non-symmetric conic optimization. *Optimization Methods and Software*, 27(4-5):893–917, 2012.

[NFF+03] 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.

[Oht76] T. Ohtsuki. A fast algorithm for finding an optimal ordering for vertex elimination on a graph. *SIAM Journal on Computing*, 5(1):133–145, 1976.

[Par03] P. A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320, 2003.

[PS90] A. Pothen and C. Sun. Compact clique tree data structures in sparse matrix factorizations. *SIAM Journal on Large-Scale Numerical Optimizations*, pages 180–204, 1990.

[QDB09] Z. Qiu, G. Deconinck, and R. Belmans. A literature survey of optimal power flow problems in the electricity market context. In *2009 IEEE/PES Power Systems Conference and Exposition*, pages 1–6, March 2009.

[Roc70] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[Ros70] D. J. Rose. Triangulated graphs and the elimination process. *Journal of Mathematical Analysis and Applications*, 32(3):597 – 609, 1970.

[Ros74] D. J. Rose. On simple characterizations of k-trees. *Discrete Mathematics*, 7(3–4):317 – 322, 1974.

[RTL76] D. J. Rose, R. E. Tarjan, and G. S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing*, 5(2):266–283, 1976.

[SAV13] Y. Sun, M. S. Andersen, and L. Vandenberghe. Decomposition in conic optimization with partially separable structure. *ArXiv e-prints*, May 2013.

[Stu98] Jos F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones, 1998.

[Sun15] Y. Sun. *Decomposition Methods for Semidefinite Optimization*. PhD thesis, PhD thesis, University of California, Los Angeles, 2015.

[SV04] G. Srijuntongsiri and S. A. Vavasis. A fully sparse implementation of a primal-dual interior-point potential reduction method for semidefinite programming. *eprint arXiv:cs/0412009*, December 2004.

[SY15] A. Skajaa and Y. Ye. A homogeneous interior-point algorithm for nonsymmetric convex conic optimization. *Math. Program.*, 150(2):391–422, May 2015.

[Tay15] J. A. Taylor. *Convex Optimization of Power Systems*. Cambridge University Press, 2015.

[TY84] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984.

[VA14] L. Vandenberghe and M. S. Andersen. Chordal graphs and semidefinite optimization. *Foundations and Trends in Optimization*, 1(4):241–433, 2014.

[VB96] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Rev.*, 38(1):49–95, March 1996.

[WPS05] K. Q. Weinberger, B. D. Packer, and L. K. Saul. Nonlinear dimensionality reduction by semidefinite programming and kernel matrix factorization. In *in Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, 2005.

[WS04] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages II–988–II–995 Vol.2, June 2004.

[WS06] K. Q. Weinberger and L. K. Saul. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1683–1686. AAAI Press, 2006.

[WSZS07] K.Q. Weinberger, F. Sha, Q. Zhu, and L. Saul. Graph Laplacian regularization for large-scale semidefinite programming. In B. Schölkopf, J. Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[XSB06] L. Xiao, J. Sun, and S. Boyd. A duality view of spectral methods for dimensionality reduction. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 1041–1048, New York, NY, USA, 2006. ACM.

[Yan81] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods*, 2(1):77–79, 1981.

[ZFP+16] Y. Zheng, G. Fantuzzi, A. Papachristodoulou, P. Goulart, and A. Wynn. Fast ADMM for semidefinite programs with hordal sparsity, 2016.

[ZG14] H. Zhu and G. B. Giannakis. Power system nonlinear state estimation using distributed semidefinite programming. *IEEE Journal of Selected Topics in Signal Processing*, 8(6):1039–1050, Dec 2014.

[ZMST11] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas. Matpower: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on Power Systems*, 26(1):12–19, Feb 2011.

[ZT13] B. Zhang and D. Tse. Geometry of injection regions of power networks. *IEEE Transactions on Power Systems*, 28(2):788–797, May 2013.