Large-Scale Sparse Inverse Covariance Estimation via Thresholding and Max-Det Matrix Completion[*]

Richard Y. Zhang[†] Salar Fattahi[†] Somayeh Sojoudi[‡]

June 6, 2018

Abstract

The sparse inverse covariance estimation problem is commonly solved using an ℓ_1-regularized Gaussian maximum likelihood estimator known as "graphical lasso", but its computational cost becomes prohibitive for large data sets. A recent line of results showed–under mild assumptions–that the graphical lasso estimator can be retrieved by soft-thresholding the sample covariance matrix and solving a maximum determinant matrix completion (MDMC) problem. This paper proves an extension of this result, and describes a Newton-CG algorithm to efficiently solve the MDMC problem. Assuming that the thresholded sample covariance matrix is sparse with a sparse Cholesky factorization, we prove that the algorithm converges to an ϵ-accurate solution in $O(n\log(1/\epsilon))$ time and $O(n)$ memory. The algorithm is highly efficient in practice: we solve the associated MDMC problems with as many as 200,000 variables to 7-9 digits of accuracy in less than an hour on a standard laptop computer running MATLAB.

1 Introduction

Consider the problem of estimating an $n\times n$ covariance matrix Σ (or its inverse Σ^{-1}) of a n-variate probability distribution from N independently and identically distributed samples $\mathbf{x}_1,\mathbf{x}_2,\dots,\mathbf{x}_N$ drawn from the same probability distribution. In applications spanning from computer vision, natural language processing, to economics [24, 25, 10], the matrix Σ^{-1} is often *sparse*, meaning that its matrix elements are mostly zero. For Gaussian distributions, the statistical interpretation of sparsity in Σ^{-1} is that most of the variables are pairwise conditionally independent [27, 45, 14, 5].

Imposing sparsity upon Σ^{-1} can regularize the associated estimation problem and greatly reduce the number of samples required. This is particularly important in high-dimensional settings where n is large, often significantly larger than the number of samples $N \ll n$. One popular approach regularizes the associated maximum likelihood estimation (MLE) problem by a sparsity-promoting ℓ_1 term, as in

$$\underset{X\succ 0}{\text{minimize}}\ \operatorname{tr} CX - \log\det X + \lambda\sum_{i=1}^{n}\sum_{j=1}^{n}|X_{i,j}|. \tag{1}$$

Here, $C=\frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i-\bar{\mathbf{x}})(\mathbf{x}_i-\bar{\mathbf{x}})^T$ is the sample covariance matrix with sample mean $\bar{\mathbf{x}}=\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$, and X is the resulting estimator for Σ^{-1}. This approach, commonly known as the *graphical lasso* [14], is

[*]Proceedings of the 35-th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s). This work was supported by the ONR grants N00014-17-1-2933 and N00014-15-1-2835, DARPA grant D16AP00002, and AFOSR grant FA9550-17-1-0163.

[†]Department of Industrial Engineering and Operations Research, University of California, Berkeley, CA USA. (ryz@berkeley.edu, fattahi@berkeley.edu)

[‡]Department of Electrical Engineering and Computer Sciences / Department of Mechanical Engineering, University of California, Berkeley, CA USA. (sojoudi@berkeley.edu)

known to enjoy a number of statistical guarantees [37, 34], some of which are direct extensions of earlier work on the classical lasso [31, 29, 42, 22]. A variation on this theme is to only impose the ℓ_1 penalty on the off-diagonal elements of X, or to place different weights λ on the elements of the matrix X, as in the classical weighted lasso.

While the ℓ_1-regularized problem (1) is technically convex, it is commonly considered intractable for large-scale datasets. The decision variable is an $n \times n$ matrix, so simply fitting all $O(n^2)$ variables into memory is already a significant issue. General-purpose algorithms have either prohibitively high complexity or slow convergence. In practice, (1) is solved using problem-specific algorithms. The state-of-the-art include GLASSO [14], QUIC [20], and its "big-data" extension BIG-QUIC [21]. These algorithms use between $O(n)$ and $O(n^3)$ time and between $O(n^2)$ and $O(n)$ memory per iteration, but the number of iterations needed to converge to an accurate solution can be very large.

1.1 Graphical lasso, soft-thresholding, and MDMC

The high practical cost of graphical lasso has inspired a number of heuristics, which enjoy less guarantees but are significantly cheaper to use. Indeed, heuristics are often the only viable option once n exceeds the order of a few tens of thousands.

One simple idea is to *threshold* the sample covariance matrix C: to examine all of its elements and keep only the ones whose magnitudes exceed some threshold. Thresholding can be fast—even for very large-scale datasets—because it is embarassingly parallel; its quadratic $O(n^2)$ total work can be spread over thousands or millions of parallel processors, in a GPU or distributed on cloud servers. When the number of samples N is small, i.e. $N \ll n$, thresholding can also be performed using $O(n)$ memory, by working directly with the $n \times N$ centered matrix-of-samples $\mathbf{X} = [\mathbf{x}_1 - \bar{\mathbf{x}}, \mathbf{x}_2 - \bar{\mathbf{x}}, \ldots, \mathbf{x}_N - \bar{\mathbf{x}}]$ satisfying $C = \frac{1}{N}\mathbf{X}\mathbf{X}^T$.

In a recent line of work [26, 39, 12, 13], the simple heuristic of thresholding was shown to enjoy some surprising guarantees. In particular, [39, 12] proved that when the lasso weight is imposed over only the off-diagonal elements of X that—under some assumptions—the *sparsity pattern* of the associated graphical lasso estimator can be recovered by performing a soft-thresholding operation on C, as in

$$(C_\lambda)_{i,j} = \begin{cases} C_{i,j} & i = j, \\ C_{i,j} - \lambda & C_{i,j} > \lambda,\ i \neq j, \\ 0 & |C_{i,j}| \leq \lambda\ i \neq j, \\ C_{i,j} + \lambda & -\lambda \leq C_{i,j}\ i \neq j, \end{cases} \tag{2}$$

and recovering

$$G = \{(i,j) \in \{1, \ldots, n\}^2 : (C_\lambda)_{i,j} \neq 0\}. \tag{3}$$

The associated graph (also denoted as G when there is no ambiguity) is obtained by viewing each nonzero element (i,j) in G as an edge between the i-th and j-th vertex in an undirected graph on n nodes. Moreover, they showed that the estimator X can be recovered by solving a version of (1) in which the sparsity pattern G is explicitly imposed, as in

$$\begin{aligned} &\underset{X \succ 0}{\text{minimize}}\ \operatorname{tr} C_\lambda X - \log\det X \\ &\text{subject to } X_{i,j} = 0 \quad \forall (i,j) \notin G. \end{aligned} \tag{4}$$

Recovering the exact value of X (and not just its sparsity pattern) is important because it provides a shrinkage MLE when the true MLE is ill-defined; for Gaussian fields, its nonzero values encode the partial correlations between variables. Problem (4) is named the *maximum determinant matrix completion* (MDMC) in the literature, for reasons explained below. The problem has a recursive closed-form solution whenever the graph of G is *acyclic* (i.e. a tree or forest) [12], or more generally, if it is *chordal* [13]. It is worth emphasizing that the closed-form solution is *extremely* fast to evaluate: a chordal example in [13] with 13,659 variables took just ≈ 5 seconds to solve on a laptop computer.

The assumptions needed for graphical lasso to be equivalent to thresolding are hard to check but relatively mild. Indeed, [12] proves that they are automatically satisfied whenever λ is sufficiently large relative to the sample covariance matrix. Their numerical study found "sufficiently large" to be a fairly loose criterion in practice, particularly in view of the fact that large values of λ are needed to induce a sufficiently sparse estimate of Σ^{-1}, e.g. with $\approx 10n$ nonzero elements.

However, the requirement for G to be chordal is very strong. Aside from trivial chordal graphs like trees and cliques, thresholding will produce a chordal graph with probability zero. When G is nonchordal, no closed-form solution exists, and one must resort to an iterative algorithm. The state-of-the-art for nonchordal MDMC is to embed the nonchordal graph within a chordal graph, and to solve the resulting problem as a semidefinite program using an interior-point method [8, 2].

1.2 Main results

The purpose of this paper is two-fold. First, we derive an extension of the guarantees derived by [26, 39, 12, 13] for a slightly more general version of the problem that we call *restricted* graphical lasso (RGL):

$$\hat{X} = \underset{X\succ 0}{\text{minimize}}\ \operatorname{tr} CX - \log\det X + \sum_{i=1}^{n}\sum_{j=i+1}^{n} \lambda_{i,j}|X_{i,j}| \qquad (5)$$
$$\text{subject to } X_{i,j} = 0 \qquad \forall (i,j) \notin H.$$

In other words, RGL is (1) penalized by a weighted lasso penalty $\lambda_{i,j}$ on the off-diagonals, and with an *a priori* sparsity pattern H imposed as an additional constraint. We use the sparsity pattern H to incorporate prior information on the structure of the graphical model. For example, if the sample covariance C is collected over a graph, such as a communication system or a social network, then far-away variables can be assumed as pairwise conditionally independent [33, 19, 7]. Including these neighborhood relationships into H can regularize the statistical problem, as well as reduce the numerical cost for a solution.

In Section 2, we describe a procedure to transform RGL (5) into MDMC (4), in the same style as prior results by [12, 13] for graphical lasso. More specifically, we soft-threshold the sample covariance C and then project this matrix onto the sparsity pattern H. We give conditions for the resulting sparsity pattern to be equivalent to the one obtained by solving (5). Furthermore, we prove that the resulting estimator X can be recovered by solving the same MDMC problem (4) with C_λ appropriately modified.

The second purpose is to describe an efficient algorithm to solve MCDC when the graph G is *nonchordal*, based on the chordal embedding approach of [8, 2, 4]. We embed G within a chordal $\tilde{G} \supset G$, to result in a convex optimization problem over $\mathbb{S}^n_{\tilde{G}}$, the space of real symmetric matrices with sparsity pattern $\tilde{G}$. This way, the constraint $X \in \mathbb{S}^n_{\tilde{G}}$ is *implicitly* imposed, meaning that we simply ignore the nonzero elements not in $\tilde{G}$. Next, we solve an optimization problem on $\mathbb{S}^n_{\tilde{G}}$ using a custom Newton-CG method[1]. The main idea is to use an inner conjugate gradients (CG) loop to solve the Newton subproblem of an outer Newton's method. The actual algorithm has a number of features designed to exploit problem structure, including the sparse chordal property of $\tilde{G}$, duality, and the ability for CG and Newton to converge superlinearly; these are outlined in Section 3.

Assuming that the chordal embedding is sparse with $|\tilde{G}| = O(n)$ nonzero elements, we prove in Section 3.4, that our algorithm converges to an ϵ-accurate solution of MDMC (4) in

$$O(n \cdot \log \epsilon^{-1} \cdot \log\log \epsilon^{-1}) \text{ time and } O(n) \text{ memory.} \qquad (6)$$

[1]The MATLAB source code for our solver can be found at `http://alum.mit.edu/www/ryz`

Most importantly, the algorithm is highly efficient in practice. In Section 4, we present computation results on a suite of test cases. Both synthetic and real-life graphs are considered. Using our approach, we solve sparse inverse covariance estimation problems containing as many as 200,000 variables, in less than an hour on a laptop computer.

1.3 Related Work

Graphical lasso with prior information. A number of approaches are available in the literature to introduce prior information to graphical lasso. The weighted version of graphical lasso mentioned before is an example, though RGL will generally be more efficient to solve due to a reduction in the number of variables. [11] introduced a class of graphical lasso in which the true graphical model is assumed to have Laplacian structure. This structure commonly appears in signal and image processing [28]. For the *a priori* graph-based correlation structure described above, [17] introduced a *pathway* graphical lasso method similar to RGL.

Algorithms for graphical lasso. Algorithms for graphical lasso are usually based on some mixture of Newton [32], proximal Newton [21, 20], iterative thresholding [35], and (block) coordinate descent [14, 40]. All of these suffer fundamentally from the need to keep track and act on all $O(n^2)$ elements in the matrix X decision variable. Even if the final solution matrix were sparse with $O(n)$ nonzeros, it is still possible for the algorithm to traverse through a "dense region" in which the iterate X must be fully dense. Thresholding heuristics have been proposed to address issue, but these may adversely affect the outer algorithm and prevent convergence. It is generally impossible to guarantee a figure lower than $O(n^2)$ time per-iteration, even if the solution contains only $O(n)$ nonzeros. Most of the algorithms mentioned above actually have worst-case per-iteration costs of $O(n^3)$.

Graphical lasso via thresholding. The elementary estimator for graphical models (EE-GM) [43] is another thresholding-based low-complexity method that is able to recover the actual graphical lasso estimator. Both EE-GM and our algorithm have a similar level of performance in practice, because both algorithm are bottlenecked by the initial thresholding step, which is a quadratic $O(n^2)$ time operation.

Algorithms for MDMC. Our algorithm is inspired by a line of results [8, 2, 4, 23] for minimizing the log-det penalty on chordal sparsity patterns, culminating in the CVXOPT package [3]. These are Newton algorithms that solve the Newton subproblem by explicitly forming and factoring the fully-dense Newton matrix. When $|\tilde{G}| = O(n)$, these algorithms cost $O(nm^2 + m^3)$ time and $O(m^2)$ memory per iteration, where m is the number of edges added to G to yield the chordal $\tilde{G}$. In practice, m is usually a factor of 0.1 to 20 times n, so these algorithms are cubic $O(n^3)$ time and $O(n^2)$ memory. Our algorithm solves the Newton subproblem iteratively using CG. We prove that CG requires just $O(n)$ time to compute the Newton direction to machine precision (see Section 3.4). In practice, CG converges much faster than its worst-case bound, because it is able to exploit eigenvalue clustering to achieve superlinear convergence.

Notations

Let $\mathbb{R}^n$ be the set of $n\times 1$ real vectors, and $\mathbb{S}^n$ be the set of $n\times n$ real symmetric matrices. (We denote $x \in \mathbb{R}^n$ using lower-case, $X \in \mathbb{S}^n$ using upper-case, and index the (i, j)-th element of X as $X_{i,j}$.) We endow $\mathbb{S}^n$ with the usual matrix inner product $X \bullet Y = \operatorname{tr} XY$ and Euclidean (i.e. Frobenius) norm $\|X\|_F^2 = X \bullet X$. Let $\mathbb{S}^n_+ \subset \mathbb{S}^n$ and $\mathbb{S}^n_{++} \subset \mathbb{S}^n_+$ be the associated set of positive semidefinite and positive definite matrices. We will frequently write $X \succeq 0$ to mean $X \in \mathbb{S}^n_+$ and write $X \succ 0$ to mean $X \in \mathbb{S}^n_{++}$. Given a sparsity pattern G, we define $\mathbb{S}^n_G \subseteq \mathbb{S}^n$ as the set of $n \times n$ real symmetric matrices with this sparsity pattern.

2 Restricted graphical lasso, soft-thresholding, and MDMC

Let $P_H(X)$ denote the projection operator from $\mathbb{S}^n$ onto $\mathbb{S}^n_H$, i.e. by setting all $X_{i,j} = 0$ if $(i,j) \notin H$. Let C_λ be the sample covariance matrix C individually soft-thresholded by $[\lambda_{i,j}]$, as in

$$(C_\lambda)_{i,j} = \begin{cases} C_{i,j} & i = j, \\ C_{i,j} - \lambda_{i,j} & C_{i,j} > \lambda_{i,j},\ i \neq j, \\ 0 & |C_{i,j}| \leq \lambda_{i,j}\ i \neq j, \\ C_{i,j} + \lambda_{i,j} & -\lambda_{i,j} \leq C_{i,j}\ i \neq j, \end{cases} \tag{7}$$

In this section, we state the conditions for $P_H(C_\lambda)$—the projection of the soft-thresholded matrix C_λ in (7) onto H—to have the same sparsity pattern as the RGL estimator $\hat{X}$ in (5). Furthermore, the estimator $\hat{X}$ can be explicitly recovered by solving the MDMC problem (4) while replacing $C_\lambda \leftarrow P_H(C_\lambda)$ and $G \leftarrow P_H(G)$. For brevity, all proofs and remarks are omitted; these can be found in the appendix.

Before we state the exact conditions, we begin by adopting the some definitions and notations from the literature.

Definition 1. [12] Given a matrix $M \in \mathbb{S}^n$, define $G_M = \{(i,j) : M_{i,j} \neq 0\}$ as its sparsity pattern. Then M is called **inverse-consistent** if there exists a matrix $N \in \mathbb{S}^n$ such that

$$M + N \succ 0 \tag{8a}$$

$$N = 0 \qquad \forall (i,j) \in G_M \tag{8b}$$

$$(M + N)^{-1} \in \mathbb{S}^n_{G_M} \tag{8c}$$

The matrix N is called an **inverse-consistent complement** of M and is denoted by $M^{(c)}$. Furthermore, M is called **sign-consistent** if for every $(i,j) \in G_M$, the (i,j)-th elements of M and $(M + M^{(c)})^{-1}$ have opposite signs.

Moreover, we take the usual matrix max-norm to exclude the diagonal, as in $\|M\|_{\max} = \max_{i \neq j} |M_{ij}|$, and adopt the $\beta(G, \alpha)$ function defined with respect to the sparsity pattern G and scalar $\alpha > 0$

$$\begin{aligned} \beta(G, \alpha) = \max_{M \succ 0} & \|M^{(c)}\|_{\max} \\ \text{s.t. } & M \in \mathbb{S}^n_G \text{ and } \|M\|_{\max} \leq \alpha \\ & M_{i,i} = 1 \quad \forall i \in \{1, \dots, n\} \\ & M \text{ is inverse-consistent.} \end{aligned}$$

We are now ready to state the conditions for soft-thresholding to be equivalent to RGL.

Theorem 2. *Define C_λ as in (7), define $C_H = P_H(C_\lambda)$ and let $G_H = \{(i,j) : (C_H)_{i,j} \neq 0\}$ be its sparsity pattern. If the normalized matrix $\tilde{C} = D^{-1/2} C_H D^{-1/2}$ where $D = \operatorname{diag}(C_H)$ satisfies the following conditions:*

1. *$\tilde{C}$ is positive definite,*
2. *$\tilde{C}$ is sign-consistent,*
3. *We have*

$$\beta\left(G_H, \|\tilde{C}\|_{\max}\right) \leq \min_{(k,l) \notin G_H} \frac{\lambda_{k,l} - |(C_H)_{k,l}|}{\sqrt{(C_H)_{k,k} \cdot (C_H)_{l,l}}} \tag{9}$$

Then C_H has the same sparsity pattern and opposite signs as $\hat{X}$ in (5)*, i.e.*

$$\begin{aligned} (C_H)_{i,j} = 0 &\qquad \iff \qquad \hat{X}_{i,j} = 0, \\ (C_H)_{i,j} > 0 &\qquad \iff \qquad \hat{X}_{i,j} < 0, \\ (C_H)_{i,j} < 0 &\qquad \iff \qquad \hat{X}_{i,j} > 0. \end{aligned}$$

Proof. See Appendix A. □

Theorem 2 leads to the following corollary, which asserts that the optimal solution of RGL can be obtained by *maximum determinant matrix completion*: computing the matrix $Z \succeq 0$ with the largest determinant that "fills-in" the zero elements of $P_H(C_\lambda)$.

Corollary 3. *Suppose that the conditions in Theorem 2 are satisfied. Define $\hat{Z}$ as the solution to the following*

$$\begin{aligned} \hat{Z} = \underset{Z \succeq 0}{\textit{maximize}}\ & \log\det Z \\ \textit{subject to } & Z_{i,j} = P_H(C_\lambda) \textit{ for all } (i,j) \\ & \qquad \textit{where } [P_H(C_\lambda)]_{i,j} \neq 0 \end{aligned} \tag{10}$$

Then $\hat{Z} = \hat{X}^{-1}$, where $\hat{X}$ is the solution of (5).

Proof. Under the conditions of Theorem 2, the (i,j)-th element of the solution $\hat{X}_{i,j}$ has *opposite signs* to the corresponding element in $(C_H)_{i,j}$, and hence also $C_{i,j}$. Replacing each $|X_{i,j}|$ term in (5) with $\operatorname{sign}(\hat{X}_{i,j})X_{i,j} = -\operatorname{sign}(C_{i,j})X_{i,j}$ yields

$$\begin{aligned} \hat{X} = \underset{X \succ 0}{\text{minimize}}\ & \underbrace{\operatorname{tr} CX - \sum_{i=1}^{n}\sum_{j=i+1}^{n} \operatorname{sign}(C_{i,j})\lambda_{i,j}X_{i,j}}_{\equiv \operatorname{tr} C_\lambda X} - \log\det X \\ \text{subject to } & X_{i,j} = 0 \qquad \forall (i,j) \notin H. \end{aligned} \tag{11}$$

The constraint $X \in \mathbb{S}^n_H$ further makes $\operatorname{tr} C_\lambda X = \operatorname{tr} C_\lambda P_H(X) = \operatorname{tr} P_H(C_\lambda)X \equiv \operatorname{tr} C_H X$. Taking the dual of (11) yields (10); complementary slackness yields $\hat{Z} = \hat{X}^{-1}$. □

Standard manipulations show that (10) is the Lagrangian dual of (4), thus explaining the etymology of (4) as MDMC.

3 Proposed Algorithm

This section describes an efficient algorithm to solve MDMC (4) in which the sparsity pattern G is *non-chordal*. If we assume that the input matrix C_λ is sparse, and that sparse Cholesky factorization is able to solve $C_\lambda x = b$ in $O(n)$ time, then our algorithm is guaranteed to compute an ϵ-accurate solution in $O(n \log \epsilon^{-1})$ time and $O(n)$ memory.

The algorithm is fundamentally a Newton-CG method, i.e. Newton's method in which the Newton search directions are computed using conjugate gradients (CG). It is developed from four key insights:

1. Chordal embedding is easy via sparse matrix heuristics. State-of-the-art algorithms for (4) begin by computing a chordal embedding $\tilde{G}$ for G. The optimal chordal embedding with the fewest number of nonzeros $|\tilde{G}|$ is NP-hard to compute, but a good-enough embedding with $O(n)$ nonzeros is sufficient for our purposes. Computing a good $\tilde{G}$ with $|\tilde{G}| = O(n)$ is exactly the same problem as finding a sparse Cholesky factorization $C_\lambda = LL^T$ with $O(n)$ fill-in. Using heuristics developed for numerical linear algebra, we are able to find sparse chordal embeddings for graphs containing millions of edges and hundreds of thousands of nodes in seconds.

2. Optimize directly on the sparse matrix cone. Using log-det barriers for sparse matrix cones [8, 2, 4, 41], we can optimize directly in the space $\mathbb{S}^n_{\tilde{G}}$, while ignoring all matrix elements outside of $\tilde{G}$. If $|\tilde{G}| = O(n)$, then only $O(n)$ decision variables must be explicitly optimized. Moreover, each function evaluation, gradient evaluation, and matrix-vector product with the Hessian can be performed in $O(n)$ time, using the numerical recipes in [4].

3. The dual is easier to solve than the primal. The primal problem starts with a feasible point $X \in \mathbb{S}^n_{\tilde{G}}$ and seeks to achieve first-order optimality. The dual problem starts with an infeasible optimal point $X \notin \mathbb{S}^n_{\tilde{G}}$ satisfying first-order optimality, and seeks to make it feasible. Feasibility is easier to achieve than optimality, so the dual problem is easier to solve than the primal.

4. Conjugate gradients (CG) converges in $O(1)$ iterations. Under the same conditions that allow Theorem 2 to work, our main result (Theorem 6) bounds the condition number of the Newton subproblem to be $O(1)$, independent of the problem dimension n and the current accuracy ϵ. It is therefore cheaper to solve this subproblem using CG *to machine precision* δ_{mach} in $O(n \log \delta_{\text{mach}}^{-1})$ time than it is to solve for it directly in $O(nm^2 + m^3)$ time using Cholesky factorization [8, 2, 4]. Moreover, CG is an optimal Krylov subspace method, and as such, it is often able to exploit clustering in the eigenvalues to converge superlinearly. Finally, computing the Newton direction to high accuracy further allows the outer Newton method to also converge quadratically.

The remainder of this section describes each consideration in further detail. We state the algorithm explicitly in Section 3.5.

3.1 Efficient chordal embedding

Following [8], we begin by reformulating (4) into a sparse chordal matrix program

$$\begin{aligned} \hat{X} = \ & \text{minimize } \operatorname{tr} CX - \log\det X \\ & \text{subject to } X_{i,j} = 0 \quad \forall (i,j) \in \tilde{G}\backslash G. \\ & \qquad\qquad X \in \mathbb{S}^n_{\tilde{G}}. \end{aligned} \tag{12}$$

in which $\tilde{G}$ is a *chordal embedding* for G: a sparsity pattern $\tilde{G} \supset G$ whose graph contains no induced cycles greater than three. This can be implemented using standard algorithms for large-and-sparse linear equations, due to the following result.

Proposition 4. *Let $C \in \mathbb{S}^n_G$ be a positive definite matrix with sparsity pattern G. Compute its unique lower-triangular Cholesky factor L satisfying $C = LL^T$. Ignoring perfect numerical cancellation, the sparsity pattern of $L + L^T$ is a chordal embedding $\tilde{G} \supset G$.*

Proof. The original proof is due to [36]; see also [41]. □

Note that $\tilde{G}$ can be determined directly from G using a *symbolic* Cholesky algorithm, which simulates the steps of Gaussian elimination using Boolean logic. Moreover, we can substantially reduce the number of edges added to G by reordering the columns and rows of C using a *fill-reducing ordering*.

Corollary 5. *Let Π be a permutation matrix. For the same $C \in \mathbb{S}^n_G$ in Proposition 4, compute the unique Cholesky factor satisfying $\Pi C \Pi^T = LL^T$. Ignoring perfect numerical cancellation, the sparsity pattern of $\Pi(L + L^T)\Pi^T$ is a chordal embedding $\tilde{G} \supset G$.*

```
p = amd(C); % fill-reducing ordering
[~,~,~,~,R] = symbfact(C(p,p)); % chordal embedding by elimination
Gt = R+R'; Gt(p,p) = Gt; % recover embedded pattern
m = nnz(R)-nnz(tril(C)); % count the number of added eges
```

Figure 1: MATLAB code for chordal embedding via its internal approximate minimum degree ordering. Given a sparse matrix (C), compute a chordal embedding (Gt) and the number of added edges (m).

The problem of finding the best choice of Π is known as the *fill-minimizing* problem, and is NP-complete [44]. However, good orderings are easily found using heuristics developed for numerical linear algebra, like minimum degree ordering [15] and nested dissection [16, 1]. In fact, [16] proved that nested dissection is $O(\log(n))$ suboptimal for bounded-degree graphs, and notes that "we do not know a class of graphs for which [nested dissection is suboptimal] by more than a constant factor."

If G admits sparse chordal embeddings, then a good-enough $|\tilde{G}| = O(n)$ will usually be found using minimum degree or nested dissection. In MATLAB, the minimum degree ordering and symbolic factorization steps can be performed in two lines of code; see the snippet in Figure 1.

3.2 Logarithmic barriers for sparse matrix cones

Define the cone of *sparse positive semidefinite matrices* $\mathcal{K}$, and the cone of *sparse matrices with positive semidefinite completions* $\mathcal{K}_*$, as the following

$$\mathcal{K} = \mathbb{S}_+^n \cap \mathbb{S}_{\tilde{G}}^n, \qquad \mathcal{K}_* = \{S \bullet X \ge 0 : S \in \mathbb{S}_{\tilde{G}}\} = P_{\tilde{G}}(\mathbb{S}_+^n).$$

Then (12) can be posed as the primal-dual pair:

$$\arg\min_{X\in\mathcal{K}} \{C \bullet X + f(X) : A^T(X) = 0\}, \tag{13}$$

$$\arg\max_{S\in\mathcal{K}_*, y\in\mathbb{R}^m} \{-f_*(S) : S = C - A(y)\}, \tag{14}$$

with in which f and f_* are the "log-det" barrier functions on $\mathcal{K}$ and $\mathcal{K}_*$ as introduced by [8, 2, 4]

$$f(X) = -\log\det X, \qquad f_*(S) = -\min_{X\in\mathcal{K}}\{S \bullet X - \log\det X\}.$$

The linear map $A : \mathbb{R}^m \to \mathbb{S}^n_{\tilde{G}\backslash G}$ converts a list of m variables into the corresponding matrix in $\tilde{G}\backslash G$. The gradients of f are simply the projections of their usual values onto $\mathbb{S}^n_{\tilde{G}}$, as in

$$\nabla f(X) = -P_{\tilde{G}}(X^{-1}), \qquad \nabla^2 f(X)[Y] = P_{\tilde{G}}(X^{-1}YX^{-1}).$$

Given any $S \in \mathcal{K}_*$ let $X \in \mathcal{K}$ be the unique matrix satisfying $P_{\tilde{G}}(X^{-1}) = S$. Then we have

$$f_*(S) = n + \log\det X, \qquad \nabla f_*(S) = -X, \qquad \nabla^2 f_*(S)[Y] = \nabla^2 f(X)^{-1}[Y].$$

Assuming that $\tilde{G}$ is *sparse* and *chordal*, all six operations can be efficiently evaluated in $O(n)$ time and $O(n)$ memory, using the numerical recipes described in [4].

3.3 Solving the dual problem

Our algorithm actually solves the dual problem (14), which can be rewritten as an unconstrained optimization problem

$$\hat{y} \equiv \arg\min_{y \in \mathbb{R}^m} g(y) \equiv f_*(C_\lambda - A(y)). \tag{15}$$

After the solution $\hat{y}$ is found, we can recover the optimal estimator for the primal problem via $\hat{X} = -\nabla f_*(C_\lambda - A(y))$. The dual problem (14) is easier to solve than the primal (13) because the origin $y = 0$ often lies very close to the solution $\hat{y}$. To see this, note that $y = 0$ produces a candidate estimator $\tilde{X} = -\nabla f_*(C_\lambda)$ that solves the *chordal* matrix completion problem

$$\tilde{X} = \arg\min\{\operatorname{tr} C_\lambda X - \log\det X : X \in \mathbb{S}^n_{\tilde{G}}\},$$

which is a relaxation of the nonchordal problem posed over $\mathbb{S}^n_G$. As observed by several previous authors [8], this relaxation is a high quality guess, and $\tilde{X}$ is often "almost feasible" for the original nonchordal problem posed over $\mathbb{S}^n_G$, as in $\tilde{X} \approx P_G(\tilde{X})$. Some simple algebra shows that the gradient ∇g evaluated at the origin has Euclidean norm $\|\nabla g(0)\| = \|\tilde{X} - P_G(\tilde{X})\|_F$, so if $\tilde{X} \approx P_G(\tilde{X})$ holds true, then the origin $y = 0$ is close to optimal. Starting from this point, we can expect Newton's method to rapidly converge at a quadratic rate.

3.4 CG converges in $O(1)$ iterations

The most computationally expensive part of Newton's method is the solution of the Newton direction Δy via the $m \times m$ system of equations

$$\nabla^2 g(y)\Delta y = -\nabla g(y). \tag{16}$$

The Hessian matrix $\nabla^2 g(y)$ is fully dense, but matrix-vector products are linear $O(n)$ time using the algorithms in Section 3.2. This insight motivates solving (16) using an iterative Krylov subspace method like conjugate gradients (CG), which is a *matrix-free* method that requires a single matrix-vector product with $\nabla^2 g(y)$ at each iteration [6]. Starting from the origin $p = 0$, the method converges to an ϵ-accurate search direction p satisfying

$$(p - \Delta y)^T \nabla^2 g(y)(p - \Delta y) \le \epsilon |\Delta y^T \nabla g(y)|$$

in at most

$$\left\lceil \sqrt{\kappa_g} \log(2/\epsilon) \right\rceil \text{ CG iterations,} \tag{17}$$

where $\kappa_g = \|\nabla^2 g(y)\| \|\nabla^2 g(y)^{-1}\|$ is the condition number of the Hessian matrix [18, 38]. In many important convex optimization problems, the condition number κ_g grows like $O(1/\epsilon)$ or $O(1/\epsilon^2)$ as the outer Newton iterates approach an ϵ-neighborhood of the true solution. As a consequence, Newton-CG methods typically require $O(1/\sqrt{\epsilon})$ or $O(1/\epsilon)$ CG iterations.

It is therefore surprising that we are able to bound κ_g globally for the MDMC problem. Below, we state our main result, which says that the condition number κ_g depends polynomially on the problem data and the quality of the initial point, but is *independent of the problem dimension n and the accuracy of the current iterate ϵ.*

Theorem 6. *At any y satisfying $g(y) \le g(y_0)$ and $\nabla g(y)^T(y - y_0) \le \phi_{\max}$, the condition number κ_g of the Hessian matrix $\nabla^2 g(y)$ is bound*

$$\kappa_g \le 4\left(1 + \frac{\phi_{\max}^2 \lambda_{\max}(X_0)}{\lambda_{\min}(\hat{X})}\right)^2 \tag{18}$$

where:

- $\phi_{\max} = g(y_0) - g(\hat{y})$ *is the suboptimality of the initial point,*
- $\mathbf{A} = [\operatorname{vec} A_1, \dots, \operatorname{vec} A_m]$ *is the vectorized version of the problem data,*
- $X_0 = -\nabla f_*(S_0)$ *and* $S_0 = C - A(y_0)$ *are the initial primal-dual pair,*
- $\hat{X} = -\nabla f_*(\hat{S})$ *and* $\hat{S} = C - A(\hat{y})$ *are the solution primal-dual pair.*

Proof. See Appendix B. □

Remark 7. Newton's method is a descent method, so its k-th iterate y_k trivially satisfies $g(y_k) \le g(y_0)$. Technically, the condition $\nabla g(y_k)^T(y_k - y_0) \le \phi_{\max}$ can be guaranteed by enclosing Newton's method within an outer auxillary path-following loop; see Section 4.3.5 of [30]. In practice, naive Newton's method will usually satisfy the condition on its own; see our numerical experiments in Section 4.

Applying Theorem 6 to (17) shows that CG solves each Newton subproblem to ϵ-accuracy in $O(\log \epsilon^{-1})$ iterations. Multiplying this figure by the $O(\log\log \epsilon^{-1})$ Newton steps to converge yields a *global* iteration bound of

$$O(\log \epsilon^{-1} \cdot \log\log \epsilon^{-1}) \approx O(1) \text{ CG iterations.}$$

Multiplying this figure by the $O(n)$ cost of each CG iteration proves the claimed time complexity in (6). In practice, CG typically converges much faster than this worst-case bound, due to its ability to exploit the clustering of eigenvalues in $\nabla^2 g(y)$; see [18, 38]. Moreover, accurate Newton directions are only needed to guarantee quadratic convergence close to the solution. During the initial Newton steps, we may loosen the error tolerance for CG for a significant speed-up. Inexact Newton steps can be used to obtain a speed-up of a factor of 2-3.

3.5 The full algorithm

To summarize, we begin by computing a chordal embedding $\tilde{G}$ for the sparsity pattern G of C_λ, using the code snippet in Figure 1. We use the embedding to reformulate (4) as (12), and solve the unconstrained problem $\hat{y} = \min_y g(y)$ defined in (15), using Newton's method

$$y_{k+1} = y_k + \alpha_k \Delta y_k, \qquad \Delta y_k \equiv -\nabla^2 g(y_k)^{-1} \nabla g(y_k)$$

starting at the origin $y_0 = 0$. The function value $g(y)$, gradient $\nabla g(y)$ and Hessian matrix-vector products are all evaluated using the numerical recipes described by [4].

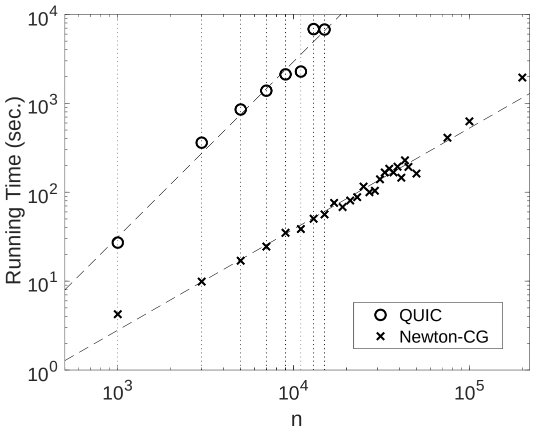
At each k-th Newton step, we compute the Newton search direction Δy_k using conjugate gradients. A loose tolerance is used when the Newton decrement $\delta_k = |\Delta y_k^T \nabla g(y_k)|$ is large, and a tight tolerance is used when the decrement is small, implying that the iterate is close to the true solution.

Once a Newton direction Δy_k is computed with a sufficiently large Newton decrement δ_k, we use a backtracking line search to determine the step-size α_k. In other words, we select the first instance of the sequence $\{1, \rho, \rho^2, \rho^3, \dots\}$ that satisfies the Armijo–Goldstein condition

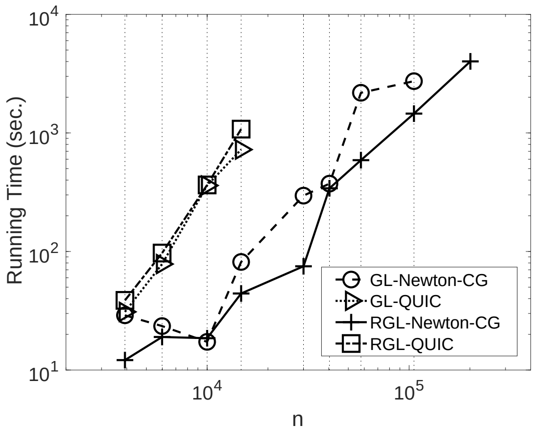
$$g(y + \alpha \Delta y) \le g(y) + \gamma \alpha \Delta y^T \nabla g(y),$$

in which $\gamma \in (0, 0.5)$ and $\rho \in (0, 1)$ are line search parameters. Our implementation used $\gamma = 0.01$ and $\rho = 0.5$. We complete the step and repeat the process, until convergence.

We terminate the outer Newton's method if the Newton decrement δ_k falls below a threshold. This implies either that the solution has been reached, or that CG is not converging to a good enough Δy_k to make significant progress. The associated estimator for Σ^{-1} is recovered by evaluating $\hat{X} = -\nabla f_*(C_\lambda - A(\hat{y}))$.



(a)



(b)

Figure 2: CPU time Newton-CG vs QUIC: (a) case study 1; (b) case study 2.

4 Numerical Results

Finally, we benchmark our algorithm[2] against `QUIC` [20], commonly considered the fastest solver for graphical lasso or RGL[3]. (Another widely-used algorithm is `GLASSO` [14], but we found it to be significantly slower than `QUIC`.) We consider two case studies. The first case study numerically verifies the claimed $O(n)$ complexity of our MDMC algorithm on problems with a nearly-banded structure. The second case study performs the full threshold-MDMC procedure for graphical lasso and RGL, on graphs collected from real-life applications.

All experiments are performed on a laptop computer with an Intel Core i7 quad-core 2.50 GHz CPU and 16GB RAM. The reported results are based on a serial implementation in MATLAB-R2017b. Both our Newton decrement threshold and `QUIC`'s convergence threshold are 10^{-7}.

We implemented the soft-thresholding set (7) as a serial routine that uses $O(n)$ memory by taking the $n \times N$ matrix-of-samples $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$ satisfying $C = \frac{1}{N}\mathbf{X}\mathbf{X}^T$ as input. The routine implicitly partitions C into submatrices of size 4000×4000, and iterates over the submatrices one at a time. For each submatrix, it explicitly forms the submatrix, thresholds it using dense linear algebra, and then stores the result as a sparse matrix.

4.1 Case Study 1: Banded Patterns

The first case study aims to verify the claimed $O(n)$ complexity of our algorithm for MDMC. Here, we avoid the proposed thresholding step, and focus solely on the MDMC (4) problem. Each sparsity pattern G is a corrupted banded matrices with bandwidth 101. The off-diagonal nonzero elements of C are selected from the uniform distribution in $[-2, 0)$ and then corrupted to zero with probability 0.3. The diagonal elements are fixed to 5. Our numerical experiments fix the bandwidth and vary the number of variables n from 1,000 to 200,000. A time limit of 2 hours is set for both algorithms.

Figure 2a compares the running time of both algorithms. A log-log regression results in an empirical time complexity of $O(n^{1.1})$ for our algorithm, and $O(n^2)$ for `QUIC`. The extra 0.1 in the exponent is most likely an artifact our MATLAB implementation. In either case, `QUIC`'s quadratic complexity limits it to

[2]The MATLAB source code for our solver can be found at `http://alum.mit.edu/www/ryz`

[3]`QUIC` was taken from `http://bigdata.ices.utexas.edu/software/1035/`

						Newton-CG			QUIC		
#	file name	type	n	m	m/n	sec	gap	feas	sec	diff. gap	speed-up
1	freeFlyingRobot-7	GL	3918	20196	5.15	28.9	5.7e-17	2.3e-7	31.0	3.9e-4	1.07
1	freeFlyingRobot-7	RGL	3918	20196	5.15	12.1	6.5e-17	2.9e-8	38.7	3.8e-5	3.20
2	freeFlyingRobot-14	GL	5985	27185	4.56	23.5	5.4e-17	1.1e-7	78.3	3.8e-4	3.33
2	freeFlyingRobot-14	RGL	5985	27185	4.56	19.0	6.0e-17	1.7e-8	97.0	3.8e-5	5.11
3	cryg10000	GL	10000	170113	17.0	17.3	5.9e-17	5.2e-9	360.3	1.5e-3	20.83
3	cryg10000	RGL	10000	170113	17.0	18.5	6.3e-17	1.0e-7	364.1	1.9e-5	19.68
4	epb1	GL	14734	264832	18.0	81.6	5.6e-17	4.3e-8	723.5	5.1e-4	8.86
4	epb1	RGL	14734	264832	18.0	44.2	6.2e-17	3.3e-8	1076.4	4.2e-4	24.35
5	bloweya	GL	30004	10001	0.33	295.8	5.6e-17	9.4e-9	$*$	$*$	$*$
5	bloweya	RGL	30004	10001	0.33	75.0	5.5e-17	3.6e-9	$*$	$*$	$*$
6	juba40k	GL	40337	18123	0.44	373.3	5.6e-17	2.6e-9	$*$	$*$	$*$
6	juba40k	RGL	40337	18123	0.44	341.1	5.9e-17	2.7e-7	$*$	$*$	$*$
7	bayer01	GL	57735	671293	11.6	2181.3	5.7e-17	5.2e-9	$*$	$*$	$*$
7	bayer01	RGL	57735	671293	11.6	589.1	6.4e-17	1.0e-7	$*$	$*$	$*$
8	hcircuit	GL	105676	58906	0.55	2732.6	5.8e-17	9.0e-9	$*$	$*$	$*$
8	hcircuit	RGL	105676	58906	0.55	1454.9	6.3e-17	7.3e-8	$*$	$*$	$*$
9	co2010	RGL	201062	1022633	5.08	4012.5	6.3e-17	4.6e-8	$*$	$*$	$*$

Table 1: Details of case study 2. Here, "n" is the size of the covariance matrix, "m" is the number of edges added to make its sparsity graph chordal, "sec" is the running time in seconds, "gap" is the optimality gap, "feas" is the feasibility the solution, "diff. gap" is the difference in duality gaps for the two different methods, and "speed-up" is the fact speed-up over QUIC achieved by our algorithm.

$n = 1.5 \times 10^4$. By contrast, our algorithm solves an instance with $n = 2 \times 10^5$ in less than 33 minutes. The resulting solutions are extremely accurate, with optimality and feasibility gaps of less than 10^{-16} and 10^{-7}, respectively.

4.2 Case Study 2: Real-Life Graphs

The second case study aims to benchmark the full thresholding-MDMC procedure for sparse inverse covariance estimation on real-life graphs. The actual graphs (i.e. the sparsity patterns) for Σ^{-1} are chosen from *SuiteSparse Matrix Collection* [9]—a publicly available dataset for large-and-sparse matrices collected from real-world applications. Our chosen graphs vary in size from $n = 3918$ to $n = 201062$, and are taken from applications in chemical processes, material science, graph problems, optimal control and model reduction, thermal processes and circuit simulations.

For each sparsity pattern G, we design a corresponding Σ^{-1} as follows. For each $(i, j) \in G$, we select $(\Sigma^{-1})_{i,j} = (\Sigma^{-1})_{j,i}$ from the uniform distribution in $[-1, 1]$, and then corrupt it to zero with probability 0.3. Then, we set each diagonal to $(\Sigma^{-1})_{i,i} = 1 + \sum_j |(\Sigma^{-1})_{i,j}|$. Using this Σ, we generate $N = 5000$ samples i.i.d. as $\mathbf{x}_1, \ldots, \mathbf{x}_N \sim \mathcal{N}(0, \Sigma)$. This results in a sample covariance matrix $C = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T$.

We solve graphical lasso and RGL with the C described above using our proposed soft-thresholding-MDMC algorithm and `QUIC`, in order to estimate Σ^{-1}. In the case of RGL, we assume that the graph G is known *a priori*, while noting that 30% of the elements of Σ^{-1} have been corrupted to zero. Our goal here is to discover the location of these corrupted elements. In all of our simulations, the threshold λ is set so that the number of nonzero elements in the the estimator is roughly the same as the ground truth. We limit both algorithms to 3 hours of CPU time.

Figure 2b compares the CPU time of both two algorithms for this case study; the specific details are provided in Table 1. A log-log regression results in an empirical time complexity of $O(n^{1.64})$ and $O(n^{1.55})$ for graphical lasso and RGL using our algorithm, and $O(n^{2.46})$ and $O(n^{2.52})$ for the same using `QUIC`. The exponents of our algorithm are ≥ 1 due to the initial soft-thresholding step, which is quadratic-time on a serial computer, but ≤ 2 because the overall procedure is dominated by the solution of the MDMC. Both algorithms solve graphs with $n \leq 1.5 \times 10^4$ within the allotted time limit, though our algorithm is 11 times faster on average. Only our algorithm is able to solve the estimation problem with $n \approx 2 \times 10^5$ in a little more than an hour.

To check whether thresholding-MDMC really does solve graphical lasso and RGL, we substitute the two sets of estimators back into their original problems (1) and (5). The corresponding objective values have a relative difference $\leq 4 \times 10^{-4}$, suggesting that both sets of estimators are about equally optimal. This observation verifies our claims in Theorem 2 and Corollary 3 that (1) and (5): thresholding-MDMC does indeed solve graphical lasso and RGL.

5 Conclusions

Graphical lasso is a widely-used approach for estimating a covariance matrix with a sparse inverse from limited samples. In this paper, we consider a slightly more general formulation called *restricted* graphical lasso (RGL), which additionally enforces a prior sparsity pattern to the estimation. We describe an efficient approach that substantially reduces the cost of solving RGL: 1) soft-thresholding the sample covariance matrix and projecting onto the prior pattern, to recover the estimator's sparsity pattern; and 2) solving a maximum determinant matrix completion (MDMC) problem, to recover the estimator's numerical values. The first step is quadratic $O(n^2)$ time and memory but embarrassingly parallelizable. If the resulting sparsity pattern is *sparse* and *chordal*, then under mild technical assumptions, the second step can be performed using the Newton-CG algorithm described in this paper in linear $O(n)$ time and memory. The algorithm is tested on both synthetic and real-life data, solving instances with as many as 200,000 variables to 7-9 digits of accuracy within an hour on a standard laptop computer.

References

[1] Ajit Agrawal, Philip Klein, and R Ravi. Cutting down on fill using nested dissection: Provably good elimination orderings. In *Graph Theory and Sparse Matrix Computation*, pages 31–55. Springer, 1993.

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

[3] Martin S Andersen, Joachim Dahl, and Lieven Vandenberghe. CVXOPT: A Python package for convex optimization. *Available at cvxopt. org*, 54, 2013.

[4] Martin S Andersen, Joachim Dahl, and Lieven Vandenberghe. Logarithmic barriers for sparse matrix cones. *Optimization Methods and Software*, 28(3):396–423, 2013.

[5] Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine learning research*, 9:485–516, 2008.

[6] Richard Barrett, Michael W Berry, Tony F Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk Van der Vorst. *Templates for the solution of linear systems: building blocks for iterative methods*, volume 43. Siam, 1994.

[7] D. Croft, G. O'Kelly, G. Wu, R. Haw, M. Gillespie, L. Matthews, M. Caudy, P. Garapati, G. Gopinath, B. Jassal, and S. Jupe. Reactome: a database of reactions, pathways and biological processes. *Nucleic acids research*, 39:691–697, 2010.

[8] Joachim Dahl, Lieven Vandenberghe, and Vwani Roychowdhury. Covariance selection for nonchordal graphs via chordal embedding. *Optimization Methods & Software*, 23(4):501–520, 2008.

[9] Timothy A. Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software (TOMS)*, 38(1):1, 2011.

[10] Steven N. Durlauf. Nonergodic economic growth. *The Review of Economic Studies*, 60(2):349–366, 1993.

[11] Hilmi E. Egilmez, Eduardo Pavez, and Antonio Ortega. Graph learning from data under laplacian and structural constraints. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):825–841, 2017.

[12] Salar Fattahi and Somayeh Sojoudi. Graphical lasso and thresholding: Equivalence and closed-form solutions. `https://arxiv.org/abs/1708.09479`, 2017.

[13] Salar Fattahi, Richard Y Zhang, and Somayeh Sojoudi. Sparse inverse covariance estimation for chordal structures. `https://arxiv.org/abs/1711.09131`, 2018.

[14] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

[15] Alan George and Joseph WH Liu. The evolution of the minimum degree ordering algorithm. *Siam review*, 31(1):1–19, 1989.

[16] John Russell Gilbert. Some nested dissection order is nearly optimal. *Information Processing Letters*, 26(6):325–328, 1988.

[17] Maxim Grechkin, Maryam Fazel, Daniela M. Witten, and Su-In Lee. Pathway graphical lasso. *AAAI*, pages 2617–2623, 2015.

[18] Anne Greenbaum. *Iterative methods for solving linear systems*, volume 17. Siam, 1997.

[19] Jean Honorio, Dimitris Samaras, Nikos Paragios, Rita Goldstein, and Luis E. Ortiz. Sparse and locally constant gaussian graphical models. *Advances in Neural Information Processing Systems*, pages 745–753, 2009.

[20] C. J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar. Quic: quadratic approximation for sparse inverse covariance estimation. *Journal of Machine Learning Research*, 15(1):2911–2947, 2014.

[21] Cho-Jui Hsieh, Mátyás A Sustik, Inderjit S Dhillon, Pradeep K Ravikumar, and Russell Poldrack. Big & quic: Sparse inverse covariance estimation for a million variables. In *Advances in neural information processing systems*, pages 3165–3173, 2013.

[22] Junzhou Huang and Tong Zhang. The benefit of group sparsity. *The Annals of Statistics*, 38(4):1978–2004, 2010.

[23] Jinchao Li, Martin S Andersen, and Lieven Vandenberghe. Inexact proximal newton methods for self-concordant functions. *Mathematical Methods of Operations Research*, 85(1):19–41, 2017.

[24] Stan Z. Li. Markov random field models in computer vision. *European conference on computer vision*, pages 351–370, 1994.

[25] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[26] Rahul Mazumder and Trevor Hastie. Exact covariance thresholding into connected components for large-scale graphical lasso. *Journal of Machine Learning Research*, 13:781–794, 2012.

[27] Nicolai Meinshausen and Peter Bühlmann. High-dimensional graphs and variable selection with the lasso. *The annals of statistics*, 1436–1462, 2006.

[28] Peyman Milanfar. A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE Signal Processing Magazine*, 30(1):106–128, 2013.

[29] Sahand Negahban and Martin J. Wainwright. Joint support recovery under high-dimensional scaling: Benefits and perils of $l_{1,\infty}$-regularization. *Proceedings of the 21st International Conference on Neural Information Processing Systems*, pages 1161–1168, 2008.

[30] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[31] Guillaume Obozinski, Martin J. Wainwright, and Michael I. Jordan. Union support recovery in high-dimensional multivariate regression. *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*, pages 21–26, 2008.

[32] Figen Oztoprak, Jorge Nocedal, Steven Rennie, and Peder A Olsen. Newton-like methods for sparse inverse covariance estimation. In *Advances in neural information processing systems*, pages 755–763, 2012.

[33] Dongjoo Park and Laurence R. Rilett. Forecasting freeway link travel times with a multilayer feedforward neural network. *Computer Aided Civil and Infrastructure Engineering*, 14(5):357–367, 1999.

[34] P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu. High-dimensional covariance estimation by minimizing l_1-penalized log-determinant divergence. *Electronic Journal of Statistics*, 5:935–980, 2011.

[35] Benjamin Rolfs, Bala Rajaratnam, Dominique Guillot, Ian Wong, and Arian Maleki. Iterative thresholding algorithm for sparse inverse covariance estimation. In *Advances in Neural Information Processing Systems*, pages 1574–1582, 2012.

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

[37] Adam J. Rothman, Peter J. Bickel, Elizaveta Levina, and Ji Zhu. Sparse permutation invariant covariance estimation. *Electronic Journal of Statistics*, 2:494–515, 2008.

[38] Yousef Saad. *Iterative methods for sparse linear systems*, volume 82. Siam, 2003.

[39] Somayeh Sojoudi. Equivalence of graphical lasso and thresholding for sparse graphs. *Journal of Machine Learning Research*, 17(115):1–21, 2016.

[40] Eran Treister and Javier S Turek. A block-coordinate descent approach for large-scale sparse inverse covariance estimation. In *Advances in neural information processing systems*, pages 927–935, 2014.

[41] Lieven Vandenberghe, Martin S Andersen, et al. Chordal graphs and semidefinite optimization. *Foundations and Trends® in Optimization*, 1(4):241–433, 2015.

[42] Martin J Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1-constrained quadratic programming (lasso). *IEEE transactions on information theory*, 55(5):2183–2202, 2009.

[43] Eunho Yang, Aurélie C. Lozano, and Pradeep K. Ravikumar. Elementary estimators for graphical models. *Advances in neural information processing systems*, pages 2159–2167, 2014.

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

[45] Ming Yuan and Yi Lin. Model selection and estimation in the Gaussian graphical model. *Biometrika*, pages 19–35, 2007.

A Restricted graphical Lasso and MDMC

Our aim is to elucidate the connection between the RGL and MDMC problem under the assumption that the regularization coefficients are chosen to be large, i.e., when a *sparse* solution for the RGL is sought. Recall that RGL is formulated as follows

$$\begin{aligned}
&\text{minimize } \operatorname{tr} CX - \log\det X + \sum_{(i,j)\in V} \lambda_{ij}|X_{i,j}| && \text{(19a)}\\
&\text{s.t. } \; X \in \mathbb{S}^n_V && \text{(19b)}\\
&\qquad X \succ 0 && \text{(19c)}
\end{aligned}$$

Now, consider the following modified soft-thresholded sample covariance matrix

$$(C_\lambda)_{i,j} = \begin{cases} C_{i,j} & i = j \\ 0 & i \neq j \text{ and } (i,j) \notin V \\ 0 & i \neq j \text{ and } (i,j) \in V \text{ and } -\lambda_{i,j} \leq, C_{i,j} \leq \lambda_{i,j} \\ C_{i,j} - \lambda_{i,j}\,\text{sign}(C_{i,j}) & i \neq j \text{ and } (i,j) \in V \text{ and } |C_{i,j}| > \lambda_{i,j} \end{cases} \tag{20}$$

Definition 8 ([12])**.** For a given symmetric matrix M, let V_M denote the minimal set such that $M \in \mathbb{S}^n_{V_M}$. M is called **inverse-consistent** if there exists a matrix N with zero diagonal such that

$$\begin{aligned}
&M + N \succ 0 && \text{(21a)}\\
&N_{i,j} = 0 \quad \forall (i,j) \notin V_M && \text{(21b)}\\
&(M+N)^{-1} \in \mathbb{S}^n_{V_M} && \text{(21c)}
\end{aligned}$$

The matrix N is called an **inverse-consistent complement** of M and is denoted by $M^{(c)}$. Furthermore, M is called **sign-consistent** if for every $(i,j) \in V_M$, the (i,j) entries of M and $(M + M^{(c)})^{-1}$ have opposite signs.

Definition 9 ([12])**.** Given a sparsity pattern V and a scalar α, define $\beta(V,\alpha)$ as the maximum of $\|M^{(c)}\|_{\max}$ over all inverse-consistent positive-definite matrices M with the diagonal entries all equal to 1 such that $M \in \mathbb{S}^n_V$ and $\|M\|_{\max} \leq \alpha$.

Without loss of generality, we make the following mild assumption.

Assumption 1. $\lambda_{i,j} \neq |C_{i,j}|$ *and* $\lambda_{i,j} > 0$ *for every* $(i,j) \in V$.

We are now ready to restate Theorem 2.

Theorem 2. *Define C_λ as in (7), define $C_H = P_H(C_\lambda)$ and let $G_H = \{(i,j) : (C_H)_{i,j} \neq 0\}$ be its sparsity pattern. If the normalized matrix $\tilde{C} = D^{-1/2} C_H D^{-1/2}$ where $D = \operatorname{diag}(C_H)$ satisfies the following conditions:*

1. *$\tilde{C}$ is positive definite,*
2. *$\tilde{C}$ is sign-consistent,*
3. *We have*
$$\beta\left(G_H, \|\tilde{C}\|_{\max}\right) \leq \min_{(k,l)\notin G_H} \frac{\lambda_{k,l} - |(C_H)_{k,l}|}{\sqrt{(C_H)_{k,k}\cdot(C_H)_{l,l}}} \tag{9}$$

Then C_H has the same sparsity pattern and opposite signs as $\hat{X}$ in (5)*, i.e.*

$$\begin{aligned}
(C_H)_{i,j} = 0 \quad &\iff \quad \hat{X}_{i,j} = 0,\\
(C_H)_{i,j} > 0 \quad &\iff \quad \hat{X}_{i,j} < 0,\\
(C_H)_{i,j} < 0 \quad &\iff \quad \hat{X}_{i,j} > 0.
\end{aligned}$$

First, note that the diagonal elements of $\tilde{C}_\lambda$ are 1 and its off-diagonal elements are between -1 and 1. A sparse solution for RGL requires large regularization coefficients. This leads to numerous zero elements in $\tilde{C}_\lambda$ and forces the magnitude of the nonzero elements to be small. This means that, in most instances, $\tilde{C}_\lambda$ is positive definite or even diagonally dominant. Certifying Condition (ii) is hard in general. However, [12] shows that this condition is automatically implied by Condition (i) when V_{C_λ} induces an acyclic structure. More generally, [39] shows that $\tilde{C}_\lambda$ is sign-consistent if $(\tilde{C}_\lambda + \tilde{C}_\lambda^{(c)})^{-1}$ is close to its first order Taylor expansion. This assumption holds in practice due to the fact that the magnitude of the off-diagonal elements of $\tilde{C}_\lambda + \tilde{C}_\lambda^{(c)}$ is small. Furthermore, [13] proves that this condition is necessary for the equivalence between the sparsity patterns of thresholding and GL when the regularization matrix is large enough. Finally, [12] shows that the left hand side of (9) is upper bounded by $c \times \|\tilde{C}_\lambda\|_{\max}^2$ for some $c > 0$ which only depends on V_{C_λ}. This implies that, when $\|\tilde{C}_\lambda\|_{\max}$ is small, or equivalently the regularization matrix is large, Condition (iii) is automatically satisfied.

A.1 Proofs

In this section, we present the technical proofs of our theorems. To prove Theorem 2, we need a number of lemmas. First, consider the RGL problem, with $\mathbb{S}_V^n = \mathbb{S}^n$. The first lemma offers optimality (KKT) conditions for the unique solution of this problem.

Lemma 10. *X^* is the optimal solution of RGL problem with $\mathbb{S}_V^n = \mathbb{S}^n$ if and only if it satisfies the following conditions for every $i, j \in \{1, 2, ..., n\}$:*

$$(X^*)^{-1}_{i,j} = C_{i,j} \qquad \text{if} \quad i = j \tag{22a}$$
$$(X^*)^{-1}_{i,j} = C_{i,j} + \lambda_{i,j} \times \operatorname{sign}(X^*_{i,j}) \qquad \text{if} \quad X^*_{i,j} \neq 0 \tag{22b}$$
$$C_{i,j} - \lambda_{i,j} \leq (X^*)^{-1}_{i,j} \leq \Sigma_{i,j} + \lambda_{i,j} \qquad \text{if} \quad X^*_{i,j} = 0 \tag{22c}$$

where $(X^*)^{-1}_{i,j}$ denotes the $(i,j)^{th}$ entry of $(X^*)^{-1}$.

Proof. The proof is straightforward and omitted for brevity. □

Now, consider the following optimization:

$$\min_{X\in\mathbb{S}_+^n} -\log\det(X)+\mathrm{trace}(\tilde{C}X)+\sum_{(i,j)\in V}\tilde{\lambda}_{i,j}|X_{i,j}|+2\max_k\{C_{k,k}\}\sum_{(i,j)\in V^{(c)}}|X_{i,j}| \tag{23}$$

where

$$\tilde{C}_{i,j}=\frac{C_{i,j}}{\sqrt{C_{i,i}\times C_{j,j}}}\quad \tilde{\lambda}_{i,j}=\frac{\lambda_{i,j}}{\sqrt{C_{i,i}\times C_{j,j}}} \tag{24}$$

Let $\tilde{X}$ denotes the optimal solution of (23). Furthermore, define D as a diagonal matrix with $D_{i,i}=C_{i,i}$ for every $i\in\{1,2,...,n\}$. The following lemma relates $\tilde{X}$ to X^*.

Lemma 11. *We have* $X^*=D^{-1/2}\times\tilde{X}\times D^{-1/2}$.

Proof. To prove this lemma, we define an intermediate optimization problem. Consider

$$\min_{X\in\mathbb{S}_+^n} f(X)=-\log\det(X)+\mathrm{trace}(\Sigma X)+\sum_{(i,j)\in V}\lambda_{i,j}|X_{i,j}|+2\max_k\{C_{kk}\}\sum_{(i,j)\in V^{(c)}}|X_{i,j}| \tag{25}$$

Denote $X^\sharp$ as the optimal solution for (25). First, we show that $X^\sharp=X^*$. Trivially, X^* is a feasible solution for (25) and hence $f(X^\sharp)\le f(X^*)$. Now, we prove that $X^\sharp$ is a feasible solution for (19). To this goal, we show that $X^\sharp_{ij}=0$ for every $(i,j)\in V^{(c)}$. By contradiction, suppose $X^\sharp_{ij}\neq 0$ for some $(i,j)\in V^{(c)}$. Note that, due to the positive definiteness of ${X^\sharp}^{-1}$, we have

$$(X^\sharp)^{-1}_{i,i}\times(X^\sharp)^{-1}_{j,j}-((X^\sharp)^{-1}_{i,j})^2>0 \tag{26}$$

Now, based on Lemma 10, one can write

$$(X^\sharp)^{-1}_{ij}=C_{i,j}+2\max_k\{C_{k,k}\}\times\mathrm{sign}(X^\sharp_{i,j}) \tag{27}$$

Considering the fact that $C\succeq 0$, we have $|C_{i,j}|\le\max_k\{C_{k,k}\}$. Together with (27), this implies that $|(X^\sharp)^{-1}_{i,j}|\ge\max_k\{C_{k,k}\}$. Furthermore, due to Lemma 10, one can write $(X^\sharp)^{-1}_{i,i}=C_{i,i}$ and $(X^\sharp)^{-1}_{j,j}=C_{jj}$. This leads to

$$(X^\sharp)^{-1}_{i,i}\times(X^\sharp)^{-1}_{j,j}-((X^\sharp)^{-1}_{i,j})^2=C_{i,i}\times C_{j,j}-(\max_k\{C_{k,k}))^2\le 0 \tag{28}$$

which contradicts with (26). Therefore, $X^\sharp$ is a feasible solution for (19). This implies that $f(X^\sharp)\ge f(X^*)$ and hence, $f(X^*)=f(X^\sharp)$. Due to the uniqueness of the solution of (25), we have $X^*=X^\sharp$. Now, note that (25) can be reformulated as

$$\min_{X\in\mathbb{S}_+^n} -\log\det(X)+\mathrm{trace}(\tilde{C}D^{1/2}XD^{1/2})+\sum_{(i,j)\in V}\lambda_{i,j}|X_{i,j}|+2\max_k\{C_{k,k}\}\sum_{(i,j)\in V^{(c)}}|X_{ij}| \tag{29}$$

Upon defining

$$\tilde{X}=D^{1/2}XD^{1/2} \tag{30}$$

and following some algebra, one can verify that 25 is equivalent to

$$\min_{\tilde{X}\in\mathbb{S}_+^n} -\log\det(\tilde{X})+\mathrm{trace}(\tilde{C}\tilde{X})+\sum_{(i,j)\in V}\tilde{\lambda}_{i,j}|\tilde{X}_{i,j}|+2\max_k\{\tilde{C}_{k,k}\}\sum_{(i,j)\in V^{(c)}}|\tilde{X}_{i,j}|+\log\det(D) \tag{31}$$

Dropping the constant term in (31) gives rise to the optimization (23). Therefore, $X^*=D^{-1/2}\times\tilde{X}\times D^{-1/2}$ holds in light of 30. This completes the proof. □

Now, we present the proof of Theorem 2.

Proof of Theorem 2: Note that, due to the definition of $\tilde{C}_\lambda$ and Lemma 11, $\tilde{C}_\lambda$ and $\tilde{X}$ have the same signed sparsity pattern as C_λ and X^*, respectively. Therefore, it suffices to show that the signed sparsity structures of $\tilde{C}_\lambda$ and $\tilde{X}$ are the same.

To verify this, we focus on the optimality conditions for optimization (23). Due to Condition (1-i), $\tilde{C}_\lambda$ is inverse-consistent and has a unique inverse-consistent complement, which is denoted by N. First, we will show that $(\tilde{C}_\lambda + N)^{-1}$ is the optimal solution of (23). For an arbitrary pair $(i,j) \in \{1, ..., n\}$, the KKT conditions, introduced in Lemma 10, imply that one of the following cases holds:

1) $i = j$: We have $(\tilde{C}_\lambda + N)_{i,j} = [\tilde{C}_\lambda]_{i,i} = \tilde{C}_{i,i}$.

2) $(i,j) \in V_{C_\lambda}$: In this case, we have

$$(\tilde{C}_\lambda + N)_{ij} = [\tilde{C}_\lambda]_{ij} = \tilde{C}_{ij} - \tilde{\lambda}_{ij} \times \text{sign}(\tilde{C}_{ij}) \tag{32}$$

Note that since $|\tilde{C}_{ij}| > \tilde{\lambda}_{ij}$, we have that $\text{sign}([\tilde{C}_\lambda]_{ij}) = \text{sign}(\tilde{C}_{ij})$. On the other hand, due to the sign-consistency of $\tilde{C}_\lambda$, we have $\text{sign}([\tilde{C}_\lambda]_{ij}) = -\text{sign}\left(\left((\tilde{C}_\lambda + N)^{-1}\right)_{ij}\right)$. This implies that

$$(\tilde{C}_\lambda + N)_{ij} = \tilde{C}_{ij} + \tilde{\lambda}_{ij} \times \text{sign}\left(\left((\tilde{C}_\lambda + N)^{-1}\right)_{ij}\right) \tag{33}$$

3) $(i,j) \notin V_{C_\lambda}$: One can verify that $(\tilde{C}_\lambda + N)_{ij} = N_{ij}$. Therefore, due to Condition (1-iii), we have

$$\begin{aligned} |(\tilde{C}_\lambda + N)_{ij}| &\leq \beta\left(V_{C_\lambda}, \|\tilde{C}_\lambda\|_{\max}\right) \\ &\leq \min_{(k,l)\in V^{(c)}} \frac{\lambda_{kl} - |C_{kl}|}{\sqrt{C_{kk} \times C_{ll}}} \\ &= \min_{(k,l)\in V^{(c)}} \tilde{\lambda}_{kl} - |\tilde{C}_{kl}| \end{aligned} \tag{34}$$

This leads to

$$|(\tilde{C}_\lambda + N)_{ij} - \tilde{C}_{ij}| \leq |(\tilde{C}_\lambda + N)_{ij}| + |\tilde{C}_{ij}| \leq \min_{(k,l)\in V^{(c)}} (\tilde{\lambda}_{kl} - |\tilde{C}_{kl}|) + |\tilde{C}_{ij}| \leq \tilde{\lambda}_{ij} \tag{35}$$

Therefore, it can be concluded that $(\tilde{C}_\lambda + N)^{-1}$ satisfies the KKT conditions for (23). On the other hand, note that $V_{(\tilde{C}_\lambda+N)^{-1}} = V_{\tilde{C}_\lambda}$ and the nonzero off-diagonal elements of $(\tilde{C}_\lambda + N)^{-1})$ and $\tilde{C}_\lambda$ have opposite signs. This concludes the proof. ☐

B Solving the Newton Subproblem in $O(1)$ CG Iterations

Let $\mathbb{S}^n$ be the set of $n \times n$ real symmetric matrices. Given a sparsity pattern V, we define $\mathbb{S}^n_V \subseteq \mathbb{S}^n$ as the set of $n \times n$ real symmetric matrices with this sparsity pattern. We consider the following minimization problem

$$\hat{y} \equiv \arg\min_{y\in\mathbb{R}^m} g(y) \equiv f_*(C - A(y)). \tag{36}$$

Here, the problem data $A : \mathbb{R}^m \to \mathbb{S}^n_F$ is an orthogonal basis for a sub-sparsity pattern $F \subset V$ that excludes the matrix C. In other words, the operator A satisfies

$$A(A^T(X)) = P_F(X) \qquad \forall X \in S^n_V, \qquad\qquad A^T(C) = 0.$$

The penalty function f_* is the convex conjugate of the "log-det" barrier function on $\mathbb{S}^n_V$:

$$f_*(S) = -\min_{X \in \mathbb{S}^n_V} \{S \bullet X - \log\det X\}$$

Assuming that V is *chordal*, the function $f_*(S)$, its gradient $\nabla f_*(S)$, and its Hessian matrix-vector product $\nabla^2 f_*(S)[Y]$ can all be evaluated in closed-form [8, 2, 4]; see also [41]. Furthermore, if the pattern is *sparse*, i.e. its number of elements in the pattern satisfy $|V| = O(n)$, then all of these operations can be performed to arbitrary accuracy in $O(n)$ time and memory.

It is standard to solve (36) using Newton's method. Starting from some initial point $y_0 \in \operatorname{dom} g$

$$y_{k+1} = y_k + \alpha_k \Delta y_k \qquad \Delta y_k \equiv -\nabla^2 g(y_k)^{-1} \nabla g(y_k),$$

in which the step-size α_k is determined by backtracking line search, selecting the first instance of the sequence $\{1, \rho, \rho^2, \rho^3, \dots\}$ that satisfies the Armijo–Goldstein condition

$$g(y + \alpha\Delta y) \le g(y) + \gamma\alpha\Delta y^T \nabla g(y),$$

in which $\gamma \in (0, 0.5)$ and $\rho \in (0, 1)$. The function f_* is strongly self-concordant, and g inherits this property from f_*. Accordingly, classical analysis shows that we require at most

$$\left\lceil \frac{g(y_0) - g(\hat{y})}{0.05\gamma\rho} + \log_2 \log_2(1/\epsilon) \right\rceil \approx O(1) \text{ Newton steps}$$

to an ϵ-optimal point satisfying $g(y_k) - g(\hat{y}) \le \epsilon$. The bound is very pessimistic, and in practice, no more than 20-30 Newton steps are ever needed for convergence.

The most computationally expensive of Newton's method is the solution of the Newton direction Δy via the $m \times m$ system of equations

$$\nabla^2 g(y_k) \Delta y = -\nabla g(y_k). \tag{37}$$

The main result in this section is a proof that the condition number of $\nabla^2 g(y)$ is independent of the problem dimension n.

Theorem 6. *At any y satisfying $g(y) \le g(y_0)$ and $\nabla g(y)^T (y - y_0) \le \phi_{\max}$, the condition number κ_g of the Hessian matrix $\nabla^2 g(y)$ is bound*

$$\kappa_g \le 4 \left(1 + \frac{\phi_{\max}^2 \lambda_{\max}(X_0)}{\lambda_{\min}(\hat{X})}\right)^2 \tag{18}$$

where:

- $\phi_{\max} = g(y_0) - g(\hat{y})$ *is the suboptimality of the initial point,*
- $\mathbf{A} = [\operatorname{vec} A_1, \dots, \operatorname{vec} A_m]$ *is the vectorized version of the problem data,*
- $X_0 = -\nabla f_*(S_0)$ *and* $S_0 = C - A(y_0)$ *are the initial primal-dual pair,*
- $\hat{X} = -\nabla f_*(\hat{S})$ *and* $\hat{S} = C - A(\hat{y})$ *are the solution primal-dual pair.*

As a consequence, each Newton direction can be computed in $O(\log \epsilon^{-1})$ iterations using conjugate gradients, over $O(\log\log \epsilon^{-1})$ total Newton steps. The overall minimization problem is solved to ϵ-accuracy in

$$O(\log \epsilon^{-1} \log\log \epsilon^{-1}) \approx O(1) \text{ CG iterations.}$$

The leading constant here is dependent polynomially on the problem data and the quality of the initial point, but *independent of the problem dimensions*.

B.1 Preliminaries

We endow $\mathbb{S}_V^n$ with the usual matrix inner product $X \bullet Y = \operatorname{tr} XY$ and associated Euclidean norm $\|X\|_F^2 = X \bullet X$. The projection

$$P_V : \mathbb{S}^n \to \mathbb{S}_V^n \qquad P_V(X) = \arg \min_{Y \in \mathbb{S}_V^n} \|X - Y\|_F^2$$

is the associated projection from $\mathbb{S}^n$ onto $\mathbb{S}_V^n$.

Let $\mathbb{S}_+^n \subset \mathbb{S}^n$ be the set of positive semidefinite matrices. (We will frequently write $X \succeq 0$ to mean $X \in \mathbb{S}_+^n$.) We define the cone of *sparse positive semidefinite matrices*,

$$\mathcal{K} = \mathbb{S}_+^n \cap \mathbb{S}_V^n$$

and its dual cone, the cone of *sparse matrices with positive semidefinite completions,*

$$\mathcal{K}_* = \{S \bullet X \geq 0 : S \in \mathbb{S}_V\} = P_V(\mathbb{S}_+^n).$$

Being dual cones, $\mathcal{K}$ and $\mathcal{K}_*$ satisfy Farkas' lemma.

Lemma 12 (Farkas' lemma)**.** *Given an arbitrary* $Y \in \mathbb{S}_V^n$

1. *Either* $Y \in \mathcal{K}$*, or there exists a separating hyperplane* $S \in \mathcal{K}^*$ *such that* $S \bullet Y < 0$*.*
2. *Either* $Y \in \mathcal{K}_*$*, or there exists a separating hyperplane* $X \in \mathcal{K}$ *such that* $Y \bullet X < 0$*.*

The following barrier functions on $\mathcal{K}$ and $\mathcal{K}_*$ were introduced by Dahl, Andersen and Vandenberghe [8, 4, 2]:

$$f(X) = -\log \det X, \qquad f_*(S) = -\min_{X \in \mathbb{S}_V^n} \{S \bullet X - \log \det X\}.$$

The gradients of f are simply the projections of their usual values onto $\mathbb{S}_V^n$, as in

$$\nabla f(X) = -P_V(X^{-1}), \qquad \nabla^2 f(X)[Y] = P_V(X^{-1} Y X^{-1}),$$

Then we also have for any $S \in \mathcal{K}_*$

$$f_*(S) = n + \log \det X, \qquad \nabla f(S) = -X, \qquad \nabla^2 f(S)[Y] = \nabla^2 f(X)^{-1}[Y],$$

where $X \in \mathcal{K}$ is the unique matrix satisfying $P_V(X^{-1}) = S$. Dahl, Andersen and Vandenberghe [8, 4, 2] showed that all six operations defined above can be efficiently evaluated in $O(n)$ time on a *sparse* and *chordal* sparsity pattern V.

From the above, we see that the Hessian matrix $\nabla^2 g(y)$ can be written it terms of the Hessian $\nabla^2 f(X)$ and the unique $X \in \mathcal{K}$ satisfying $P_V(X^{-1}) = C - A(y)$, as in

$$\nabla^2 g(y) = A^T(\nabla^2 f(X)^{-1}[A(y)]) = \mathbf{A}^T \nabla^2 f(X)^{-1} \mathbf{A},$$

in which $\mathbf{A} = [\operatorname{vec} A_1, \ldots, \operatorname{vec} A_m]$. Moreover, using the theory of Kronecker products, we can write

$$\operatorname{vec} \nabla^2 f(X)[Y] = Q^T (X^{-1} \otimes X^{-1}) Q \operatorname{vec} Y$$

in which the $\frac{1}{2}n(n+1) \times |V|$ matrix Q is the orthogonal basis matrix of $\mathbb{S}_V^n$ in $\mathbb{S}^n$. Because of this, we see that the eigenvalues of the Hessian $\nabla^2 g(y)$ are bound

$$\lambda_{\min}(\mathbf{A}^T\mathbf{A}) \lambda_{\min}^2(X^{-1}) \leq \lambda_i(\nabla^2 g(y)) \leq \lambda_{\max}(\mathbf{A}^T\mathbf{A}) \lambda_{\max}^2(X^{-1}),$$

and therefore its condition number is bound by the eigenvalues of X

$$\operatorname{cond}(\nabla^2 g(y)) \leq \operatorname{cond}(\mathbf{A}^T\mathbf{A}) \left(\frac{\lambda_{\max}(X)}{\lambda_{\min}(X)}\right)^2 = \left(\frac{\lambda_{\max}(X)}{\lambda_{\min}(X)}\right)^2. \tag{38}$$

The second equality arises here because A is orthogonal by construction; given that it is defined to satisfy $A(A^T(X)) = P_F(X)$ for some sparsity pattern $F \subset V$, we must have $A^T(A(y)) = y$ for all $y \in \mathbb{R}^m$. Consequently most of our effort will be expended in bounding the eigenvalues of X.

B.2 Proof of Theorem 6

To simplify notation, we will write y_0, $\hat{y}$, and y as the initial point, the solution, and any k-th iterate. From this, we define S_0, $\hat{S}$, S, with each satisfying $S = C - A(y)$, and X_0, $\hat{X}$, and X, the points in $\mathcal{K}$ each satisfying $P_V(X^{-1}) = S$.

Our goal is to bound the extremal eigenvalues of X. To do this, we first recall that the sequence is monotonously decreasing by hypothesis, as in

$$g(\hat{y}) \le g(y) \le g(y_0).$$

Evaluating each $f_*(S)$ as $n + \log\det X$, yields

$$\log\det\hat{X} \le \log\det X \le \log\det X_0. \tag{39}$$

Next, we introduce the following function, which often appears in the study of interior-point methods

$$\phi(M) = \operatorname{tr} M - \log\det M - n \ge 0,$$

and is well-known to provide a control on the arthmetic and geometric means of the eigenvalues of M. Indeed, the function attains its unique minimum at $\phi(I) = 0$, and it is nonnegative precisely because of the arithmetic-geometric inequality. Let us show that it can also bound the arithmetic-geometric means of the extremal eigenvalues of M.

Lemma 13. *Denote the n eigevalues of M as $\lambda_1 \ge \cdots \ge \lambda_n$. Then*

$$\phi(M) \ge \lambda_1 + \lambda_n - 2\sqrt{\lambda_1\lambda_n} = (\sqrt{\lambda_1} - \sqrt{\lambda_n})^2.$$

Proof. Noting that $x - \log x - 1 \ge 0$ for all $x \ge 0$, we have

$$\begin{aligned}
\phi(M) &= \sum_{i=1}^{n} (\lambda_i - \log\lambda_i - 1) \\
&\ge (\lambda_1 - \log\lambda_1 - 1) + (\lambda_n - \log\lambda_n - 1) \\
&= \lambda_1 + \lambda_n - 2\log\sqrt{\lambda_1\lambda_n} - 2 \\
&= \lambda_1 + \lambda_n - 2\sqrt{\lambda_1\lambda_n} + 2(\sqrt{\lambda_1\lambda_n} - \log\sqrt{\lambda_1\lambda_n} - 1) \\
&\ge \lambda_1 + \lambda_n - 2\sqrt{\lambda_1\lambda_n}.
\end{aligned}$$

Completing the square yields $\phi(M) \ge (\sqrt{\lambda_1} - \sqrt{\lambda_n})^2$. □

The following upper-bounds are the specific to our problem, and are the key to our intended final claim.

Lemma 14. *Define the initial suboptimality $\phi_{\max} = \log\det X_0 - \log\det\hat{X}$. Let $\nabla g(y)^T(y - y_0) \le \phi_{\max}$. Then we have*

$$\phi(\hat{X}X^{-1}) \le \phi_{\max}, \qquad \phi(XX_0^{-1}) \le 2\phi_{\max}.$$

Proof. To prove the first inequality, we take first-order optimality at the optimal point $\hat{y}$

$$\nabla g(\hat{y}) = A^T(\hat{X}) = 0.$$

Noting that $\hat{X} \in \mathbb{S}_V^n$, we further have

$$\begin{aligned}
X^{-1} \bullet \hat{X} = P_V(X^{-1}) \bullet \hat{X} = [C - A(y)] \bullet \hat{X} &= C \bullet \hat{X} - y^T \underbrace{A^T(\hat{X})}_{=0} \\
&= [P_V(\hat{X}^{-1}) + A(\hat{y})] \bullet \hat{X} \\
&= P_V(\hat{X}^{-1}) \bullet \hat{X} = n
\end{aligned}$$

and hence $\phi(X^{-1}\hat{X})$ has the value of the suboptimality at X, which is bound by the initial suboptimality in (39):

$$\begin{aligned}\phi(X^{-1}\hat{X}) &= \underbrace{X^{-1}\bullet\hat{X}}_{n} - \log\det X^{-1}\hat{X} - n\\ &= \log\det X - \log\det\hat{X}\\ &\leq \log\det X_0 - \log\det\hat{X} = \phi_{\max}.\end{aligned}$$

We begin with the same steps to prove the second inequality:

$$\begin{aligned}X_0^{-1}\bullet X = P_V(X_0^{-1})\bullet X &= [C - A(y_0)]\bullet X\\ &= [P_V(X^{-1}) + A(\hat{y})]\bullet X - A(y_0)\bullet X\\ &= n + A(y-y_0)\bullet X.\end{aligned}$$

Note that $\nabla g(y)^T(y-y_0) = (y-y_0)^T A^T(X) = A(y-y_0)\bullet X \leq \phi_{\max}$. Substituting and applying (39) yields

$$\begin{aligned}\phi(X_0^{-1}X) &= X_0^{-1}\bullet X - \log\det X_0^{-1}X - n\\ &= (n + A(y-y_0)\bullet X - \log\det X_0^{-1}X) - n\\ &= \underbrace{\log\det X_0X^{-1}}_{\leq\phi_{\max}} + \underbrace{A(y-y_0)\bullet X}_{\leq\phi_{\max}}.\end{aligned}$$

□

Using the two upper-bounds to bound the eigenvalues of their arguments is enough to derive a condition number bound on X, which immediately translates into a condition number bound on $\nabla^2 g(y)$.

Proof of Theorem 6. To prove the first bound (18), we will instead prove

$$\frac{\lambda_{\max}(X)}{\lambda_{\min}(X)} \leq 2 + \frac{2\phi_{\max}^2\lambda_{\max}(X_0)}{\lambda_{\min}(\hat{X})}, \tag{40}$$

which yields the desired condition number bound on $\nabla^2 g(y)$ by substituting into (38). Writing $\lambda_1 = \lambda_{\max}(X)$ and $\lambda_n = \lambda_{\min}(X)$, we have from the two lemmas above:

$$\begin{aligned}\phi_{\max} &\geq \lambda_{\min}(\hat{X})(\sqrt{\lambda_n^{-1}} - \sqrt{\lambda_1^{-1}})^2 > 0,\\ 2\phi_{\max} &\geq \lambda_{\min}(X_0^{-1})(\sqrt{\lambda_1} - \sqrt{\lambda_n})^2 > 0.\end{aligned}$$

Multiplying the two upper-bounds and substituing $\lambda_{\min}(X_0^{-1}) = 1/\lambda_{\max}(X_0)$ yields

$$\frac{2\phi_{\max}^2\lambda_{\max}(X_0)}{\lambda_{\min}(\hat{X})} \geq \left(\sqrt{\frac{\lambda_1}{\lambda_n}} - \sqrt{\frac{\lambda_n}{\lambda_1}}\right)^2 = \frac{\lambda_1}{\lambda_n} + \frac{\lambda_n}{\lambda_1} - 2.$$

Finally, bounding $\lambda_n/\lambda_1 \geq 0$ yields (40). □