Covariance selection for non-chordal graphs via chordal embedding

Joachim Dahl* Lieven Vandenberghe† Vwani Roychowdhury†

Abstract

We describe algorithms for maximum likelihood estimation of Gaussian graphical models with conditional independence constraints. This problem is also known as covariance selection, and it can be expressed as an unconstrained convex optimization problem with a closed-form solution if the underlying graph is chordal. The focus of the paper is on iterative algorithms for covariance selection with non-chordal graphs.

We first derive efficient methods for evaluating the gradient and Hessian of the log-likelihood function when the underlying graph is chordal. The algorithms are formulated as simple recursions on a clique tree associated with the graph. We also show that the gradient and Hessian mappings are easily inverted when the underlying graph is chordal. We then exploit these results to obtain efficient implementations of Newton's method and the conjugate gradient method for large non-chordal graphs, by embedding the graph in a chordal graph.

1 Introduction

We consider the problem of computing maximum likelihood (ML) estimates of the mean μ and covariance Σ of a multivariate normal variable $X \sim N(\mu, \Sigma)$, subject to the constraint that certain given pairs of variables are conditionally independent. As we will explain in §2, the conditional independence constraints prescribe the sparsity pattern of the inverse of Σ and, as a consequence, the ML estimation problem can be formulated as a convex optimization problem with Σ^{-1} as variable. The problem is also known as the *covariance selection* problem and was first studied in detail by Dempster [Dem72]. In a graph representation of the random variable X, the nodes represent the components X_i; two nodes are connected by an undirected edge if the corresponding variables are conditionally dependent. This is called a *normal* (or *Gaussian*) *graphical model* of the random variable [Lau96, chapter 7]. Closely related problems are the maximum-determinant positive definite *matrix completion* problem (see [GJSW84] and §2.3) and the analytic centering problem in semidefinite programming. Covariance selection can be also be regarded as a special case of determinant maximization with linear matrix inequality constraints [VBW98].

For the special case of a chordal graph (*i.e.*, a graph in which every cycle of length greater than three has an edge connecting nonconsecutive nodes) the solution of the problem can be expressed in closed form in terms of the principal minors of the sample covariance matrix (see, for example, [Wer80], [Lau96, §5.3]). For non-chordal graphs the ML estimate has to be computed iteratively. A straightforward application of standard optimization algorithms such as Newton's method is usually considered too expensive for larger problems, and several specialized algorithms have been

*Corresponding author. Department of Communication Technology, Aalborg University (joachim@kom.aau.dk).

†Department of Electrical Engineering, University of California, Los Angeles (vandenbe@ee.ucla.edu, vwani@ee.ucla.edu).

proposed in the literature. These include the *coordinate descent* algorithm [Dem72, SK86], which is extremely simple to implement but suffers from the slow convergence that often characterizes steepest descent algorithms. Another popular method, the *iterative proportional scaling* algorithm [SK86, Lau96], requires the enumeration of all the cliques in the graph, and therefore has an exponential complexity.

In this paper we describe techniques for improving the efficiency of Newton's method and the conjugate gradient method for covariance selection problems with large non-chordal graphs. The main innovation that contributes to the efficiency of the algorithms are fast techniques for computing the gradient and Hessian of the cost function. These algorithms use a chordal embedding of the graph, and are formulated as simple recursions on clique trees. The idea is particularly effective for problems with nearly chordal sparsity patterns.

The paper also contains several results that should be of independent interest to the fields of linear algebra and semidefinite programming. We can mention in particular the algorithms for evaluating the gradient and Hessian of the log-barrier function of sparse positive definite matrices with chordal sparsity patterns, given in §4.1.

Notation We use $\mathbf{S}^n$ to denote 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 positive (semi)definite matrices. The sparsity pattern of a sparse matrix $X \in \mathbf{S}^n$ will be characterized by specifying the set of indices $V \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$ of its nonzero entries. More precisely, X has the sparsity pattern V if $X_{ij} = X_{ji} = 0$ for $(i, j) \notin V$. We will assume that $j \geq i$ for all $(i, j) \in V$, and that $(i, i) \in V$ for $i = 1, \ldots, n$. The set of symmetric matrices with sparsity pattern V is denoted $\mathbf{S}^n_V$. The positive (semi)definite matrices in $\mathbf{S}^n_V$ are denoted by $\mathbf{S}^n_{V,+}$, respectively $\mathbf{S}^n_{V,++}$. $P_V(X)$ is the projection of a matrix $X \in \mathbf{S}^n$ on $\mathbf{S}^n_V$, *i.e.*, $Y = P_V(X)$ is the matrix in $\mathbf{S}^n_V$ with $Y_{ij} = X_{ij}$ for $(i, j) \in V$. X_{IJ} is the submatrix of X with rows indexed by $I \subseteq \{1, \ldots, n\}$ and columns indexed by $J \subseteq \{1, \ldots, n\}$. The notation X^{-1}_{IJ} denotes the matrix $(X_{IJ})^{-1}$; it is important to distinguish this from $(X^{-1})_{IJ}$.

2 Covariance selection

In this section we define the covariance selection problem and present two convex optimization formulations. In the first formulation (§2.2), the log-likelihood function is maximized subject to sparsity constraints on the inverse of the covariance matrix. This problem is convex in the inverse covariance matrix. In the second formulation (§2.3), which is related to the first by duality, the covariance matrix is expressed as the maximum-determinant positive definite completion of the sample covariance.

2.1 Conditional independence in normal distributions

Let x, y and z be random variables with continuous distributions. We say that x and y are *conditionally independent given* z if

$$f(x|y, z) = f(x|z),$$

where $f(x|z)$ is the conditional density of x given z, and $f(x|y, z)$ is the conditional density of x given y and z. Informally, this means that once we know z, knowledge of y gives no further

information about x. Conditional independence is a fundamental property in expert systems and graphical models [Lau96, CDLS99, Pea88].

In this paper we are interested in the special case of conditional independence of two coefficients x_i, x_j of a vector random variable $x = (x_1, x_2, \ldots, x_n)$, given the other coefficients, *i.e.*, the condition that

$$f(x_i|x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = f(x_i|x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n).$$

If this holds, we simply say that x_i and x_j are conditionally independent. There is a well-known characterization of conditional independence of variables with a joint *normal* distribution $N(\mu, \Sigma)$. This is easily seen from the fact that the conditional distribution of (x_i, x_j), given the other components of x, is Gaussian with covariance matrix

$$\begin{bmatrix} (\Sigma^{-1})_{ii} & (\Sigma^{-1})_{ij} \\ (\Sigma^{-1})_{ji} & (\Sigma^{-1})_{jj} \end{bmatrix}^{-1}.$$

The variables x_i and x_j are conditionally independent if and only if this covariance matrix is diagonal, *i.e.*,

$$(\Sigma^{-1})_{ij} = 0.$$

This classical result can be found in [Dem72].

2.2 Maximum likelihood estimation

We now consider the ML estimation of the parameters μ and Σ of $N(\mu, \Sigma)$, based on K independent samples y_i, and subject to the constraint that given pairs of variables are conditionally independent. As we have seen, the constraints are equivalent to specifying the sparsity pattern of Σ^{-1}. Let V be the set of upper triangular positions of Σ^{-1} that are allowed to be nonzero, so the constraints are

$$(\Sigma^{-1})_{ij} = 0, \quad (i,j) \notin V. \tag{1}$$

The log-likelihood function is, up to a constant,

$$\begin{aligned} L(\mu, \Sigma) &= -\frac{K}{2} \log\det\Sigma - \frac{1}{2}\sum_{i=1}^{K}(y_i - \mu)^T\Sigma^{-1}(y_i - \mu) \\ &= \frac{K}{2}(-\log\det\Sigma - \mathbf{tr}(\Sigma^{-1}\bar{\Sigma}) - (\mu - \bar{\mu})^T\Sigma^{-1}(\mu - \bar{\mu})) \end{aligned} \tag{2}$$

where $\bar{\mu}$ and $\bar{\Sigma}$ are the sample mean and covariance

$$\bar{\mu} = \frac{1}{K}\sum_{i=1}^{K} y_i, \qquad \bar{\Sigma} = \frac{1}{K}\sum_{i=1}^{K}(y_i - \bar{\mu})(y_i - \bar{\mu})^T.$$

The ML estimation problem with the conditional independence constraints (1) can therefore be expressed as

$$\begin{array}{ll} \text{maximize} & -\log\det\Sigma - \mathbf{tr}(\bar{\Sigma}\Sigma^{-1}) - (\mu - \bar{\mu})^T\Sigma^{-1}(\mu - \bar{\mu}) \\ \text{subject to} & (\Sigma^{-1})_{ij} = 0, \quad (i,j) \notin V, \end{array}$$

with domain $\{(\Sigma, \mu) \in \mathbf{S}^n \times \mathbf{R}^n \mid \Sigma \succ 0\}$. Clearly, the optimal value of μ is the sample mean $\bar{\mu}$, and if we eliminate the variable μ and make a change of variables $X = \Sigma^{-1}$, the problem reduces to

$$\begin{array}{ll} \text{maximize} & \log\det X - \mathbf{tr}(\bar{\Sigma}X) \\ \text{subject to} & X_{ij} = 0, \quad (i,j) \notin V \end{array} \tag{3}$$

with variable $X \in \mathbf{S}^n$. This is a convex optimization problem, since the objective function is concave on the set of positive definite matrices. Equivalently, we can restrict X to $\mathbf{S}^n_V$ and consider the unconstrained problem

$$\text{minimize} \quad -\log\det X + \mathbf{tr}(CX), \tag{4}$$

where $C = P_V(\bar{\Sigma})$, the projection of $\bar{\Sigma}$ on $\mathbf{S}^n_V$.

2.3 Duality and optimality conditions

The gradient of the objective function in (4) is $-P_V(X^{-1}) + C$, so $X \in \mathbf{S}^n_{V,++}$ is optimal if and only if

$$P_V(X^{-1}) = C. \tag{5}$$

If X is optimal, then the matrix $Z = X^{-1}$ solves the optimization problem

$$\begin{array}{ll} \text{maximize} & \log\det Z \\ \text{subject to} & P_V(Z) = C, \end{array} \tag{6}$$

with variable $Z \in \mathbf{S}^n_{++}$. This problem is also the dual of (3). In (6) we maximize the determinant of a positive definite matrix Z, subject to the constraint that Z agrees with C in the positions V. This is known as the *maximum determinant positive definite matrix completion* problem [GJSW84, Lau01].

3 Chordal graphs

An undirected graph $\mathcal{G}$ is *chordal* if every cycle of length greater than three has a chord, *i.e.*, an edge joining nonconsecutive nodes of the cycle. In the graphical models literature the terms *triangulated graph* and *decomposable graph* are also used as synonyms for chordal graph.

A sparsity pattern V defines an undirected graph $\mathcal{G}_V$ with vertices $1, \ldots, n$, and edges between nodes i and j if $(i,j) \in V$, $i \neq j$. In §4 we will describe simple algorithms for the ML estimation problem (4) and several related problems when the underlying graph $\mathcal{G}_V$ is chordal. The easiest way to derive these results is in terms of *clique trees* (also called *junction trees*) associated with the graph $\mathcal{G}_V$. In this section we summarize some important properties of chordal graphs [BP93, FKMN00, NFF+03].

3.1 Clique trees

A clique is a maximal subset of the nodes that defines a complete subgraph, *i.e.*, all pairs of nodes in the clique are connected by an edge. The cliques can be represented by an undirected graph that has the cliques as its nodes, and edges between any two cliques with a nonempty intersection. We call this graph the *clique graph* associated with $\mathcal{G}_V$. We can also assign to every edge (V_i, V_j) in the clique graph a weight equal to the number of nodes in the intersection $V_i \cap V_j$. A clique tree

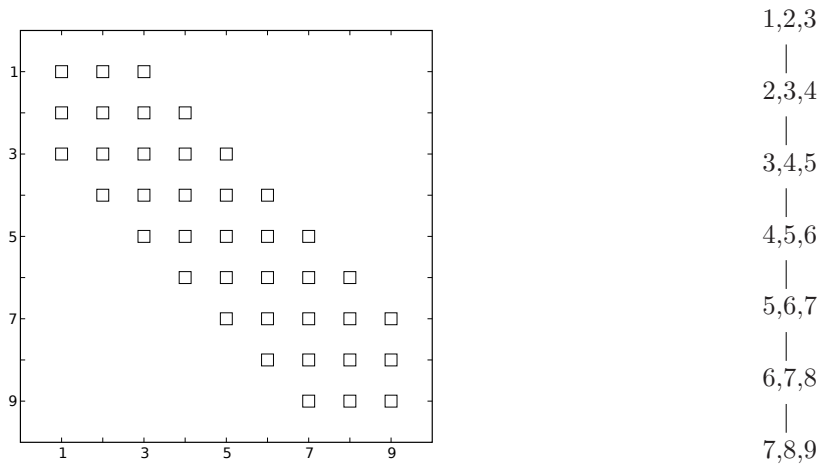

Figure 1: Sparsity pattern and clique tree of a 9×9 band matrix with bandwidth 5.

of a graph is a maximum weight spanning tree of its clique graph (as such a clique tree is non-unique). Clique trees of chordal graphs can be efficiently computed by the *maximum cardinality search* algorithm [Ros70, RTL76, TY84].

For the rest of the section we assume that there are l cliques V_1, V_2, ..., V_l in $\mathcal{G}_V$, so that the set of nonzero entries is given by

$$\{(i,j) \mid (i,j) \in V \text{ or } (j,i) \in V\} = (V_1 \times V_1) \cup (V_2 \times V_2) \cup \cdots \cup (V_l \times V_l).$$

We assume a clique tree has been computed, and we number the cliques so that V_1 is the root of the tree and every parent in the tree has a lower index than its children. We define $S_1 = V_1$, $U_1 = \emptyset$ and, for $i = 2, \ldots, l$,

$$S_i = V_i \setminus (V_1 \cup V_2 \cup \cdots \cup V_{i-1}), \qquad U_i = V_i \cap (V_1 \cup V_2 \cup \cdots \cup V_{i-1}). \tag{7}$$

It can be shown that for a chordal graph

$$S_i = V_i \setminus V_k, \qquad U_i = V_i \cap V_k \tag{8}$$

where V_k is the parent of V_i in the clique tree. This important property is known as the *running intersection property* [BP93].

Examples The simplest example of a chordal sparsity pattern is band structure. Figure 1 shows a banded matrix of order 9 with bandwidth 5. There are 7 cliques $\{k, k+1, k+2\}$ for $k = 1, \ldots, 7$. If we take as root $V_1 = \{1, 2, 3\}$ the clique tree is a chain and

$$V_k = \{k, k+1, k+2\}, \qquad S_k = \{k+2\}, \qquad U_k = \{k, k+1\}, \qquad k = 2, \ldots, 7.$$

Another simple example of a chordal sparsity pattern is the arrow pattern shown in figure 2. This graph has 7 cliques. If we take as root $V_1 = \{1, 2, 3\}$, the clique tree is as shown in the figure, and

$$V_k = \{1, 2, k+2\}, \qquad U_k = \{1, 2\}, \qquad S_k = \{k+2\}, \qquad k = 2, \ldots, 7.$$

Figure 3 shows a less obvious example with 13 cliques.

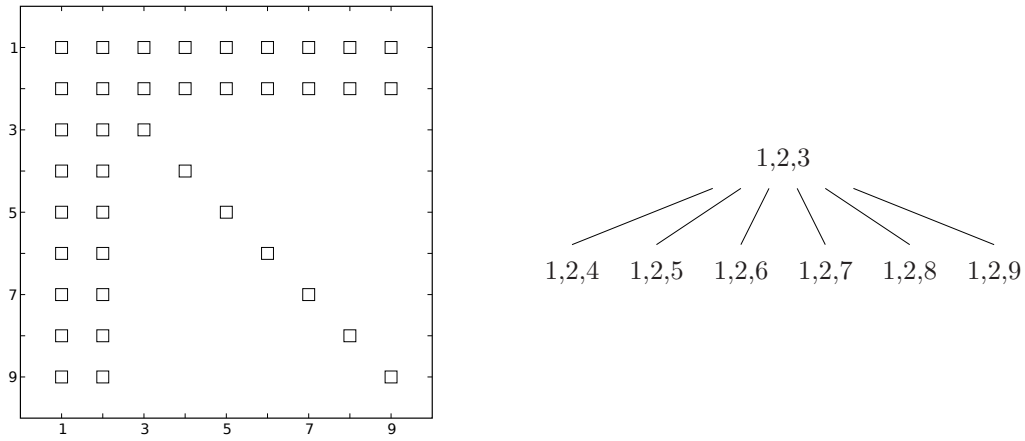

Figure 2: Sparsity pattern and clique tree of a 9×9 matrix with an arrow pattern.

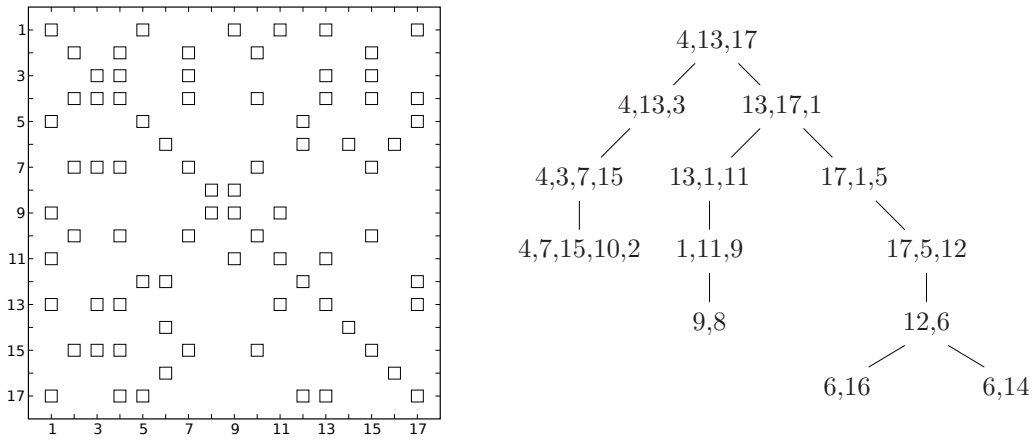

Figure 3: Sparsity pattern and clique tree of a 17×17 matrix with a chordal sparsity pattern.

3.2 Cholesky factorization with chordal sparsity pattern

If $\mathcal{G}_V$ is chordal, then a clique tree of $\mathcal{G}_V$ defines a *perfect elimination order* [BP93, FKMN00] for positive definite matrices with sparsity pattern V, *i.e.*, an elimination order that produces triangular factors with zero fill-in.

Let $X \in \mathbf{S}^n_{V,++}$ and assume that the nodes in $\mathcal{G}_V$ are numbered so that

$$S_1 = \{1, \ldots, |S_1|\}, \qquad S_k = \left\{ \sum_{j=1}^{k-1} |S_j| + 1, \quad \sum_{j=1}^{k-1} |S_j| + 2, \quad \ldots, \quad \sum_{j=1}^{k} |S_j| \right\} \text{ for } k > 1. \tag{9}$$

(In general, this assumption requires a symmetric permutation of the rows and columns of X.) We show that X can be factored as

$$X = RDR^T \tag{10}$$

where D is block diagonal with diagonal blocks $D_{S_kS_k}$, and R is unit upper triangular with zero off-diagonal entries except in positions $R_{U_kS_k}$, $k = 1, \ldots, l$. This means that $R^T + D + R$ has the same sparsity pattern as X.

Note that if we factor the matrix JXJ, where J is the identity matrix with its columns reversed, as $JXJ = RDR^T$, we obtain a factorization with lower triangular matrices

$$X = L\tilde{D}L^T$$

with $L = JRJ$ and $\tilde{D} = JDJ$ (as used, for example, in [FKMN00]).

The proof is by induction on the number of cliques. The result is obviously true if $l = 1$: If there is only one clique, then $\mathcal{G}_V$ is a complete graph so X can be factored as (10) with $R = I$, $D = X$. Next, suppose the result holds for all chordal sparsity patterns with $l - 1$ cliques. We partition X as

$$X = \begin{bmatrix} X_{WW} & X_{WS_l} \\ X_{S_lW} & X_{S_lS_l} \end{bmatrix},$$

with $W = \{1, \ldots, n\} \setminus S_l$, and examine the different blocks in the factorization

$$\begin{bmatrix} X_{WW} & X_{WS_l} \\ X_{S_lW} & X_{S_lS_l} \end{bmatrix} = \begin{bmatrix} R_{WW} & R_{WS_l} \\ 0 & I \end{bmatrix} \begin{bmatrix} D_{WW} & 0 \\ 0 & D_{S_lS_l} \end{bmatrix} \begin{bmatrix} R_{WW}^T & 0 \\ R_{WS_l}^T & I \end{bmatrix}. \tag{11}$$

We immediately notice that $D_{S_lS_l} = X_{S_lS_l}$ and $R_{WS_l}D_{S_lS_l} = X_{WS_l}$. The submatrix $X_{U_lS_l}$ of X_{WS_l} is dense, since $V_l = U_l \cup S_l$ is a clique. The submatrix $X_{W\setminus U_l,S_l}$ is zero: a nonzero entry (i, j) with $i \in W \setminus U_l$, $j \in S_l$ would mean that V_l is not the only clique that contains node j, which contradicts the definition of S_l in (7). Therefore

$$D_{S_lS_l} = X_{S_lS_l}, \qquad R_{U_lS_l} = X_{U_lS_l}D_{S_lS_l}^{-1}, \qquad R_{W\setminus U_l,S_l} = 0.$$

The rest of the factorization follows from the identity

$$X_{WW} - R_{WS_l}D_{S_lS_l}R_{WS_l}^T = R_{WW}D_{WW}R_{WW}^T$$

which is the 1,1 block of (11). The matrix

$$\tilde{X}_{WW} = X_{WW} - R_{WS_l}D_{S_lS_l}R_{WS_l}^T$$

is identical to X_{WW} except for the submatrix

$$\tilde{X}_{U_lU_l} = X_{U_lU_l} - R_{U_lS_l} D_{S_lS_l} R_{U_lS_l}^T,$$

since $R_{W\setminus U_l,S_l} = 0$. Also, since $\tilde{X}_{WW} = X_{WW} - X_{WS_l} X_{S_lS_l}^{-1} X_{S_lW}$ is a Schur complement, it is positive definite. The first term $X_{U_lU_l}$ is dense, since U_l is a subset of the clique V_l, and therefore $\tilde{X}_{WW}$ has the same sparsity pattern as X_{WW}. The sparsity pattern of X_{WW} and $\tilde{X}_{WW}$ is represented by the graph $\mathcal{G}_V$ with the nodes in S_l removed. At this point we use the running intersection property of chordal graphs (8): the fact that $U_l \subseteq V_k$, where the clique V_k is the parent of V_l in the clique tree, means that removing the nodes S_l reduces the number of cliques by one. The reduced graph is also chordal, and a clique tree of it is obtained from the clique tree of $\mathcal{G}_V$ by deleting the clique V_l. By the induction assumption the positive definite matrix $\tilde{X}_{WW}$ can therefore be factored as

$$\tilde{X}_{WW} = R_{WW} D_{WW} R_{WW}^T$$

where R_{WW} is upper triangular with the same sparsity pattern as X_{WW}. This completes the factorization (11).

To summarize the proof, we formulate the factorization algorithm as a recursion: for $k = l, l-1, \ldots, 1$,

$$D_{S_kS_k} := X_{S_kS_k}, \qquad R_{U_kS_k} := X_{U_kS_k} D_{S_kS_k}^{-1}, \qquad X_{U_kU_k} := X_{U_kU_k} - R_{U_kS_k} D_{S_kS_k} R_{U_kS_k}^T. \tag{12}$$

The following algorithm overwrites X with the factorization data.

CHOLESKY FACTORIZATION WITH CHORDAL SPARSITY PATTERN

given a matrix $X \in \mathbf{S}_{V,++}^n$ with chordal sparsity pattern V.

1. Compute a clique tree with cliques $V_1, \ldots, V_l$ numbered so that V_k has a higher index than its parents. Compute the sets S_k, U_k defined in (8).
2. For $k = l, l-1, \ldots, 2$, compute

$$X_{U_kS_k} := X_{U_kS_k} X_{S_kS_k}^{-1}, \qquad X_{U_kU_k} := X_{U_kU_k} - X_{U_kS_k} X_{S_kS_k} X_{U_kS_k}^T.$$

These steps do not alter the sparsity pattern of X but overwrite its nonzero elements with the elements of D and R. After completion of the algorithm, the diagonal blocks of D are found as $D_{S_kS_k} = X_{S_kS_k}$, $k = 1, \ldots, l$. The nonzero elements of R are its diagonal and $R_{U_kS_k} = X_{U_kS_k}$ for $k = 1, \ldots, l$. If the rows and columns of X are ordered so that (9) is satisfied, the matrix D is block diagonal and R is unit upper triangular.

Example Figure 4 shows the sparsity pattern and the clique tree for the example in figure 3, after renumbering the nodes as in (9). It can be verified that any positive definite matrix X with this sparsity pattern can be factored as RR^T, where R is upper triangular and $R + R^T$ has the same sparsity pattern as X.

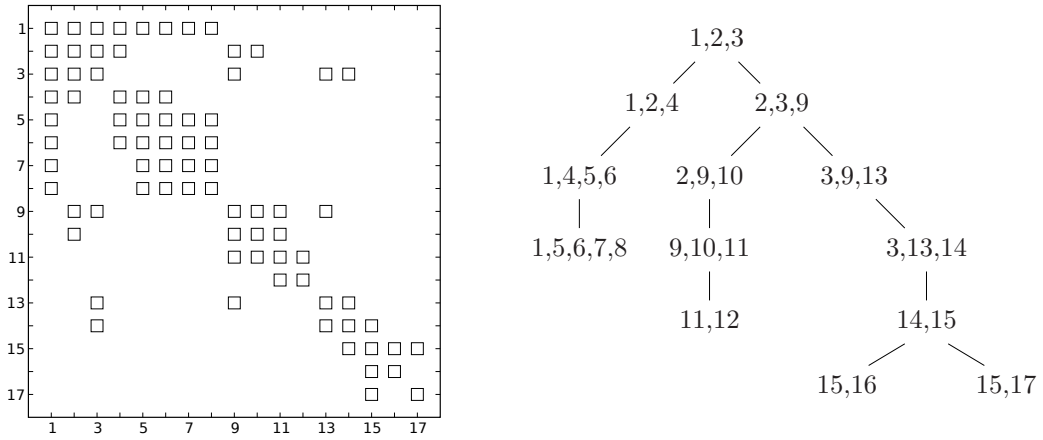

Figure 4: Sparsity pattern and clique tree of the example of figure 3, after applying a perfect elimination reordering.

Interpretation Suppose Σ^{-1} has a chordal sparsity pattern V, so it can be factored as

$$\Sigma^{-1} = (I - L)^T \tilde{D}(I - L)$$

where L is strictly lower triangular, $\tilde{D}$ is diagonal, and $L + \tilde{D} + L^T$ has the same sparsity pattern as Σ^{-1}. This factorization provides a realization of $x \sim N(0, \Sigma)$ as

$$x = Lx + v \tag{13}$$

with $v \sim N(0, \tilde{D}^{-1})$. Recursive models of this form are used in *structural equation modeling* [Bol89].

Hence, for chordal sparsity patterns, the covariance selection problem (with a prescribed sparsity pattern of Σ^{-1}) and the ML estimation problem of a model (13) (with a prescribed sparsity pattern of L) are equivalent. This is discussed in detail by Wermuth [Wer80].

4 Gradient and Hessian of $\log\det X^{-1}$ for chordal sparsity patterns

In this section we present efficient algorithms for evaluating the gradient and Hessian of $f(X) = \log\det X^{-1}$, taken as a function from $\mathbf{S}^n_V$ to $\mathbf{R}$, when the sparsity pattern V is chordal.

Throughout the section we assume that V is a chordal sparsity pattern and that a clique tree is available with l cliques $V_1, \ldots, V_l$, numbered so that every clique has a higher index than its parents. We also assume that the Cholesky factorization $X = RDR^T$ has been computed, as explained in the previous section. Finally, to simplify the description, we assume that (9) is satisfied, so the matrix D is block diagonal with diagonal blocks $D_{S_kS_k}$ and R is unit upper triangular with nonzeros in the upper triangular blocks $R_{U_kS_k}$.

4.1 Gradient

The gradient of f at X is given by

$$\nabla f(X) = -P_V(X^{-1}).$$

In general, X^{-1} is dense and may be expensive to compute. However, if the sparsity pattern V is chordal, it is possible to compute $P_V(X^{-1})$ efficiently from the Cholesky factorization $X = RDR^T$, without calculating any other entries of X^{-1}.

To see this, we examine the (S_k, V_k) block of the equation $DR^T X^{-1} = R^{-1}$:

$$D_{S_kS_k} \begin{bmatrix} R_{U_kS_k}^T & I \end{bmatrix} \begin{bmatrix} (X^{-1})_{U_kU_k} & (X^{-1})_{U_kS_k} \\ (X^{-1})_{S_kU_k} & (X^{-1})_{S_kS_k} \end{bmatrix} = \begin{bmatrix} 0 & I \end{bmatrix}.$$

We first observe that, given $(X^{-1})_{U_kU_k}$, we can compute $(X^{-1})_{S_kU_k}$ from the equation

$$R_{U_kS_k}^T (X^{-1})_{U_kU_k} + (X^{-1})_{S_kU_k} = 0. \tag{14}$$

By symmetry, this yields $(X^{-1})_{U_kS_k}$, and $(X^{-1})_{S_kS_k}$ follows from the equation

$$R_{U_kS_k}^T (X^{-1})_{U_kS_k} + (X^{-1})_{S_kS_k} = D_{S_kS_k}^{-1}. \tag{15}$$

This means that if we know $(X^{-1})_{U_kU_k}$, the rest of the submatrix $(X^{-1})_{V_kV_k}$ is straightforward to compute. At the root clique, U_1 is empty and (14), (15) reduce to

$$(X^{-1})_{S_1S_1} = D_{S_1S_1}^{-1}.$$

If we process the cliques in the order $k = 1, \ldots, l$, then the matrix $(X^{-1})_{U_kU_k}$ is available at each step, because it was calculated earlier as part of $(X^{-1})_{V_iV_i}$ where V_i is the parent clique of V_k. We therefore obtain the following recursive formulas for $Y = -P_V(X^{-1})$: for $k = 1, \ldots, l$.

$$Y_{S_kU_k} := -R_{U_kS_k}^T Y_{U_kU_k}, \qquad Y_{U_kS_k} := Y_{S_kU_k}^T, \qquad Y_{S_kS_k} := -D_{S_kS_k}^{-1} - R_{U_kS_k}^T Y_{U_kS_k}. \tag{16}$$

We summarize this by outlining an algorithm that returns $Y = -P_V(X^{-1})$.

GRADIENT OF $\log\det X^{-1}$

given a matrix $X \in \mathbf{S}_{V,++}^n$ with chordal sparsity pattern V.

1. Compute a clique tree with cliques $V_1, \ldots, V_l$ numbered so that V_k has a higher index than its parents. Compute the sets S_k, U_k defined in (8).
2. Compute the factorization $X = RDR^T$ by the algorithm in §3.2.
3. Starting at $Y := 0$, evaluate (16) for $k = 1, \ldots, l$.

Maximum likelihood estimate

The formulas (14) and (15) also provide the solution to the ML estimation problem (5). To solve the nonlinear equation

$$P_V(X^{-1}) = C,$$

we substitute $(X^{-1})_{V_kV_k} = C_{V_kV_k}$ in (14) and (15) and solve for R and D. This gives

$$R_{U_kS_k} = -C_{U_kU_k}^{-1} C_{U_kS_k}, \qquad D_{S_kS_k} = \left(C_{S_kS_k} - C_{S_kU_k} C_{U_kU_k}^{-1} C_{U_kS_k}\right)^{-1}. \tag{17}$$

From R and D we then compute $X = RDR^T$.

This gives a simple derivation of the classical analytical formulas for the ML estimate in normal graphical models, or, equivalently, the maximum determinant matrix completion, for chordal graphs. The result can be found, in different forms, in [GJSW84, BJL89], [Lau96, page 146], [FKMN00, §2], [NFF+03] and [Wer80, §3.2].

4.2 Hessian

The Hessian of f at X applied to a matrix $\Delta X \in \mathbf{S}_V^n$ is given by

$$\nabla^2 f(X)[\Delta X] = P_V(X^{-1}\Delta X X^{-1}). \tag{18}$$

We are interested in evaluating this expression using the Cholesky factorization of X, without explicitly forming X^{-1} or the product $X^{-1}\Delta X X^{-1}$.

Recall that the computation of the gradient involves two nonlinear recursions. A first recursion maps X to its triangular factors R and D. This recursion passes through the cliques in the order $l, \ldots, 1$. A second recursion computes $P_V(X^{-1})$ from the Cholesky factors, and traverses the cliques in the opposite order $1, \ldots, l$. The quantity (18) can be evaluated by linearizing the two recursions, *i.e.*, by computing the derivative

$$P_V(X^{-1}\Delta X X^{-1}) = -\frac{d}{d\alpha} P_V((X + \alpha\Delta X)^{-1})\Big|_{\alpha=0}$$

via the chain rule.

The linearized Cholesky factorization is

$$X + \Delta X = (R + \Delta R)(D + \Delta D)(R + \Delta R)^T \approx RDR^T + RD\Delta R^T + R\Delta D R^T + \Delta R D R^T.$$

The matrices ΔR, ΔD follow from (12): for $k = l, \ldots, 1$,

$$\begin{aligned}
\Delta D_{S_kS_k} &:= \Delta X_{S_kS_k} \\
\Delta R_{U_kS_k} &:= (\Delta X_{U_kS_k} - X_{U_kS_k} D_{S_kS_k}^{-1} \Delta D_{S_kS_k}) D_{S_kS_k}^{-1} \\
&= (\Delta X_{U_kS_k} - R_{U_kS_k} \Delta X_{S_kS_k}) D_{S_kS_k}^{-1} \\
\Delta X_{U_kU_k} &:= \Delta X_{U_kU_k} - R_{U_kS_k} D_{S_kS_k} \Delta R_{U_kS_k}^T - \Delta R_{U_kS_k} D_{S_kS_k} R_{U_kS_k}^T - R_{U_kS_k} \Delta D_{S_kS_k} R_{U_kS_k}^T \\
&= \Delta X_{U_kU_k} - R_{U_kS_k} \Delta X_{U_kS_k}^T - \Delta X_{U_kS_k} R_{U_kS_k}^T + R_{U_kS_k} \Delta X_{S_kS_k} R_{U_kS_k}^T.
\end{aligned}$$

If we overwrite $\Delta X_{S_kS_k}$ with $\Delta D_{S_kS_k}$, and $\Delta X_{U_kS_k}$ with $\Delta R_{U_kS_k} D_{S_kS_k}$, this can be written compactly in matrix form:

$$\begin{bmatrix} \Delta X_{U_kU_k} & \Delta X_{U_kS_k} \\ \Delta X_{S_kU_k} & \Delta X_{S_kS_k} \end{bmatrix} := \begin{bmatrix} I & -R_{U_kS_k} \\ 0 & I \end{bmatrix} \begin{bmatrix} \Delta X_{U_kU_k} & \Delta X_{U_kS_k} \\ \Delta X_{S_kU_k} & \Delta X_{S_kS_k} \end{bmatrix} \begin{bmatrix} I & 0 \\ -R_{U_kS_k}^T & I \end{bmatrix},$$

for $k = l, \ldots, 1$. The value of ΔX at the end of this iteration is

$$\Delta X_{S_kS_k} = \Delta D_{S_kS_k}, \qquad \Delta X_{U_kS_k} = \Delta R_{U_kS_k} D_{S_kS_k}, \qquad k = 1, \ldots, L.$$

Linearizing the second recursion (16) gives

$$\begin{aligned}
\Delta Y_{U_kS_k} &= -\Delta Y_{U_kU_k} R_{U_kS_k} + (X^{-1})_{U_kU_k} \Delta R_{U_kS_k} \\
\Delta Y_{S_kS_k} &= D_{S_kS_k}^{-1} \Delta D_{S_kS_k} D_{S_kS_k}^{-1} - R_{U_kS_k}^T \Delta Y_{U_kS_k} + \Delta R_{U_kS_k}^T (X^{-1})_{U_kS_k} \\
&= D_{S_kS_k}^{-1} \Delta D_{S_kS_k} D_{S_kS_k}^{-1} - R_{U_kS_k}^T \Delta Y_{U_kS_k} - \Delta R_{U_kS_k}^T (X^{-1})_{U_kU_k} R_{U_kS_k} \\
&= D_{S_kS_k}^{-1} \Delta D_{S_kS_k} D_{S_kS_k}^{-1} - R_{U_kS_k}^T (X^{-1})_{U_kU_k} \Delta R_{U_kS_k} - \Delta R_{U_kS_k}^T (X^{-1})_{U_kU_k} R_{U_kS_k} \\
&\quad + R_{U_kS_k}^T \Delta Y_{U_kU_k} R_{U_kS_k}.
\end{aligned}$$

In matrix notation, if we initialize ΔY as

$$\Delta Y_{S_kS_k} = D^{-1}_{S_kS_k}\Delta D_{S_kS_k}D^{-1}_{S_kS_k}, \qquad \Delta Y_{U_kS_k} = (X^{-1})_{U_kU_k}\Delta R_{U_kS_k}$$

and run the iteration

$$\begin{bmatrix} \Delta Y_{U_kU_k} & \Delta Y_{U_kS_k} \\ \Delta Y_{S_kU_k} & \Delta Y_{S_kS_k} \end{bmatrix} := \begin{bmatrix} I & 0 \\ -R^T_{U_kS_k} & I \end{bmatrix} \begin{bmatrix} \Delta Y_{U_kU_k} & \Delta Y_{U_kS_k} \\ \Delta Y_{S_kU_k} & \Delta Y_{S_kS_k} \end{bmatrix} \begin{bmatrix} I & -R_{U_kS_k} \\ 0 & I \end{bmatrix}$$

for $k = 1, \ldots, l$, then at completion, $\Delta Y = P_V(X^{-1}\Delta X X^{-1})$

To summarize, we can factor the Hessian as a composition of three linear functions,

$$\nabla^2 f(X) = \mathcal{A}_{\rm adj} \circ \mathcal{B} \circ \mathcal{A}. \tag{19}$$

These three mappings are easily evaluated using the factorization $X = RDR^T$ and the projected inverse $P_V(X^{-1})$.

- To evaluate $\Delta X := \mathcal{A}(\Delta X)$ we run the recursion

$$\begin{bmatrix} \Delta X_{U_kU_k} & \Delta X_{U_kS_k} \\ \Delta X_{S_kU_k} & \Delta X_{S_kS_k} \end{bmatrix} := \begin{bmatrix} I & -R_{U_kS_k} \\ 0 & I \end{bmatrix} \begin{bmatrix} \Delta X_{U_kU_k} & \Delta X_{U_kS_k} \\ \Delta X_{S_kU_k} & \Delta X_{S_kS_k} \end{bmatrix} \begin{bmatrix} I & 0 \\ -R^T_{U_kS_k} & I \end{bmatrix} \tag{20}$$

  for $k = l, \ldots, 1$.

- To evaluate $\Delta X := \mathcal{B}(\Delta X)$, we compute

$$\Delta X_{S_kS_k} := D^{-1}_{S_kS_k}\Delta X_{S_kS_k}D^{-1}_{S_kS_k}, \qquad \Delta X_{U_kS_k} := (X^{-1})_{U_kU_k}\Delta X_{U_kS_k}D^{-1}_{S_kS_k} \tag{21}$$

  for $k = 1, \ldots, l$.

- To evaluate $\Delta X := \mathcal{A}_{\rm adj}(\Delta X)$ we run the recursion

$$\begin{bmatrix} \Delta X_{U_kU_k} & \Delta X_{U_kS_k} \\ \Delta X_{S_kU_k} & \Delta X_{S_kS_k} \end{bmatrix} := \begin{bmatrix} I & 0 \\ -R^T_{U_kS_k} & I \end{bmatrix} \begin{bmatrix} \Delta X_{U_kU_k} & \Delta X_{U_kS_k} \\ \Delta X_{S_kU_k} & \Delta X_{S_kS_k} \end{bmatrix} \begin{bmatrix} I & -R_{U_kS_k} \\ 0 & I \end{bmatrix} \tag{22}$$

  for $k = 1, \ldots, l$.

Note that $\mathcal{B}$ is a self-adjoint positive definite mapping and that $\mathcal{A}_{\rm adj}$ is the adjoint of $\mathcal{A}$.

The following summary lists the different steps in the evaluation of $\Delta X := P_V(X^{-1}\Delta X X^{-1})$.

HESSIAN EVALUATION OF $\log\det X^{-1}$

given matrices $X \in \mathbf{S}^n_{V,++}$ and $\Delta X \in \mathbf{S}^n_V$ with chordal sparsity patterns V.

1. Compute a clique tree with cliques $V_1, \ldots, V_l$ numbered so that V_k has a higher index than its parents. Compute the sets S_k, U_k defined in (8).
2. Compute the factorization $X = RDR^T$ by the algorithm in §3.2.
3. Compute the gradient $-P_V(X^{-1})$ by the algorithm of §4.1.
4. Evaluate $\Delta X := \mathcal{A}_{\rm adj}(\mathcal{B}(\mathcal{A}(\Delta X)))$ via (20), (21) and (22).

For future reference we note that the mapping $\mathcal{B}$ can be factored as $\mathcal{B} = \mathcal{C}_{\rm adj} \circ \mathcal{C}$, where $\Delta X = \mathcal{C}(\Delta X)$ is computed via

$$\Delta X_{S_k S_k} := L_k^{-1} \Delta X_{S_k S_k} L_k^{-T} \qquad \Delta X_{U_k S_k} := T_k^T \Delta X_{U_k S_k} L_k^{-T}, \qquad k = 1, \ldots, l,$$

and $D_{S_k S_k} = L_k L_k^T$ and $(X^{-1})_{U_k U_k} = T_k T_k^T$ are Cholesky factorizations. The adjoint operation $\Delta X := \mathcal{C}_{\rm adj}(\Delta X)$ is

$$\Delta X_{S_k S_k} := L_k^{-T} \Delta X_{S_k S_k} L_k^{-1} \qquad \Delta X_{U_k S_k} := T_k \Delta X_{U_k S_k} L_k^{-1}, \qquad k = 1, \ldots, l.$$

This provides a factorization of $\nabla^2 f(X)$ as

$$\nabla^2 f(X) = \mathcal{A}_{\rm adj} \circ \mathcal{C}_{\rm adj} \circ \mathcal{C} \circ \mathcal{A}. \tag{23}$$

Newton equation

The three linear mappings in the factorization (19) are easily inverted, so we can use the factorization to solve equations of the form

$$P_V(X^{-1} \Delta X X^{-1}) = B$$

with variable $\Delta X \in \mathbf{S}_V^n$. The solution $\Delta X = \mathcal{A}^{-1}(\mathcal{B}^{-1}(\mathcal{A}_{\rm adj}^{-1}(B)))$ is found as follows:

- Set $\Delta X = B$ and evaluate $\mathcal{A}_{\rm adj}^{-1}(\Delta X)$:

$$\begin{bmatrix} \Delta X_{U_k U_k} & \Delta X_{U_k S_k} \\ \Delta X_{S_k U_k} & \Delta X_{S_k S_k} \end{bmatrix} := \begin{bmatrix} I & 0 \\ R_{U_k S_k}^T & I \end{bmatrix} \begin{bmatrix} \Delta X_{U_k U_k} & \Delta X_{U_k S_k} \\ \Delta X_{S_k U_k} & \Delta X_{S_k S_k} \end{bmatrix} \begin{bmatrix} I & R_{U_k S_k} \\ 0 & I \end{bmatrix}$$

  for $k = l, \ldots, 1$.

- Evaluate $\mathcal{B}^{-1}(\Delta X)$:

$$\Delta X_{S_k S_k} := D_{S_k S_k} \Delta X_{S_k S_k} D_{S_k S_k}, \qquad \Delta X_{U_k S_k} := (X^{-1})_{U_k U_k}^{-1} \Delta X_{U_k S_k} D_{S_k S_k}$$

  for $k = 1, \ldots, l$.

- Evaluate $\mathcal{A}^{-1}(\Delta X)$:

$$\begin{bmatrix} \Delta X_{U_k U_k} & \Delta X_{U_k S_k} \\ \Delta X_{S_k U_k} & \Delta X_{S_k S_k} \end{bmatrix} := \begin{bmatrix} I & R_{U_k S_k} \\ 0 & I \end{bmatrix} \begin{bmatrix} \Delta X_{U_k U_k} & \Delta X_{U_k S_k} \\ \Delta X_{S_k U_k} & \Delta X_{S_k S_k} \end{bmatrix} \begin{bmatrix} I & 0 \\ R_{U_k S_k}^T & I \end{bmatrix}$$

  for $k = 1, \ldots, l$.

5 Chordal embedding

A chordal sparsity pattern $\tilde{V}$ that contains V is known as a *chordal embedding* or *triangulation* of V. A good heuristic for computing chordal embeddings is to generate a fill-in reducing ordering for $\mathbf{S}_V^n$ (for example, an approximate minimum degree ordering [ADD96]), followed by a symbolic

Range I	#cliques with $\lvert U_k\rvert \in I$	#cliques with $\lvert S_k\rvert \in I$
1–30	3599	3646
31–60	22	1
61–90	9	2
91–120	3	1
121–150	10	0
151–180	1	0
181–210	1	0
211–240	4	0
241–270	1	0

Table 1: Distribution of the sizes of the clique subsets U_k and S_k in a clique tree for the chordal embedding of randomly generated sparse matrix of order 4000. For each bin I, we show the number of cliques with $|U_k| \in I$ and the number of cliques with $|S_k| \in I$.

Cholesky factorization. The sparsity pattern $\tilde{V}$ of the Cholesky factor defines a chordal embedding of V.

Chordal embeddings, in combination with the results of the previous section, are useful for evaluating gradients and Hessians of a function

$$g(x) = -\log\det(A_0 + x_1A_1 + \cdots + x_mA_m),$$

when the coefficients $A_k \in \mathbf{S}^n$ are sparse. If we embed the aggregate sparsity pattern of the coefficient matrices in a chordal sparsity pattern V, we can apply the methods of §4 to evaluate

$$\nabla g(x)_i = \mathbf{tr}(A_iP_{\tilde{V}}(X^{-1})), \qquad \nabla^2 g(x)_{ij} = \mathbf{tr}(A_iP_{\tilde{V}}(X^{-1}A_jX^{-1})), \qquad i,j = 1,\ldots,m,$$

where $X = A_0 + \sum_k x_kA_k$. Alternatively, we can use the factorization (23) and evaluate the Hessian as $\nabla^2 g(x)_{ij} = \mathbf{tr}(\tilde{A}_i\tilde{A}_j)$, where $\tilde{A}_i = \mathcal{C}(\mathcal{A}(A_i))$.

Example We randomly generated a positive definite matrix X of order 4000 with 14,938 non-zero elements. A symmetric minimum-degree reordering results in a a chordal embedding $\tilde{V}$ with 130,046 non-zero entries and 3650 cliques. Table 1 shows the distribution of the sizes of the clique subsets U_k and S_k (defined in (7)).

On a 2.8GHz Pentium IV PC with 2GB RAM it took approximately 12.7 seconds to compute $P_V(X^{-1})$ by computing the entire inverse using Matlab's sparse Cholesky factorization. It took only 0.32 seconds to compute $P_V(X^{-1})$ by first computing $P_{\tilde{V}}(X^{-1})$ using the algorithm of §4.1 and then discarding the entries in $\tilde{V} \setminus V$.

This fast technique for evaluating $P_V(X^{-1})$ from a factorization of X is related to Erisman and Tinney's method [ET75].

6 Maximum likelihood estimation in non-chordal graphs

If the underlying graph $\mathcal{G}_V$ is not chordal, the ML estimation problem (4) and its dual (6) require iterative solution methods. These two versions of the problem are smooth, unconstrained and

convex. They can be solved efficiently by standard implementations of Newton's method if the sparsity pattern is quite sparse (the number of variables in (4) is relatively small, *e.g.*, a few thousand), or quite dense (the number of variables in (6) is relatively small). (Duality was also exploited for this purpose in the interior-point methods for approximate positive definite completion proposed in [JKW98].) In intermediate cases, Newton's method is usually not considered a practical option, due to the high cost of assembling and inverting the Hessians. A number of specialized algorithms have therefore been proposed in the graphical models literature. In this section we first discuss two of the most popular methods, the *coordinate descent* method [Dem72, WS77] and the *iterative proportional scaling* algorithm [SK86, Lau96], and discuss their limitations.

In §6.3 and §6.4 we present two new techniques for covariance selection in non-chordal graphs. In §6.3 we describe an efficient implementation of Newton's method for problems with nearly chordal graphs, *i.e.*, graphs that can be embedded in a chordal graph by adding a relatively small number (*e.g.*, a few thousand) edges. The implementation is based on the results of §4.2. In §6.4 we formulate a preconditioner for the conjugate gradient method, also based on the ideas of §4.2. An experimental comparison is presented in §6.5.

6.1 Coordinate descent

In the coordinate descent algorithm we solve (4) one variable at a time. At each iteration, the gradient of $f(X) = -\log\det X + \mathbf{tr}(CX)$ is computed, and the nonzero entry X_{ij} with $(i,j) = \operatorname{argmax}|\partial f(X)/\partial X_{ij}|$ is updated, by minimizing f over X_{ij} while keeping the other entries fixed. This coordinate-wise minimization is repeated until convergence. The method is also known as steepest descent in ℓ_1-norm, and its convergence follows from standard results in unconstrained convex minimization [BV04, §9.4.2], [BT97, page 206].

Coordinate descent is a natural choice for the covariance selection problem, because each iteration is very cheap [Dem72, SK86, Wer80]. Suppose we want to update X as $X := X + s(e_ie_j^T + e_je_i^T)$, where s minimizes

$$f(X + s(e_ie_j^T + e_je_i^T)) = \mathbf{tr}(CX) + 2sC_{ij} - \log\det(X + s(e_ie_j^T + e_je_i^T)). \tag{24}$$

To simplify the determinant we use the formula for the determinant of a 2 by 2 block matrix. With $\Sigma = X^{-1}$,

$$\det(X + s(e_ie_j^T + e_je_i^T)) = \det\begin{bmatrix} X & se_i & se_j \\ e_j^T & -1 & 0 \\ e_i^T & 0 & -1 \end{bmatrix} = \det X \det\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + s\begin{bmatrix} \Sigma_{ij} & \Sigma_{jj} \\ \Sigma_{ii} & \Sigma_{ij} \end{bmatrix}\right).$$

We note that

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + s\begin{bmatrix} \Sigma_{ij} & \Sigma_{jj} \\ \Sigma_{ii} & \Sigma_{ij} \end{bmatrix} = \begin{bmatrix} 0 & \Sigma_{ii}^{-1/2} \\ \Sigma_{jj}^{-1/2} & 0 \end{bmatrix}\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + t\begin{bmatrix} \rho & 1 \\ 1 & \rho \end{bmatrix}\right)\begin{bmatrix} 0 & \Sigma_{jj}^{1/2} \\ \Sigma_{ii}^{1/2} & 0 \end{bmatrix}$$

where $\rho = \Sigma_{ij}/(\Sigma_{ii}\Sigma_{jj})^{1/2}$ and $t = (\Sigma_{ii}\Sigma_{jj})^{1/2}s$. If we also define $\bar\rho = C_{ij}/(\Sigma_{ii}\Sigma_{jj})^{1/2}$, we can write (24) as

$$\begin{aligned} f(X + s(e_ie_j^T + e_je_i^T)) &= f(X) + 2t\bar\rho - \log\det\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + t\begin{bmatrix} \rho & 1 \\ 1 & \rho \end{bmatrix}\right) \\ &= f(X) + 2t\bar\rho - \log((1+t\rho)^2 - t^2), \end{aligned}$$

so it is clear that we need to minimize the function

$$g(t) = 2t\bar{\rho} - \log((1+t\rho)^2 - t^2), \qquad \mathbf{dom}\, g = \begin{cases} (-\infty, 1/2) & \rho = -1 \\ (-(1+\rho)^{-1}, (1-\rho)^{-1}) & -1 < \rho < 1 \\ (-1/2, \infty) & \rho = 1. \end{cases}$$

If $\rho = -1$, then g is bounded below if and only if $\bar{\rho} < 0$, in which case the optimal solution is $t = (1+\bar{\rho})/(2\bar{\rho})$. If $\rho = 1$, then g is bounded below if and only if $\bar{\rho} > 0$, in which case the optimal solution is $t = (1-\bar{\rho})/(2\bar{\rho})$. If $-1 < \rho < 1$, then g is bounded below for all values of $\bar{\rho}$, and we can find the minimum by setting the derivative equal to zero:

$$\bar{\rho} = \frac{1}{t + (1+\rho)^{-1}} + \frac{1}{t - (1-\rho)^{-1}}.$$

This gives a quadratic equation in t,

$$\bar{\rho}(1-\rho^2)t^2 - (1 - \rho^2 + 2\rho\bar{\rho})t + \rho - \bar{\rho} = 0,$$

with exactly one root in the interval $(-(1+\rho)^{-1}, (1-\rho)^{-1})$ (the unique root if $\bar{\rho} = 0$, the smallest root if $\bar{\rho} > 0$, and the largest root if $\bar{\rho} < 0$). Hence, we obtain a simple analytical expression for the optimal step size s that minimizes (24). After calculating the optimal s, the Cholesky factorization of X can be updated efficiently via a rank-one update or downdate [GL96, page 611], [GGMS74].

The main advantage of the coordinate descent algorithm is its simplicity. The convergence rate is often slow, as is well known for steepest descent algorithms in general.

6.2 Iterative proportional scaling

Speed and Kiverii [SK86] and Lauritzen [Lau96, page 134] discuss the following iterative method. First, the cliques V_k, $k = 1, \ldots, l$, in the graph $\mathcal{G}_V$ are enumerated. Starting at an initial $X \in \mathbf{S}^n_{V,++}$, the algorithm cycles through the cliques. Each cycle consists of l updates

$$X_{V_kV_k} := X_{V_kV_k} + C^{-1}_{V_kV_k} - (X^{-1})^{-1}_{V_kV_k}, \quad k = 1, \ldots, l. \tag{25}$$

The interpretation is as follows. From the expression $(X^{-1})_{V_kV_k} = (X_{V_kV_k} - X_{V_kW_k}X^{-1}_{W_kW_k}X_{W_kV_k})^{-1}$ where $W_k = \{1, \ldots, n\} \setminus V_k$, we see that in the kth step of (25), $X_{V_kV_k}$ is modified so that

$$(X^{-1})_{V_kV_k} = (X_{V_kV_k} - X_{V_kW_k}X^{-1}_{W_kW_k}X_{W_kV_k})^{-1} = C_{V_kV_k}.$$

In other words $X_{V_kV_k}$ is adjusted so that the V_k, V_k block of the optimality condtion $P_V(X^{-1}) = C$ is satisfied. It can be shown that this iterative procedure converges to the solution [SK86].

For chordal graphs, the iterative proportional scaling (IPS) algorithm terminates after one cycle if it is started at $X = I$ and the cliques are enumerated in the order defined in section 3.1 (see [SK86, §4.4]). To see this, recall from section 4.1 that if $\mathcal{G}_V$ is chordal, the optimal solution can be factored as $\tilde{X} = RDR^T$, where R is unit upper triangular and D is block-diagonal with nonzero blocks given by (17). The product $\tilde{X} = RDR^T$ can be computed recursively as

$$\begin{aligned} \begin{bmatrix} \tilde{X}_{U_kU_k} & \tilde{X}_{U_kS_k} \\ \tilde{X}_{S_kU_k} & \tilde{X}_{S_kS_k} \end{bmatrix} &:= \begin{bmatrix} \tilde{X}_{U_kU_k} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} R_{U_kS_k} \\ I \end{bmatrix} D_{S_kS_k} \begin{bmatrix} R_{U_kS_k} \\ I \end{bmatrix}^T \\ &= \begin{bmatrix} \tilde{X}_{U_kU_k} - C^{-1}_{U_kU_k} & 0 \\ 0 & 0 \end{bmatrix} + C^{-1}_{V_kV_k} \end{aligned} \tag{26}$$

for $k = 1, \ldots, l$. The second line follows from the formula for the inverse of a 2×2 block matrix and the expressions of $R_{U_k S_k}$ and $D_{S_k S_k}$ in (17). The starting value of $\tilde{X}$ in this recursion is irrelevant because $U_1 = \emptyset$.

We now verify that this iteration gives the same result as (25). The first $k-1$ steps of (25) only modfiy the submatrices $X_{V_i V_i}$, $i < k$, so after $k-1$ steps we still have $X_{S_i S_i} = I$, $X_{U_i S_i} = 0$ for $i = k, \ldots, l$. We can therefore write (25) in partitioned form as

$$\begin{bmatrix} X_{U_k U_k} & X_{U_k S_k} \\ X_{S_k U_k} & X_{S_k S_k} \end{bmatrix} := \begin{bmatrix} X_{U_k U_k} & 0 \\ 0 & I \end{bmatrix} + C^{-1}_{V_k V_k} - \begin{bmatrix} (X^{-1})^{-1}_{U_k U_k} & 0 \\ 0 & I \end{bmatrix}, \quad k = 1, \ldots, l. \tag{27}$$

Comparing (27) and (26) it is readily shown that the two recursions provide the same result. Suppose that after $k-1$ steps of the two iterations, we have

$$X_{V_i V_i} = \tilde{X}_{V_i V_i}, \quad i = 1, \ldots, k-1. \tag{28}$$

This implies that the leading part of X, with rows and columns in $V_1 \cup \cdots \cup V_{k-1}$, solves the covariance selection problem for the (chordal) subgraph of $\mathcal{G}_V$ with cliques $V_1, \ldots, V_{k-1}$. Therefore $(X^{-1})_{U_k U_k} = C_{U_k U_k}$ in (27), and comparing with (26), we see that $X_{V_k V_k} = \tilde{X}_{V_k V_k}$ after the kth step of the two iterations. Since $U_1 = \emptyset$, we also have $X_{V_1 V_1} = \tilde{X}_{V_1 V_1}$ after the first step. By induction it follows that $X = \tilde{X}$ after l steps.

Although the IPS algorithm often achieves a good accuracy in a small number of cycles, its application to large non-chordal graphs is fundamentally limited by the very high complexity of enumerating all the cliques in a general graph.

6.3 Newton algorithm for nearly chordal graphs

Let $\tilde{V}$ be a chordal embedding of the sparsity pattern V. We can reformulate (4) as

$$\begin{array}{ll} \text{minimize} & -\log\det X + \mathbf{tr}(CX) \\ \text{subject to} & X_{ij} = 0, \quad (i,j) \in \tilde{V} \setminus V, \end{array}$$

with variable $X \in \mathbf{S}^n_{\tilde{V}}$. If the number of equality constraints is small, this problem can be solved efficiently using Newton's method in combination with the results of §4.

We explain the idea for the more general problem

$$\begin{array}{ll} \text{minimize} & -\log\det X + \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_j X) = b_j, \quad j = 1, \ldots, m, \end{array} \tag{29}$$

with variable $X \in \mathbf{S}^n_V$, where V is a chordal sparsity pattern. Without loss of generality we can assume $C \in \mathbf{S}^n_V$, $A_j \in \mathbf{S}^n_V$.

At each iteration, Newton's method applied to (29) requires the solution of the set of linear equations

$$-P_V(X^{-1}\Delta X X^{-1}) + \sum_{k=1,\ldots,m} w_k A_k = C - P_V(X^{-1}) \tag{30}$$

$$\mathbf{tr}(A_j \Delta X) = 0, \quad j = 1, \ldots, m \tag{31}$$

(see [BV04, §10.2]). Here $X \in \mathbf{S}^n_{V,++}$ is the current iterate, and the variables are $w \in \mathbf{R}^m$ and the Newton step $\Delta X \in \mathbf{S}^n_V$. If m is not too large, we can solve the Newton equation efficiently by

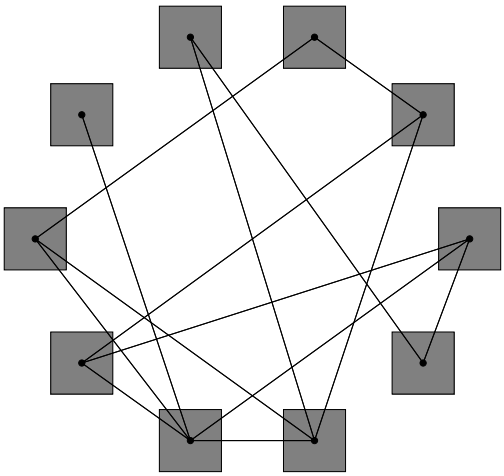

Figure 5: Construction of one of the test problems in §6.5. Each of the $N = 10$ blocks represents a complete subgraph of p nodes. The center nodes of the subgraphs are connected by a graph with 14 edges.

eliminating ΔX from (30). We first factor $X = RDR^T$ as in §3.2 and compute $P_V(X^{-1})$ as in §4.1. We then solve the linear equations

$$P_V(X^{-1}\Delta X_0 X^{-1}) = P_V(X^{-1}) - C, \qquad P_V(X^{-1}\Delta X_k X^{-1}) = A_k, \quad k = 1, \ldots, m,$$

with variables $\Delta X_0, \ldots, \Delta X_m$, using the algorithm of §4.2. This allows us to express ΔX in (30) as

$$\Delta X = \Delta X_0 + \sum_{k=1}^{m} w_k \Delta X_k. \tag{32}$$

Substituting this expression in the second equation (31) gives a dense set of linear equations

$$Hw = g \tag{33}$$

where $H \in \mathbf{S}^m$, $g \in \mathbf{R}^m$ are given by

$$H_{jk} = \mathbf{tr}(A_j \Delta X_k), \qquad g_j = -\mathbf{tr}(A_j X_0), \qquad j, k = 1, \ldots, m.$$

After solving for w we find ΔX from (32).

Note that in this method the matrix order n and the number of variables (nonzeros in V) can be quite large. For large sparse problems the complexity of the method is dominated by the cost of forming and solving the dense set of m linear equations in m variables (33).

Example We generate test graphs by connecting N complete subgraphs consisting of p nodes each. In each subgraph we select a center node, and connect the center nodes of the N subgraphs using a non-chordal graph. This is illustrated in figure 5. Table 2 shows the execution times of Newton's method for 9 problems. The number of variables listed is the total number of strictly upper triangular nonzeros; m is the number of nonzeros added in the chordal embedding. The number of Newton iterations in these experiments ranged from 10 to 18.

The algorithm was implemented in Python using the CVXOPT package [DV06]. The reported CPU time is in seconds on a 2GHz AMD64 PC.

N	p	#variables	m	CPU time
5	20	1055	5	0.3
5	50	6380	5	1.1
5	100	25255	5	4.9
10	20	2114	15	0.9
10	50	12764	15	5.4
10	100	50514	15	21.4
20	20	4265	81	10.2
20	50	25565	81	59.2
20	100	101065	81	210.0

Table 2: Execution time for Newton's method applied to a family of problems with nearly chordal sparsity patterns. The number of nodes in each problem is Np. The number of added links in the embedding is m.

6.4 Preconditioned conjugate gradient

The conjugate gradient method is a very useful method for general-purpose large scale unconstrained minimization. It does not require second derivative information, has minimal storage needs, and usually converges faster than steepest descent algorithms. It is therefore a popular alternative to Newton's method when the Newton equations are too expensive to solve.

It is well known that the practical success of the conjugate gradient method depends on the availability of a good preconditioner. In this section we formulate a preconditioner of the ML estimation problem (4). The preconditioner is based on a chordal embedding and exploits the results for chordal graphs in §4.

In the preconditioned conjugate gradient (PCG) algorithm we minimize $f(X) = -\log\det X + \mathbf{tr}(CX)$ over $\mathbf{S}^n_V$ by minimizing

$$g(Y) = f(\mathcal{H}(Y))$$

where $\mathcal{H} : \mathbf{S}^n_V \to \mathbf{S}^n_V$ is a linear mapping, selected so that the Hessian of g is expected to be better conditioned than the Hessian of f. It is also important that the mapping $\mathcal{H}$ and its adjoint are easy to evaluate.

To construct a suitable $\mathcal{H}$, we first note that if the sparsity pattern is chordal, then the factorization (23) provides a perfect preconditioner (at a given point): if we take $\mathcal{H} = \mathcal{A}^{-1} \circ \mathcal{C}^{-1}$, then, at $Y = \mathcal{H}^{-1}(X)$, we have

$$\nabla^2 g(Y) = \mathcal{H}_{\rm adj} \circ \nabla^2 f(X) \circ \mathcal{H} = I.$$

This suggests the following preconditioner for the ML problem (4) when the sparsity pattern V is not chordal. We take

$$\mathcal{H} = P_V \circ \mathcal{A}^{-1} \circ \mathcal{C}^{-1} \tag{34}$$

where $\mathcal{A}$ and $\mathcal{C}$ follow from factoring the Hessian matrix at the optimal solution after a chordal embedding. More precisely, we first compute a chordal embedding $\tilde{V}$ of V (see §5) and compute the solution $X^\star$ of the problem

$$\text{minimize} \quad \tilde{f}(X) = -\log\det X + \mathbf{tr}(CX) \tag{35}$$

n	#variables	#cliques	m	IPS	Newton	PCG
500	811	478	77	3.8+13.2	29.2	8.7
1000	1600	930	56	15.9+290.6	51.3	49.5
2000	3199	1801	94	61.4+1845.5	214.3	189.6
2000	2099	1945	2	62.8+3991.6	28.8	126.9

Table 3: Comparison of iterative proportional scaling, Newton's method, and the preconditioned conjugate gradient method, on four test probems.

where $\tilde{f} : \mathbf{S}^n_{\tilde{V}} \to \mathbf{R}$. This problem is easily solved, as described in §4.1, and the Hessian $\nabla^2 \tilde{f}(X^\star)$ can be factored as

$$\nabla^2 \tilde{f}(X^\star) = \mathcal{A}_{\rm adj} \circ \mathcal{C}_{\rm adj} \circ \mathcal{C} \circ \mathcal{A}$$

(see §4.2). This factorization is then used to construct the preconditioner (34).

6.5 Numerical results

We compare three algorithms: the iterative proportional scaling (IPS) algorithm (§6.2), Newton's method as implemented in §6.3, and the preconditioned conjugate gradient (PCG) algorithm with the preconditioner of §6.4. Table 3 shows the results for four test problems. The first three test problems use randomly generated sparsity patterns. The fourth example is a 2000-node subgraph of the WebBase graph [Dav]. The first four columns give some problem statistics: the matrix order, the number of upper triangular nonzeros, the number of cliques in the graph, and the number of nonzeros m added in the chordal embedding. The execution times for IPS are given as a sum of two terms: the time required to find the cliques in the graph, using the algorithm of [BK73] as implemented in the NetWorkX [Net] package, and the time required to run the IPS iteration. The algorithms were implemented in Python using the CVXOPT package [DV06]. The reported CPU times are in seconds on a 2GHz AMD64.

The results demonstrate the usefulness of a chordal embedding in reducing the complexity of Newton's algorithm. They also illustrate the effectiveness of the preconditioner of §6.4. This is further confirmed by the execution times for the conjugate gradient method without preconditioner (not included in the table), which were about 10 times slower than PCG.

7 Conclusion

We presented efficient algorithms for evaluating the gradient and Hessian of the function $\log\det X$ on the set of sparse positive definite matrices with a chordal sparsity pattern. The algorithms are formulated as recursions on an associated clique tree. As applications, we discussed the implementation of Newton's method and the preconditioned conjugate gradient algorithm for covariance selection problems with large nearly-chordal graphs.

References

[ADD96] P. Amestoy, T. Davis, and I. Duff. An approximate minimum degree ordering. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[BJL89] W. W. Barrett, C. R. Johnson, and M. Lundquist. Determinantal formulation for matrix completions associated with chordal graphs. *Linear Algebra and Appl.*, 121:265–289, 1989.

[BK73] C. Bron and J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, 1973.

[Bol89] K. A. Bollen. *Structural Equations with Latent Variables*. John Wiley & Sons, 1989.

[BP93] J. R. S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. In A. George, J. R. Gilbert, and J. W. H. Liu, editors, *Graph Theory and Sparse Matrix Computation*. Springer-Verlag, 1993.

[BT97] D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, Mass., 1997.

[BV04] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004. `www.stanford.edu/~boyd/cvxbook`.

[CDLS99] G. R. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, 1999.

[Dav] T. Davis. University of florida sparse matrix collection. Available from `http://www.cise.ufl.edu/research/sparse/mat/Kamvar`.

[Dem72] A. P. Dempster. Covariance selection. *Biometrics*, 28:157–175, 1972.

[DV06] J. Dahl and L. Vandenberghe. *CVXOPT: A Python Package for Convex Optimization*. Available from `www.ee.ucla.edu/~vandenbe/cvxopt`, 2006.

[ET75] A. M. Erisman and W. F. Tinney. On computing certain elements of the inverse of a sparse matrix. *Communications of the ACM*, 18(3):177–179, 1975.

[FKMN00] M. Fukuda, M. Kojima, K. Murota, and K. Nakata. Exploiting sparsity in semidefinite programming via matrix completion I: general framework. *SIAM Journal on Optimization*, 11:647–674, 2000.

[GGMS74] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28:505–535, 1974.

[GJSW84] R. Grone, C. R. Johnson, E. M Sá, and H. Wolkowicz. Positive definite completions of partial Hermitian matrices. *Linear Algebra and Appl.*, 58:109–124, 1984.

[GL96] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, 3rd edition, 1996.

[JKW98] C. R. Johnson, B. Kroschel, and H. Wolkowicz. An interior-point method for approximate positive definite completions. *Comput. Optim. Appl.*, 9(2):175–190, 1998.

[Lau96] S. L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[Lau01] M. Laurent. Matrix completion problems. In C. A. Floudas and P. M. Pardalos, editors, *Encyclopedia of Optimization*, volume III, pages 221–229. Kluwer, 2001.

[Net] Networkx. Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. Available from `networkx.lanl.gov`.

[NFF+03] K. Nakata, K. Fujitsawa, M. Fukuda, M. Kojima, and K. Murota. Exploiting sparsity in semidefinite programming via matrix completion II: implementation and numerical details. *Mathematical Programming Series B*, 95:303–327, 2003.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

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

[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.

[SK86] T. P. Speed and H. T. Kiiveri. Gaussian Markov distribution over finite graphs. *The Annals of Statistics*, 14(1), 1986.

[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.

[VBW98] L. Vandenberghe, S. Boyd, and S.-P. Wu. Determinant maximization with linear matrix inequality constraints. *SIAM J. on Matrix Analysis and Applications*, 19(2):499–533, April 1998.

[Wer80] N. Wermuth. Linear recursive equations, covariance selection, and path analysis. *Journal of the American Statistical Association*, 75(372):963–972, 1980.

[WS77] N. Wermuth and E. Scheidt. Algorithm AS 105: Fitting a covariance selection model to a matrix. *Applied Statistics*, 26(1):88–92, 1977.