Linear-Time Algorithm for Learning Large-Scale Sparse Graphical Models

Salar Fattahi, Richard Y. Zhang and Somayeh Sojoudi

***Abstract*—We consider the graphical lasso, a popular optimization problem for learning the sparse representations of high-dimensional datasets, which is well-known to be computationally expensive for large-scale problems. A recent line of results has shown–under mild assumptions–that the sparsity pattern of the graphical lasso estimator can be retrieved by soft-thresholding the sample covariance matrix. Based on this result, a closed-form solution has been obtained that is optimal when the thresholded sample covariance matrix has an acyclic structure. In this work, we prove an extension of this result to *generalized* graphical lasso (GGL), where additional sparsity constraints are imposed based on prior knowledge. Furthermore, we describe a recursive closed-form solution for the problem when the thresholded sample covariance matrix is chordal. By building upon this result, we describe a novel *Newton-Conjugate Gradient* algorithm that can efficiently solve the GGL with general structures. 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 instances 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.**

I. Introduction

With the growing size of the datasets collected from real-world problems, the graphical models have become a powerful statistical tool to extract and represent the underlying hidden structure of the variables. These models, commonly known as Markov Random Fields (MRFs), have simple and intuitive interpretation: The conditional dependencies of different variables are captured via weighted edges in their associated graph representation. The application of MRFs span from computer vision and natural language processing to economics [11], [25], [27]. One of the most commonly known classes of MRFs is Gaussian Graphical Models (GGMs). The applicability of GGM is contingent upon estimating the covariance or inverse covariance matrix (also known as concentration matrix) based on a limited number of independent and identically distribution (i.i.d.) samples drawn from a multivariate Gaussian distribution [6], [14], [29], [47].

In particular, we consider the problem of estimating an $n\times n$ covariance matrix Σ (or its inverse Σ^{-1}) of an n-variate probability distribution from N i.i.d. samples $\mathbf{x}_1,\mathbf{x}_2,\dots,\mathbf{x}_N$ drawn from the same probability distribution. In many applications [11], [25], [27], 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 [14].

Imposing sparsity upon Σ^{-1} gives rise to a shrinkage estimator for the inverse covariance matrix. 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 is to solve the following ℓ_1-regularized problem

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

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 the 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], stems from the famous maximum likelihood estimation of the inverse covariance matrix in the Gaussian setting. For more general distributions, (1) corresponds to the ℓ_1-regularized log-determinant Bregman divergence, which is a widely used method for measuring the distance between the true and estimated parameters of a probability distribution [36]. The graphical lasso is known to enjoy surprising statistical guarantees [36], [39], some of which are direct extensions of earlier work on the classical lasso [23], [32], [33], [45]. Variations on this theme are 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.

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 [21], and its "big-data" extension BIG-QUIC [22]. 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. Therefore, while the ℓ_1-regularized problem (1) is technically convex, it is considered intractable for real large-scale datasets: graphical lasso is used in gene regulatory networks to infer the interactions between various genes in response to different disease processes [30]. The number of genes under study in humans can easily reach tens of thousands; a size that is far beyond the capability of existing solvers. Graphical lasso is also a popular method for understanding the partial correlation between different

Salar Fattahi and Richard Y. Zhang are with the Department of Industrial Engineering and Operations Research, University of California, Berkeley. Somayeh Sojoudi is with the Departments of Electrical Engineering and Computer Sciences and Mechanical Engineering. This work was supported by the ONR Award N00014-18-1-2526, NSF Award 1808859, and DARPA Award D16AP00002.

brain regions in response to physical and mental activities. According to MyConnectome project[1], the full brain mask can include hundreds of thousands of voxels, thus making it prohibitive to solve using available numerical solvers.

This paper is organized as follows. In the rest of this section, we explain the existing results on graphical lasso and thresholding, a summary of our contributions, and its connection to state-of-the-art methods. In Section II, we provide sufficient conditions for the equivalence between *generalized* graphical lasso and a soft-thresholding method. Section III is devoted to the statement of our recursive closed-form solution for graphical lasso with chordal structures. Our main iterative algorithm for solving graphical lasso with general structures is provided in Section IV. Section VII focuses on the sketch of the proofs and the technical details of intermediate lemmas. The simulation results are demonstrated in Section V.

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, denote $X \in \mathbb{S}^n$ 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 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 sets 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 $X \in \mathbb{S}_{++}^n$. Sets are denoted by calligraphic letters. Given a sparsity set $\mathcal{G}$, we define $\mathbb{S}_{\mathcal{G}}^n \subseteq \mathbb{S}^n$ as the set of $n \times n$ real symmetric matrices whose (i,j)-th element is zero if $(i,j) \notin \mathcal{G}$. For a square matrix M, let $\operatorname{diag}(M)$ be a same-size diagonal matrix collecting the diagonal elements of M.

A. Graphical lasso, soft-thresholding, and MDMC

The high computational cost of graphical lasso has inspired a number of heuristics that are significantly cheaper to use. 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 $[\mathbf{x}_1 - \bar{\mathbf{x}}, \mathbf{x}_2 - \bar{\mathbf{x}}, \dots, \mathbf{x}_N - \bar{\mathbf{x}}]$.

In a recent line of work [13], [28], [41], the simple heuristic of thresholding was shown to enjoy some surprising guarantees. In particular, [41] and [13] proved that when the ℓ_1 regularization is imposed over only the off-diagonal elements of X, under some assumptions, the *sparsity pattern* of the associated estimator from the graphical lasso 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 its **sparsity set**

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

The **support graph** of $\mathcal{G}$, shown as $\operatorname{supp}(\mathcal{G})$, is obtained by viewing each element (i,j) in $\mathcal{G}$ as an edge between the i-th and j-th vertex in an undirected graph on n nodes. Moreover, [41] and [13] showed that the estimator X can be recovered by solving a version of (1) in which the sparsity set $\mathcal{G}$ is explicitly imposed, as in

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

Recovering the exact value of X (and not just its sparsity pattern) is important because it provides a quantitative measure of the conditional dependence between variables.

The assumptions needed for the sparsity pattern of the graphical lasso to be equivalent to that of the thresholding are hard to check but relatively mild. Indeed, [13] proved that these assumptions 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. By building upon this result, [13] introduces a closed-form solution for graphical lasso that is proven to be optimal when the soft-thresholded sample covariance matrix induces an *acyclic* graph.

B. Summary of Contributions

The purpose of this paper is three-fold.

Goal 1: We derive an extension of the guarantees derived in [13], [28], [41] for a slightly more general version of the problem that we call *generalized* graphical lasso (GGL):

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

GGL is (1) penalized by a weighted regularization coefficient $\lambda_{i,j}$ on the off-diagonals, and with *a priori* sparsity set $\mathcal{H}$ imposed as an additional constraint. We use the sparsity set $\mathcal{H}$ to incorporate prior information on the structure of the graphical model. For example, if the sample covariance C is collected over a large-scale graph–such as gene regulatory networks, social networks, and transportation systems–then far-away variables can be assumed as pairwise conditionally independent [8], [20], [35]. Including these neighborhood

[1]The reader is referred to http://myconnectome.org/wp/ for more information.

relationships into $\mathcal{H}$ can regularize the statistical problem, as well as reduce the numerical cost for a solution.

In Section II, we describe a procedure to transform GGL (5) into *maximum determinant matrix completion* (MDMC) problem–which is the dual of the problem (4)–in the same style as prior results by [13] for graphical lasso. More specifically, we soft-threshold the sample covariance C and then project this matrix onto the sparsity set $\mathcal{H}$. 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 an appropriately designed MDMC problem.

Goal 2: Upon converting the GGL to an MDMC problem, we generalize the closed-form solution of [13] to the cases where the soft-thresholded sample covariance has a chordal structure. More specifically, we show that the optimal solution of (5) can be obtained using a closed-form recursive formula when the soft-thresholded sample covariance matrix has a chordal structure. As previously pointed out, most of the numerical algorithms require a large number of iterations (at least $\mathcal{O}(n)$) for solving the graphical lasso and have the worst-case per-iteration complexity of $\mathcal{O}(n^3)$. We show that our proposed recursive formula requires a number of iterations growing linearly in the size of the problem, and that the complexity of each iteration is dependent on the size of the maximum clique in its support graph. Therefore, given the soft-thresholded sample covariance matrix (which can be obtained while constructing the sample covariance matrix), the complexity of solving the graphical lasso for sparse and chordal structures reduces from $\mathcal{O}(n^4)$ to $\mathcal{O}(n)$. Notice that acyclic graphs are chordal and therefore, the explicit formula of [13] is a special case of our proposed solution.

Goal 3: Based on the connection between the GGL and MDMC problems and the closed-form recursive solution for chordal structures, we develop an efficient iterative algorithm for the GGL with nonchordal structures. In particular, instead of directly solving the GGL, we solve its MDMC counterpart based on the chordal embedding approach of [3], [5], [9] and then recover the optimal solution. We embed $\mathcal{G}_{\mathcal{H}} = \mathcal{G} \cap \mathcal{H}$ within a chordal $\tilde{\mathcal{G}} \supset \mathcal{G}_{\mathcal{H}}$ to result in a convex optimization problem over $\mathbb{S}^n_{\tilde{\mathcal{G}}}$, namely the space of real symmetric matrices with the sparsity pattern $\tilde{\mathcal{G}}$. This way, the constraint $X \in \mathbb{S}^n_{\tilde{\mathcal{G}}}$ is *implicitly* imposed, meaning that we simply ignore the nonzero elements not in $\tilde{\mathcal{G}}$. This will significantly reduce the number of variables in the problem. After the embedding step, we solve the GGL (or to be precise, its associated MDMC) on $\mathbb{S}^n_{\tilde{\mathcal{G}}}$–as opposed to significantly larger cone of positive definite matrices–using a custom Newton-CG method[2]. Roughly speaking, this will enable us to take advantage of fast operations on chordal structures, such as inversion, multiplication, and projection, in order to achieve a cheap per-iteration complexity in the algorithm, thereby significantly reducing its running time. The main idea behind our proposed algorithm is to use an inner conjugate gradients (CG) loop to solve the Newton subproblem of an outer Newton's method. The proposed algorithm has a number of features designed to exploit problem structure, including the sparse chordal property of $\tilde{\mathcal{G}}$, duality, and the ability for CG and Newton to converge superlinearly; these are outlined in Section IV.

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

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

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

from which the optimal solution of the GGL can be efficiently recovered. Most importantly, the algorithm is highly efficient in practice. In Section V, 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 as large as $n = 200,000$ (more than 20 billion variables in graphical lasso), in less than an hour on a laptop computer, while the state-of-the-art methods cannot even start their iterations for these instances.

C. 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. [12] introduced a class of graphical lasso in which the true graphical model is assumed to have a Laplacian structure. This structure commonly appears in signal and image processing [31]. *Pathway* graphical model is introduced in [18] which takes into account *a priori* knowledge on the conditional independence between different variables.

Algorithms for graphical lasso. Algorithms for graphical lasso are usually based on some mixture of Newton [34], proximal Newton [21], [22], iterative thresholding [37], and (block) coordinate descent [14], [42]. 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 $O(n^3)$.

Algorithms for MDMC. Our algorithm is inspired by a line of results [3], [5], [9], [24] for minimizing the log-det penalty on chordal sparsity patterns, culminating in the CVXOPT package [4]. These are Newton algorithms that solve the Newton subproblem by explicitly forming and factoring the fully-dense Newton matrix. When $|\tilde{\mathcal{G}}| = O(n)$, 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 $\mathcal{G}$ to yield the chordal $\tilde{\mathcal{G}}$. 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 IV-D). In practice, CG converges much faster than its worst-case bound, because it is able to exploit eigenvalue clustering to achieve superlinear convergence. The preconditioned CG (PCG) algorithm of [9] is superficially similar to our algorithm, but instead applies PCG directly to the log-det minimization. Its convergence is much slower than the companion sparse interior-point method, as demonstrated by their numerical results.

II. Equivalence between GGL and Thresholding

Let $P_{\mathcal{H}}(X)$ denote the projection operator from $\mathbb{S}^n$ onto $\mathbb{S}^n_{\mathcal{H}}$, i.e. by setting all $X_{i,j} = 0$ if $(i,j) \notin \mathcal{H}$. 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 $P_{\mathcal{H}}(C_\lambda)$—the projection of the soft-thresholded matrix C_λ in (7) onto $\mathcal{H}$—to have the same sparsity pattern as the GGL estimator $\hat{X}$ in (5). For brevity, the technical proofs are omitted; these can be found in Section VII.

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

Definition 1. *[13] Given a matrix $M \in \mathbb{S}^n$, define $\mathcal{G}_M = \{(i,j) : M_{i,j} \neq 0\}$ as its sparsity pattern. Then, M is called* ***inverse-consistent*** *if there exists a matrix $N \in \mathbb{S}^n$ such that*

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

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

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

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 $(i,j) \in \mathcal{G}_M$, the (i,j)-th elements of M and $(M + M^{(c)})^{-1}$ have opposite signs.*

Example 1. Consider the matrix:

$$M = \begin{bmatrix} 1 & 0.3 & 0 & 0 \\ 0.3 & 1 & -0.4 & 0 \\ 0 & -0.4 & 1 & 0.2 \\ 0 & 0 & 0.2 & 1 \end{bmatrix} \tag{9}$$

We show that M is both inverse- and sign-consistent. Consider the matrix $M^{(c)}$ defined as

$$M^{(c)} = \begin{bmatrix} 0 & 0 & -0.12 & -0.024 \\ 0 & 0 & 0 & -0.08 \\ -0.12 & 0 & 0 & 0 \\ -0.024 & -0.08 & 0 & 0 \end{bmatrix} \tag{10}$$

$(M + M^{(c)})^{-1}$ can be written as

$$\begin{bmatrix} \frac{1}{0.91} & \frac{-0.3}{0.91} & 0 & 0 \\ \frac{-0.3}{0.91} & 1 + \frac{0.09}{0.91} + \frac{0.16}{0.84} & \frac{0.4}{0.84} & 0 \\ 0 & \frac{0.4}{0.84} & 1 + \frac{0.16}{0.84} + \frac{0.04}{0.96} & \frac{-0.2}{0.96} \\ 0 & 0 & \frac{-0.2}{0.96} & \frac{1}{0.96} \end{bmatrix} \tag{11}$$

Note that:

- $M + M^{(c)}$ is positive-definite.
- The sparsity pattern of M is the complement of that of $M^{(c)}$.
- The sparsity pattern of M and $(M + M^{(c)})^{-1}$ are equivalent.
- The nonzero off-diagonal entries of M and $(M+M^{(c)})^{-1}$ have opposite signs.

Therefore, it can be inferred that M is both inverse- and sign-consistent, and $M^{(c)}$ is its inverse-consistent complement. □

Definition 2. *We take the usual element-wise max-norm to exclude the diagonal, as in $\|M\|_{\max} = \max_{i \neq j} |M_{ij}|$, and adopt the $\beta(\mathcal{G}, \alpha)$ function defined with respect to the sparsity set $\mathcal{G}$ and scalar $\alpha > 0$:*

$$\beta(\mathcal{G}, \alpha) = \max_{M \succ 0} \|M^{(c)}\|_{\max} \tag{12}$$

$$\text{s.t. } M \in \mathbb{S}^n_{\mathcal{G}} \text{ and } \|M\|_{\max} \leq \alpha \tag{13}$$

$$M_{i,i} = 1 \quad \forall i \in \{1, \ldots, n\} \tag{14}$$

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

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

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

$$\beta\left(G_{\mathcal{H}}, \|\tilde{C}\|_{\max}\right) \leq \min_{(k,l) \notin \mathcal{G}_{\mathcal{H}}} \frac{\lambda_{k,l} - |C_{k,l}|}{\sqrt{C_{k,k} \cdot C_{l,l}}} \tag{15}$$

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

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

Proof. See Section VII. □

Remark 1. *Theorem 1 is an extension to Theorem 12 in [13], which shows that, under similar conditions, the soft-thresholded sample covariance matrix has the same sparsity pattern as the optimal solution of graphical lasso. We have extended these results to the GGL by considering an intermediate* weighted graphical lasso *and showing that, by carefully selecting the weights of the regularization coefficients, one can adopt the same arguments in [13] to show the equivalence between the GGL and thresholding; a detailed discussion can be found in Section VII.*

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 GGL requires large regularization coefficients. This leads to numerous zero elements in $\tilde{C}$ and forces the magnitude of the nonzero elements to be small. This means that, in most instances, $\tilde{C}$ is positive definite or even diagonally dominant. Furthermore, the next two propositions reveal that the second and third conditions of Theorem 1 are not restrictive when the goal is to recover a sparse solution for the GGL.

Proposition 1. *There exists a strictly positive constant number $\zeta(\mathcal{G}_{\mathcal{H}})$ such that*

$$\beta\left(\mathcal{G}_{\mathcal{H}}, \|\tilde{C}\|_{\max}\right) \leq \zeta(\mathcal{G}_{\mathcal{H}})\|\tilde{C}\|_{\max}^2, \tag{16}$$

and therefore, Condition 3 of Theorem 1 is satisfied if

$$\zeta(\mathcal{G}_{\mathcal{H}})\|\tilde{C}\|_{\max}^2 \leq \min_{\substack{k \neq l \\ (k,l)\notin\mathcal{G}_{\mathcal{H}}}} \frac{\lambda_{k,l} - |C_{k,l}|}{\sqrt{C_{k,k}\cdot C_{l,l}}} \tag{17}$$

Proof. The proof is similar to that of Lemma 13 in [13]. The details are omitted for brevity. □

Proposition 2. *There exist strictly positive constant numbers $\alpha_0(\mathcal{G}_{\mathcal{H}})$ and $\gamma(\mathcal{G}_{\mathcal{H}})$ such that $\tilde{C}$ is sign-consistent if $\|\tilde{C}\|_{\max} \leq \alpha_0(\mathcal{G}_{\mathcal{H}})$ and*

$$\gamma(\mathcal{G}_{\mathcal{H}}) \times \|\tilde{C}\|_{\max}^2 \leq \|\tilde{C}\|_{\min} \tag{18}$$

where $\|\tilde{C}\|_{\min}$ is defined as the minimum absolute value of the nonzero off-diagonal entries of $\tilde{C}$.

Proof. The proof is similar to that of Lemma 14 in [13]. The details are omitted for brevity. □

Both of these propositions suggest that, when λ is large, i.e., when a sparse solution is sought for the GGL, $\|\tilde{C}\|_{\max} \ll 1$ and hence, Conditions 2 and 3 of Theorem 1 are satisfied.

Theorem 1 leads to the following corollary, which asserts that the optimal solution of GGL 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 $C_{\mathcal{H}} = P_{\mathcal{H}}(C_\lambda)$.

Corollary 1. *Suppose that the conditions in Theorem 1 are satisfied. Define $\hat{S}$ as the solution to the following MDMC problem*

$$\begin{aligned}\hat{S} = \; & \underset{S\succeq 0}{\textit{maximize}} \;\; \log\det S \\ & \textit{subject to } S_{i,j} = C_{\mathcal{H}} \textit{ for all } (i,j) \textit{ where } (C_{\mathcal{H}})_{i,j} \neq 0\end{aligned} \tag{19}$$

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

Proof. Under the conditions of Theorem 1, the (i,j)-th element of the solution $\hat{X}_{i,j}$ has *opposite signs* to the corresponding element in $(C_{\mathcal{H}})_{i,j}$, 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

$$\begin{aligned}\hat{X} = \underset{X\succ 0}{\text{minimize}} \;\; & \underbrace{\operatorname{tr} CX - \sum_{(i,j)\in\mathcal{G}} \operatorname{sign}(C_{i,j})\lambda_{i,j}X_{i,j}}_{\equiv \operatorname{tr} C_\lambda X} - \log\det X \\ \text{subject to } & X_{i,j} = 0 \qquad \forall (i,j) \notin H.\end{aligned} \tag{20}$$

The constraint $X \in \mathbb{S}^n_{\mathcal{H}}$ further makes $\operatorname{tr} C_\lambda X = \operatorname{tr} C_\lambda P_{\mathcal{H}}(X) = \operatorname{tr} P_{\mathcal{H}}(C_\lambda) X \equiv \operatorname{tr} C_{\mathcal{H}} X$. Taking the dual of (20) yields (19); complementary slackness yields $\hat{S} = \hat{X}^{-1}$. □

Recall that the main goal of the GGL is to promote sparsity in the estimated inverse covariance matrix. Corollary 1 shows that, instead of merely identifying the sparsity structure of the optimal solution using Theorem 1, the soft-thresholded sample covariance matrix can be further exploited to convert the GGL to the MDMC problem; this conversion is at the core of our subsequent algorithms for solving the GGL to optimality.

III. Warm-up: Chordal Structures

In this section, we propose a recursive closed-form solution for the GGL when the soft-thresholded sample covariance has a chordal structure. To this goal, we first review the properties of sparse and chordal matrices and their connections to the MDMC problem.

A. *Sparse Cholesky factorization*

Consider solving an $n \times n$ symmetric positive definite linear system

$$Sx = b$$

by Gaussian elimination. The standard procedure comprises a *factorization* step, where S is decomposed into the (unique) Cholesky factor matrices LDL^T, in which D is diagonal and L is lower-triangular with a unit diagonal, and a *substitution* step, where the two triangular systems of linear equations $Ly = b$ and $DL^Tx = y$ are solved to yield x.

In the case where S is sparse, the Cholesky factor L is often also sparse. It is common to store the sparsity pattern of L in the *compressed column storage* format: a set of indices $\mathcal{I}_1, \ldots, \mathcal{I}_d \subseteq \{1, \ldots, d\}$ in which

$$\mathcal{I}_j = \{i \in \{1, \ldots, d\} : i > j, L_{i,j} \neq 0\}, \tag{21}$$

encodes the locations of off-diagonal nonzeros in the j^{th} column of L. (The diagonal elements are not included because the matrix L has a unit diagonal by definition).

After storage has been allocated and the sparsity structure has been determined, the numerical values of D and L are computed using a sparse Cholesky factorization algorithm. This requires the use of the associated *elimination tree* $T = \{\mathcal{V}, \mathcal{E}\}$, which is a rooted tree (or forest) on n vertices, with edges $\mathcal{E} = \{\{1, p(1)\}, \ldots, \{d, p(d)\}\}$ defined to connect each j^{th} vertex to its parent at the $p(j)^{\text{th}}$ vertex (except root nodes, which have "0" as their parent), as in

$$p(j) = \begin{cases} \min \mathcal{I}_j & |\mathcal{I}_j| > 0, \\ 0 & |\mathcal{I}_j| = 0, \end{cases} \tag{22}$$

in which $\min \mathcal{I}_j$ indicates the (numerically) smallest index in the index set $\mathcal{I}_j$ [26]. The elimination tree encodes the dependency information between different columns of L, thereby allowing information to be passed without explicitly forming the matrix.

B. Chordal sparsity patterns

For a sparsity set $\mathcal{S}$, recall that its support graph, denoted by supp($\mathcal{S}$), is defined as a graph with the vertex set $\mathcal{V} = \{1, 2, ..., n\}$ and the edge set $\mathcal{S}$. Moreover, $\mathcal{S}$ is said to be *chordal* if its support graph does not contain an induced cycle with length greater than three. $\mathcal{S}$ is said to *factor without fill* if every $U \in \mathbb{S}^n_{\mathcal{S}}$ can be factored into LDL^T such that $L + L^T \in \mathbb{S}^n_{\mathcal{S}}$. If $\mathcal{S}$ factors without fill, then its support graph is chordal. Conversely, if the support graph is chordal, then there exists a permutation matrix Q such that QUQ^T factors without fill for every matrix $U \in \mathbb{S}^n_{\mathcal{S}}$ [15]. This permutation matrix Q is called the *perfect elimination ordering* of the chordal set $\mathcal{S}$.

C. Recursive solution of the MDMC problem.

An important application of chordal sparsity patterns is the efficient solution of the MDMC problem (19). The first-order optimality conditions for the GGL entails that

$$P_{\mathcal{G}_{\mathcal{H}}}(\hat{X}^{-1}) = C_\lambda \tag{23}$$

On the other hand, Corollary 1 shows that

$$\hat{S} = \hat{X}^{-1} \tag{24}$$

In the case that the sparsity set $\mathcal{G}_{\mathcal{H}}$ factors without fill, [2] showed that (23) is actually a linear system of equations over the Cholesky factor L and D of the solution $\hat{X} = LDL^T$; their numerical values can be explicitly computed using a recursive formula.

Algorithm 1. *([2], Algorithm 4.2)*
Input. *Matrix $C_\lambda \in \mathbb{S}^n_{\mathcal{G}_{\mathcal{H}}}$ that has a positive definite completion.*
Output. *The Cholesky factors L and D of $\hat{X} = LDL^T \in \mathbb{S}^n_{\mathcal{G}_{\mathcal{H}}}$ that satisfy $P_{\mathcal{G}_{\mathcal{H}}}(\hat{X}^{-1}) = C_\lambda$.*
Algorithm. *Iterate over $j \in \{1, 2, \ldots, n\}$ in reverse, i.e. starting from n and ending in 1. For each j, compute $D_{j,j}$ and the jth column of L from*

$$L_{I_j,j} = -V_j^{-1}(C_\lambda)_{\mathcal{I}_j,j},$$
$$D_{j,j} = ((C_\lambda)_{j,j} + (C_\lambda)^T_{\mathcal{I}_j,j} L_{\mathcal{I}_j,j})^{-1}$$

and compute the update matrices

$$V_i = P^T_{\mathcal{I}_i \cup \{i\}, \mathcal{I}_i} \begin{bmatrix} C_{j,j} & C^T_{\mathcal{I}_j,j} \\ C_{\mathcal{I}_j,j} & V_j \end{bmatrix} P_{\mathcal{I}_i \cup \{i\}, \mathcal{I}_i}$$

for each i satisfying $p(i) = j$, i.e. each child of j in the elimination tree.

Note that finding the perfect elimination ordering is NP-hard in general. However, if $\mathcal{G}_{\mathcal{H}}$ is chordal, then we may find the perfect elimination ordering Q in linear time [43], and apply the above algorithm to the matrix $QC_{\mathcal{H}}Q^T$, whose sparsity set does indeed factor without fill.

The algorithm takes n steps, and the jth step requires a size-$|\mathcal{I}_j|$ linear solve and vector-vector product. Define $w = \max_j |\mathcal{I}_j| - 1$, which is commonly referred to as the *treewidth* of supp($\mathcal{G}_{\mathcal{H}}$) and has the interpretation of the largest clique in supp($\mathcal{G}_{\mathcal{H}}$) minus one. Combined, the algorithm has the time complexity $\mathcal{O}(w^3 n)$. This means that the matrix completion algorithm is linear-time if the treewidth of supp($\mathcal{G}_{\mathcal{H}}$) is on the order of $\mathcal{O}(1)$. Note that, for acyclic graphs, we have $w = 1$.

Next, we show that if the equivalence between the thresholding method and GGL holds, the optimal solution of the GGL can be obtained using Algorithm 1.

Proposition 3. *Suppose that the conditions in Theorem 1 are satisfied. Then, Algorithm 1 can be used to find $\hat{X}$ if $\mathcal{G}_{\mathcal{H}}$ is chordal.*

Proof. According to Algorithm 1, the output satisfies $P_{\mathcal{G}_{\mathcal{H}}}(\hat{X}^{-1}) = C_\lambda$ which is the first-order optimality condition for the GGL. Together with the uniqueness of the optimal solution, this concludes the proof. □

Proposition 3 suggests that solving the GGL through its MDMC counterpart is much easier and can be performed in linear time using Algorithm 1 when the soft-thresholded sample covariance matrix has a sparse and chordal structure. Inspired by this observation, in the next section, we propose a highly efficient iterative algorithm for solving the GGL with nonchordal soft-thresholded sample covariance matrix, which is done by converting it to an MDMC problem and embedding its nonchordal structure within a chordal pattern. Through this chordal embedding, we show that the number of iterations of the proposed algorithm is essentially constant, each with a linear time complexity.

IV. General Case: Nonchordal Structures

This section describes our algorithm to solve MDMC (4) in which the sparsity set $\mathcal{G}$ is *nonchordal*. 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 $\tilde{\mathcal{G}}$ for $\mathcal{G}$. The optimal chordal embedding with the fewest number of nonzeros $|\tilde{\mathcal{G}}|$ is NP-hard to compute, but a good-enough embedding with $O(n)$ nonzeros is sufficient for our purposes. Obtaining a good $\tilde{\mathcal{G}}$ with $|\tilde{\mathcal{G}}| = O(n)$ 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 [3], [5], [9], [44], we can optimize directly in the space $\mathbb{S}^n_{\tilde{\mathcal{G}}}$, while ignoring all matrix elements outside of $\tilde{\mathcal{G}}$. If $|\tilde{\mathcal{G}}| = O(n)$, 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 [5].

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

4. Conjugate gradients (CG) converges in $O(1)$ iterations. Our main result (Theorem 2) 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, as in existing methods [3], [5], [9]. 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 IV-E. For the sake of simplicity of notation, we use $\mathcal{G}$ instead of $\mathcal{G}_{\mathcal{H}}$–which is the intersection of the sparsity sets $\mathcal{G}$ and $\mathcal{H}$–in the rest of this section.

A. Efficient chordal embedding

Following [9], we begin by reformulating (4) into a sparse chordal matrix program

$$\begin{aligned} \hat{X} = \;& \text{minimize } \operatorname{tr} CX - \log\det X \\ & \text{subject to } X_{i,j} = 0 \quad \forall (i,j) \in \tilde{\mathcal{G}} \backslash \mathcal{G}. \\ & \qquad X \in \mathbb{S}^n_{\tilde{\mathcal{G}}}. \end{aligned} \tag{25}$$

in which $\tilde{\mathcal{G}}$ is a *chordal embedding* for $\mathcal{G}$: a sparsity pattern $\tilde{\mathcal{G}} \supset \mathcal{G}$ whose support 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 $C \in \mathbb{S}^n_{\mathcal{G}}$ be a positive definite matrix. Compute its unique lower-triangular Cholesky factor L satisfying $C = LL^T$. Ignoring perfect numerical cancellation, the sparsity set of $L + L^T$ is a chordal embedding $\tilde{\mathcal{G}} \supset \mathcal{G}$.*

Proof. The original proof is due to [38]; see also [44]. □

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

Corollary 2. *Let Π be a permutation matrix. For the same $C \in \mathbb{S}^n_{\mathcal{G}}$ in Proposition 4, we compute the unique Cholesky factor satisfying $\Pi C \Pi^T = LL^T$. Ignoring perfect numerical cancellation, the sparsity set of $\Pi(L + L^T)\Pi^T$ is a chordal embedding $\tilde{\mathcal{G}} \supset \mathcal{G}$.*

Proof. See [44]. □

```
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
```

Fig. 1: MATLAB code for chordal embedding. 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 [46]. However, good orderings are easily found using heuristics developed for numerical linear algebra, like minimum degree ordering [16] and nested dissection [1], [17]. In fact, [17] proved that nested dissection is $\mathcal{O}(\log(n))$ suboptimal for bounded-degree graphs, and noted that "we do not know a class of graphs for which [nested dissection is suboptimal] by more than a constant factor."

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

B. 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

$$\mathcal{K} = \mathbb{S}^n_+ \cap \mathbb{S}^n_{\tilde{\mathcal{G}}}, \qquad \mathcal{K}_* = \{S \bullet X \geq 0 : S \in \mathbb{S}_{\tilde{\mathcal{G}}}\} = P_{\tilde{\mathcal{G}}}(\mathbb{S}^n_+) \tag{26}$$

Then, (25) can be posed as the primal-dual pair:

$$\arg\min_{X \in \mathcal{K}} \{C \bullet X + f(X) : A^T(X) = 0\}, \tag{27}$$

$$\arg\max_{S \in \mathcal{K}_*, y \in \mathbb{R}^m} \{-f_*(S) : S = C - A(y)\}, \tag{28}$$

in which f and f_* are the "log-det" barrier functions on $\mathcal{K}$ and $\mathcal{K}_*$ as introduced by [3], [5], [9]:

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

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

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

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

$$f_*(S) = n + \log\det X, \tag{29a}$$

$$\nabla f_*(S) = -X, \tag{29b}$$

$$\nabla^2 f_*(S)[Y] = \nabla^2 f(X)^{-1}[Y]. \tag{29c}$$

Assuming that $\tilde{\mathcal{G}}$ 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 [5].

C. Solving the dual problem

Our algorithm actually solves the dual problem (28), which can be rewritten as an unconstrained optimization problem

$$\hat{y} \equiv \arg\min_{y\in\mathbb{R}^m} g(y) \equiv f_*(C_\lambda - A(y)). \tag{30}$$

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 (28) is easier to solve than the primal (27) 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

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

which is a relaxation of the nonchordal problem posed over $\mathbb{S}^n_{\mathcal{G}}$ since $\tilde{\mathcal{G}} \supset \mathcal{G}$. As observed by several previous authors [9], this relaxation is a high quality guess, and $\tilde{X}$ is often "almost feasible" for the original nonchordal problem posed over $\mathbb{S}^n_{\mathcal{G}}$, as in $\tilde{X} \approx P_{\mathcal{G}}(\tilde{X})$. Simple algebra shows that the gradient ∇g evaluated at the origin has the Euclidean norm $\|\nabla g(0)\| = \|\tilde{X} - P_{\mathcal{G}}(\tilde{X})\|_F$, so if $\tilde{X} \approx P_{\mathcal{G}}(\tilde{X})$ 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.

D. 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

$$\nabla^2 g(y)\Delta y = -\nabla g(y). \tag{31}$$

The Hessian matrix $\nabla^2 g(y)$ is fully dense, but matrix-vector products with it cost just $O(n)$ operations. This insight motivates the solution of (31) using an iterative Krylov subspace method like conjugate gradients (CG). We defer to standard texts [7] for implementation details, and only note that CG requires a single matrix-vector product with the Hessian $\nabla^2 g(y)$ at each iteration, in $O(n)$ time and memory when $|\tilde{\mathcal{G}}| = O(n)$. 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

$$\left\lceil \sqrt{\kappa_g}\log(2/\epsilon)\right\rceil \text{ CG iterations,} \tag{32}$$

where $\kappa_g = \|\nabla^2 g(y)\|\|\nabla^2 g(y)^{-1}\|$ is the condition number of the Hessian matrix [19], [40]. 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, over the entire trajectory of Newton's method. Below, we state our main result, which asserts that 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 2. *The condition number of the Hessian matrix is bound*

$$\operatorname{cond}(\nabla^2 g(y_k)) \le \frac{\lambda_{\max}(\mathbf{A}^T\mathbf{A})}{\lambda_{\min}(\mathbf{A}^T\mathbf{A})}\left(2 + \frac{\phi_{\max}^2 \lambda_{\max}(X_0)}{\lambda_{\min}(\hat{X})}\right)^2$$

where:
- $\phi_{\max} = g(y_0) - g(\hat{y})$ *is the initial infeasibility,*
- $\mathbf{A} = [\operatorname{vec} A_1, \ldots, \operatorname{vec} A_m]$ *is the vectorized data matrix,*
- $X_0 = -\nabla f_*(C - A(y_0))$ *satisfies* $P_{\tilde{\mathcal{G}}}(X_0^{-1}) = C - A(y_0)$, $X_0 \in \mathcal{K}$,
- *and* $\hat{X} = -\nabla f_*(C - A(\hat{y}))$ *satisfies* $P_{\tilde{\mathcal{G}}}(\hat{X}^{-1}) = C - A(\hat{y})$, $\hat{X} \in \mathcal{K}$.

Proof. See Section VII. □

Note that, without loss of generality, we assume $\lambda_{\min}(\mathbf{A}^T\mathbf{A}) > 0$; otherwise, the linearly-dependent rows of A^T can be eliminated until the reduced A^T is full row-rank. Applying Theorem 2 to (32) 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 by the $O(n)$ cost of each CG iteration proves the claimed time complexity in (6). The associated memory complexity is $O(n)$, i.e. the number of decision variables in $\tilde{\mathcal{G}}$.

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 [19], [40]. 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.

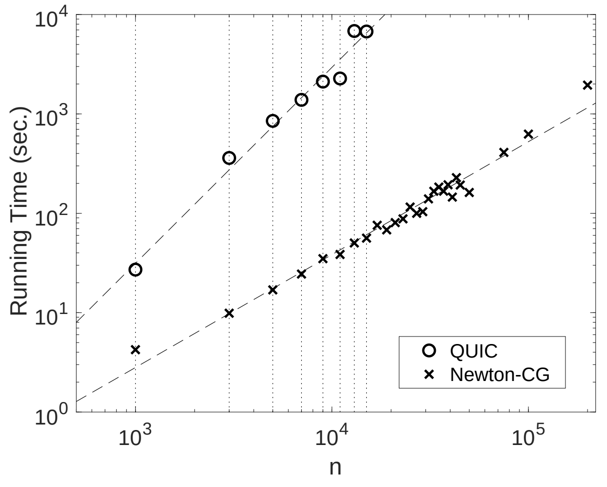
E. The full algorithm

In this subsection, we assemble the previously-presented steps and summarize the full algorithm. To begin with, we compute a chordal embedding $\tilde{\mathcal{G}}$ for the sparsity pattern $\mathcal{G}$ of C_λ, using the code snippet in Figure 1. We use the embedding to reformulate (4) as (25), and solve the unconstrained problem $\hat{y} = \min_y g(y)$ defined in (30), using Newton's method

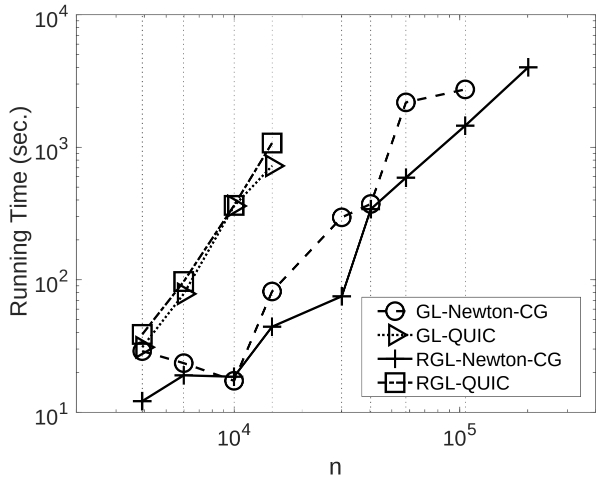
$$\begin{aligned} y_{k+1} &= y_k + \alpha_k \Delta y_k, \\ \Delta y_k &\equiv -\nabla^2 g(y_k)^{-1}\nabla g(y_k) \end{aligned}$$

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 [5].

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.



(a)



(b)

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

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

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

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 $\hat{X} = -\nabla f_*(C_\lambda - A(\hat{y}))$.

V. Numerical Results

Finally, we benchmark our algorithm[3] against `QUIC` [21], commonly considered the fastest solver for graphical lasso or RGL[4]. (Another widely-used algorithm is `GLASSO` [14], but we found it to be significantly slower than `QUIC`.) We consider three case studies:

1) Banded graphs without thresholding, to verify the claimed $O(n)$ complexity of our MDMC algorithm on problems with a nearly-banded structure.
2) Banded graphs, to verify the ability of our threshold-MDMC procedure to recover the correct GL solution on problems with a nearly-banded structure.
3) Real-life Graphs, to benchmark the full threshold-MDMC procedure on graphs collected from real-life applications.

Experiments 1 and 3 are performed on a laptop computer with an Intel Core i7 quad-core 2.50 GHz CPU and 16GB RAM. Experiment 2 is performed on a desktop workstation with a slower Intel Core CPU, but 48 GB of 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 $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ 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.

After thresholding, we use our Newton-CG algorithm to solve the MDMC problem with respected to the thresholded matrix C_λ. We measure the quality of the resulting X by the following two metrics:

$$\text{relative optimality gap} \equiv \frac{\|P_G(C_\lambda - X^{-1})\|_F}{\|C_\lambda\|_F}, \tag{33}$$

$$\text{relative infeasibility} \equiv \frac{\|P_G(X) - X\|_F}{\|X\|_F}, \tag{34}$$

where G is the sparsity pattern associated with C_λ. If X is an exact solution to MDMC (4), then the first-order optimality condition would read $P_G(X^{-1}) = C_\lambda$ while the associated feasibility condition $X \in \mathbb{S}^n_G$ would imply $P_G(X) = X$. In this case, both the relative optimality gap and the relative infeasibility are identically zero.

A. Case Study 1: Banded Patterns without thresholding

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 the 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

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

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

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 of our MATLAB implementation. In either case, `QUIC`'s quadratic complexity limits it to $n = 1.5 \times 10^4$. 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.

B. Case Study 2: Banded Patterns

In the second case study, we repeat our first experiment over banded patterns, but also include the initial thresholding step. We restrict our attention to the RGL version of the problem, meaning that the bandwidth is known ahead of time, but the exact location of the nonzeros are unknown. The quality of an estimator $\hat{X}$ compared to the true inverse covariance matrix $\Theta \equiv \Sigma^{-1}$ is measured using the following three standard metrics:

$$\text{relative Frobenius loss } \equiv \|\hat{X} - \Theta\|_F / \|\Theta\|_F, \tag{35}$$

$$\text{TPR} \equiv \frac{|\{(i,j) : i \neq j, \hat{X}_{i,j} \neq 0, \Theta_{i,j} \neq 0\}|}{|\{(i,j) : i \neq j, \Theta_{i,j} \neq 0\}|}, \tag{36}$$

$$\text{FPR} \equiv \frac{|\{(i,j) : i \neq j, \hat{X}_{i,j} \neq 0, \Theta_{i,j} = 0\}|}{|\{(i,j) : i \neq j, \Theta_{i,j} = 0\}|}. \tag{37}$$

In words, the relative Frobenius loss is the normalized error of the estimator. The true positive rate (TPR) is a measure of *sensitivity*, and is defined as the proportion of actual nonzeros that are correctly identified as such. The false positive rate (FPR) is a measure of *specificity*, and is defined as the portion of zero elements that are misidentified as being nonzero.

The results are shown in Table I. The threshold-MDMC procedure is able to achieve the same statistical recovery properties as QUIC in a fraction of the solution time. In larger instances, both methods are able to almost exactly recover the true sparsity pattern. However, only our procedure continues to work with n up to 200000.

C. Case Study 3: Real-Life Graphs

The third 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* [10]—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 the two algorithms for this case study; the specific details are provided in Table II. 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 1 and Corollary 1 about (1) and (5): thresholding-MDMC does indeed solve graphical lasso and RGL.

VI. 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 the second step can be solved in closed-form. But more generally, if the resulting sparsity pattern has a *sparse chordal embedding*, then the second step can be performed using the Newton-CG algorithm described in this paper. In either case, the complexity of this second step is linear $O(n)$ time and memory. We tested the algorithm on both synthetic and real-life data, solving instances with as many as 200,000 variables

		Newton-CG						QUIC					
n	m	ℓ_F	TPR	FPR	sec	gap	infeas	ℓ_F	TPR	FPR	sec	diff. gap	speed-up
1000	2432	0.134	0.774	0.00	0.8	4.2e-17	2.4e-8	0.133	0.782	0.00	2.8	8.7e-5	3.5
2000	5105	0.187	0.873	0.00	1.9	7.0e-17	3.6e-8	0.186	0.870	0.00	35.4	5.9e-5	18.6
5000	12 802	0.200	0.897	0.00	6.0	8.7e-17	4.7e-8	0.199	0.897	0.00	383.6	3.3e-4	63.9
10 000	17 566	0.198	0.929	0.00	14.8	9.0e-17	7.7e-10	0.197	0.930	0.00	2432.8	2.1e-4	164.4
20 000	36 990	0.190	0.929	0.00	31.6	8.8e-17	7.0e-10	$*$	$*$	$*$	$*$	$*$	$*$
50 000	92 800	0.181	0.927	0.00	88.5	8.8e-17	4.9e-10	$*$	$*$	$*$	$*$	$*$	$*$
100 000	188 594	0.175	0.926	0.00	190.8	8.6e-17	5.2e-10	$*$	$*$	$*$	$*$	$*$	$*$
200 000	394 004	0.170	0.925	0.00	1255.3	8.5e-17	1.1e-9	$*$	$*$	$*$	$*$	$*$	$*$

TABLE I: 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, "ℓ_F" is the relative Frobenius loss, "TPR" is the true positive rate, "FPR" is the false positive rate, "sec" is the running time in seconds, "gap" is the relative optimality gap, "infeas" is the relative infeasibility, "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.

						Newton-CG			QUIC		
#	file name	type	n	m	m/n	sec	gap	infeas	sec	diff. gap	speed-up
1	freeFlyingRobot-7	GL	3918	20 196	5.15	28.9	5.7e-17	2.3e-7	31.0	3.9e-4	1.07
1	freeFlyingRobot-7	RGL	3918	20 196	5.15	12.1	6.5e-17	2.9e-8	38.7	3.8e-5	3.20
2	freeFlyingRobot-14	GL	5985	27 185	4.56	23.5	5.4e-17	1.1e-7	78.3	3.8e-4	3.33
2	freeFlyingRobot-14	RGL	5985	27 185	4.56	19.0	6.0e-17	1.7e-8	97.0	3.8e-5	5.11
3	cryg10000	GL	10 000	170 113	17.0	17.3	5.9e-17	5.2e-9	360.3	1.5e-3	20.83
3	cryg10000	RGL	10 000	170 113	17.0	18.5	6.3e-17	1.0e-7	364.1	1.9e-5	19.68
4	epb1	GL	14 734	264 832	18.0	81.6	5.6e-17	4.3e-8	723.5	5.1e-4	8.86
4	epb1	RGL	14 734	264 832	18.0	44.2	6.2e-17	3.3e-8	1076.4	4.2e-4	24.35
5	bloweya	GL	30 004	10 001	0.33	295.8	5.6e-17	9.4e-9	$*$	$*$	$*$
5	bloweya	RGL	30 004	10 001	0.33	75.0	5.5e-17	3.6e-9	$*$	$*$	$*$
6	juba40k	GL	40 337	18 123	0.44	373.3	5.6e-17	2.6e-9	$*$	$*$	$*$
6	juba40k	RGL	40 337	18 123	0.44	341.1	5.9e-17	2.7e-7	$*$	$*$	$*$
7	bayer01	GL	57 735	671 293	11.6	2181.3	5.7e-17	5.2e-9	$*$	$*$	$*$
7	bayer01	RGL	57 735	671 293	11.6	589.1	6.4e-17	1.0e-7	$*$	$*$	$*$
8	hcircuit	GL	105 676	58 906	0.55	2732.6	5.8e-17	9.0e-9	$*$	$*$	$*$
8	hcircuit	RGL	105 676	58 906	0.55	1454.9	6.3e-17	7.3e-8	$*$	$*$	$*$
9	co2010	RGL	201 062	1 022 633	5.08	4012.5	6.3e-17	4.6e-8	$*$	$*$	$*$

TABLE II: Details of case study 3. 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 relative optimality gap, "feas" is the relative infeasibility, "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.

to 7-9 digits of accuracy within an hour on a standard laptop computer. Our algorithm achieves the same Frobenius loss, true positive rate, and false positive rate as the existing state-of-the-art, but using just a fraction of the computation time.

VII. Proofs

In this section, we present the technical proofs of Theorems 1 and 2. To this goal, we need a number of lemmas.

A. Proof of Theorem 1

To prove Theorem 1, first we consider the optimality (KKT) conditions for the unique solution of the GGL with $\mathbb{S}^n_{\mathcal{H}} = \mathbb{S}^n$.

Lemma 1. *$\hat{X}$ is the optimal solution of the GGL with $\mathbb{S}^n_{\mathcal{H}} = \mathbb{S}^n$ if and only if it satisfies the following conditions for every $i,j \in \{1,2,...,n\}$:*

$$(\hat{X})^{-1}_{i,j} = C_{i,j} \qquad \text{if} \quad i = j \tag{38a}$$

$$(\hat{X})^{-1}_{i,j} = C_{i,j} + \lambda_{i,j} \times \operatorname{sign}(\hat{X}_{i,j}) \qquad \text{if} \quad \hat{X}_{i,j} \neq 0 \tag{38b}$$

$$C_{i,j} - \lambda_{i,j} \leq (\hat{X})^{-1}_{i,j} \leq \Sigma_{i,j} + \lambda_{i,j} \qquad \text{if} \quad \hat{X}_{i,j} = 0 \tag{38c}$$

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

Now, consider the following optimization:

$$\underset{X \succ 0}{\text{minimize}} \; \operatorname{tr} \tilde{C}X - \log\det X + \sum_{(i,j)\in\mathcal{H}} \tilde{\lambda}_{i,j}|X_{i,j}| + 2\sum_{(i,j)\notin\mathcal{H}} |X_{i,j}| \tag{39}$$

where

$$\tilde{C}_{i,j} = \frac{C_{i,j}}{\sqrt{C_{i,i} \times C_{j,j}}} \qquad \tilde{\lambda}_{i,j} = \frac{\lambda_{i,j}}{\sqrt{C_{i,i} \times C_{j,j}}} \tag{40}$$

Let $\tilde{X}$ denotes the optimal solution of (39) and recall $D = \operatorname{diag}(C)$. The following lemma relates $\tilde{X}$ to $\hat{X}$.

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

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

$$\min_{X\in\mathbb{S}^n_+} f(X) = \operatorname{tr} CX - \log\det X + \sum_{(i,j)\in\mathcal{H}} \lambda_{i,j}|X_{i,j}| + 2\max_k\{C_{k,k}\} \sum_{(i,j)\notin\mathcal{H}} |X_{i,j}| \tag{41}$$

Denote $X^\sharp$ as the optimal solution for (41). First, we show that $X^\sharp = \hat{X}$. Trivially, $\hat{X}$ is a feasible solution for (41) and hence $f(X^\sharp) \leq f(\hat{X})$. Now, we prove that $X^\sharp$ is a feasible solution for the GGL. To this goal, we show that $X^\sharp_{i,j} = 0$ for every $(i,j) \notin \mathcal{H}$. By contradiction, suppose $X^\sharp_{i,j} \neq 0$ for some $(i,j) \notin \mathcal{H}$. Note that, due to the positive definiteness of ${X^\sharp}^{-1}$, we have

$$(X^\sharp)^{-1}_{i,i} \times (X^\sharp)^{-1}_{j,j} - ((X^\sharp)^{-1}_{i,j})^2 > 0 \tag{42}$$

Now, based on Lemma 1, one can write

$$(X^\sharp)^{-1}_{ij} = C_{i,j} + 2\max_k\{C_{k,k}\} \times \operatorname{sign}(X^\sharp_{i,j}) \tag{43}$$

Considering the fact that $C \succeq 0$, we have $|C_{i,j}| \leq \max_k\{C_{k,k}\}$. Together with (43), this implies that $|(X^\sharp)^{-1}_{i,j}| \geq \max_k\{C_{k,k}\}$. Furthermore, due to Lemma 1, one can write $(X^\sharp)^{-1}_{i,i} = C_{i,i}$ and $(X^\sharp)^{-1}_{j,j} = C_{j,j}$. This leads to

$$(X^\sharp)^{-1}_{i,i} \times (X^\sharp)^{-1}_{j,j} - ((X^\sharp)^{-1}_{i,j})^2 = C_{i,i} \times C_{j,j} - (\max_k\{C_{k,k}))^2 \leq 0 \tag{44}$$

contradicting (42). Therefore, $X^\sharp$ is a feasible solution for the GGL. This implies that $f(X^*) = f(X^\sharp)$. Due to the uniqueness of the solution of (41), we have $X^* = X^\sharp$. Now, note that (41) can be reformulated as

$$\min_{X\in\mathbb{S}^n_+} \operatorname{tr} \tilde{C} D^{1/2} X D^{1/2} - \log\det X + \sum_{(i,j)\in\mathcal{H}} \lambda_{i,j}|X_{i,j}| + 2\max_k\{C_{k,k}\} \sum_{(i,j)\notin\mathcal{H}} |X_{i,j}| \tag{45}$$

Upon defining

$$\tilde{X} = D^{1/2} X D^{1/2} \tag{46}$$

and following some algebra, one can verify that 41 is equivalent to

$$\min_{\tilde{X}\in\mathbb{S}^n_+} \operatorname{tr} \tilde{C}\tilde{X} - \log\det \tilde{X} + \sum_{(i,j)\in\mathcal{H}} \tilde{\lambda}_{i,j}|\tilde{X}_{i,j}| + 2 \sum_{(i,j)\notin\mathcal{H}} |\tilde{X}_{i,j}| + \log\det(D) \tag{47}$$

Dropping the constant term in (47) gives rise to the optimization (39). Therefore, $\hat{X} = D^{-1/2}\tilde{X}D^{-1/2}$ holds in light of 46. □

Proof of Theorem 1.. Define $\tilde{C}_\lambda = D^{-1/2} C_\lambda D^{-1/2}$ and note that, due to Lemma 2, $\tilde{C}_\lambda$ and $\tilde{X}$ have the same signed sparsity pattern as C_λ and $\hat{X}$, respectively. Therefore, it suffices to show that the signed sparsity structures of $\tilde{C}_\lambda$ and $\tilde{X}$ are the same, which can be done by analyzing the optimization problem (39) and its connection to the GGL (explained in Lemma 2). Since (39) is the weighted analog of graphical lasso, the arguments made in the proof of Theorem 12 in [13] can be adopted to prove Theorem 1. The details are omitted for brevity. □

B. Proof of Theorem 2

Recall the definition of the cone $\mathcal{K}$ and its dual $\mathcal{K}_*$ as in 26. Being dual cones, $\mathcal{K}$ and $\mathcal{K}_*$ satisfy Farkas' lemma.

Lemma 3 (Farkas' lemma)**.** *Given an arbitrary $Y \in \mathbb{S}^n_V$*

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 $Y \bullet X < 0$.*

From the definition of $g(y)$ in (30) and the relations (29), 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 $P_{\tilde{\mathcal{G}}}(X^{-1}) = C - A(y)$, 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 $\mathbf{A} = [\operatorname{vec} A_1, \dots, \operatorname{vec} A_m]$. 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 $\frac{1}{2}n(n+1) \times |\tilde{\mathcal{G}}|$ matrix Q is the orthogonal basis matrix of $\mathbb{S}^n_{\tilde{\mathcal{G}}}$ in $\mathbb{S}^n$. Because of this, we see that the eigenvalues of the Hessian $\nabla^2 g(y)$ are bound

$$\lambda_{\min}(\mathbf{A}^T\mathbf{A})\lambda^2_{\min}(X^{-1}) \leq \lambda_i(\nabla^2 g(y)) \leq \lambda_{\max}(\mathbf{A}^T\mathbf{A})\lambda^2_{\max}(X^{-1}), \tag{48}$$

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

$$\operatorname{cond}(\nabla^2 g(y)) \leq \frac{\lambda_{\max}(\mathbf{A}^T\mathbf{A})}{\lambda_{\min}(\mathbf{A}^T\mathbf{A})} \left(\frac{\lambda_{\max}(X)}{\lambda_{\min}(X)}\right)^2. \tag{49}$$

Consequently most of our effort will be expended in bounding the eigenvalues of X.

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 make two observations. The first is that the sequence generated by Newton's method is monotonously decreasing, as in

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

Evaluating each $f_*(S)$ as $n + \log\det X$ yields our first key inequality

$$\log\det \hat{X} \leq \log\det X \leq \log\det X_0. \tag{50}$$

Our second observation is the fact that every Newton direction Δy points away from the boundary of $\operatorname{dom} g$.

Lemma 4 (Newton direction is positive definite)**.** *Given $y \in \operatorname{dom} g$, define the Newton direction $\Delta y = -\nabla^2 g(y)^{-1} \nabla g(y)$. Then, $\Delta S = -A(\Delta y) \in \mathcal{K}_*$.*

Proof. The Newton direction is explicitly written as

$$\Delta y = -[\mathbf{A}^T \nabla f_*(X)^{-1} \mathbf{A}]^{-1} \mathbf{A}^T \operatorname{vec} X$$

and $\Delta S = -A(\Delta y)$. For any $W \in \mathcal{K}$, we have by substituting

$$\begin{aligned} W \bullet \Delta S &= (\mathrm{vec}\, W)^T(\mathrm{vec}\, \Delta S) \\ &= (\mathrm{vec}\, W)^T(\mathbf{A}[\mathbf{A}^T\nabla^2 f(X)^{-1}\mathbf{A}]^{-1}\mathbf{A}^T)\mathrm{vec}\, X \\ &\geq c_1(\mathrm{vec}\, W)^T(\mathrm{vec}\, X) = c_1(W \bullet X) \\ &\geq c_2 \mathrm{tr}\, W > 0 \end{aligned}$$

where $c_1 = \sigma_{\min}(\mathbf{A}[\mathbf{A}^T\nabla^2 f(X)^{-1}\mathbf{A}]^{-1}\mathbf{A}^T) \geq \mathrm{cond}^{-1}(\mathbf{A}^T\mathbf{A})\lambda_{\max}^{-2}(X) > 0$ and $c_2 = \lambda_{\min}(X) > 0$. Since there can be no separating hyperplane $W \bullet \Delta S < 0$, we have $\Delta S \in \mathcal{K}_*$ by Farkas' lemma. $\square$

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

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

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 5. *Denote the n eigevalues of M as $\lambda_1 \geq \cdots \geq \lambda_n$. Then*

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

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

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

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

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

Lemma 6. *Define the initial suboptimality $\phi_{\max} = \log\det X_0 - \log\det \hat{X}$. Then we have*

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

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 (50):

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

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

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

Now, observe that we have arrived at y by taking Newton steps $y = y_0 + \sum_{j=1}^k \alpha_j \Delta y_j$, and that each Newton direction points away from the boundary of the feasible set, as in $-A(\Delta y) \in \mathcal{K}_*$ (See Lemma 4). Since $X \in \mathcal{K}$, we must always have

$$\begin{aligned} X \bullet A(y - y_0) &= \alpha_j \sum_{j=1}^k X \bullet A(\Delta y_j) \\ &= -\alpha_j \sum_{j=1}^k X \bullet \Delta S \leq 0. \end{aligned}$$

Substituting yields the second bound and applying (50) yields

$$\begin{aligned} \phi(X_0^{-1}X) &= X_0^{-1} \bullet X - \log\det X_0^{-1}X - n \\ &= (n + A(y - y_0) \bullet X - \log\det X_0^{-1}X) - n \\ &= \underbrace{\log\det X_0 X^{-1}}_{\leq \phi_{\max}} + \underbrace{A(y - y_0) \bullet X}_{\leq 0} \leq \phi_{\max}. \end{aligned}$$

$\square$

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 2. Here, we will prove

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

which yields the desired condition number bound on $\nabla^2 g(y)$ by substituting into (49). Writing $\lambda_1 = \lambda_{\max}(X)$ and $\lambda_n = \lambda_{\min}(X)$, we have from the two lemmas above:

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

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

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

Finally, bounding $\lambda_n/\lambda_1 \geq 0$ yields the desired bound. $\square$

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

[3] 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.

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

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

[6] 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.

[7] 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.

[8] 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.

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

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

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

[12] 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.

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

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

[15] Delbert Fulkerson and Oliver Gross. Incidence matrices and interval graphs. *Pacific journal of mathematics*, 15(3):835–855, 1965.

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

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

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

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

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

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

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

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

[28] 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.

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

[30] Patricia Menéndez, Yiannis AI Kourmpetis, Cajo JF ter Braak, and Fred A van Eeuwijk. Gene regulatory networks from multifactorial perturbations using graphical lasso: application to the dream4 challenge. *PloS one*, 5(12):e14147, 2010.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

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

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

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

[42] 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.

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

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

[45] 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.

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

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