An Efficient Algorithm for the Symmetric Principal Minor Assignment Problem

Justin Rising[a,*], Alex Kulesza[b], Ben Taskar[c]

[a] *Department of Statistics*
University of Pennsylvania, Philadelphia, PA 19104 USA
[b] *Computer Science and Engineering*
University of Michigan, Ann Arbor, MI 48109 USA
[c] *Computer Science and Engineering*
University of Washington, Seattle, WA 98195 USA

Abstract

We consider the problem of reconstructing a symmetric matrix from its principal minors, which has several applications in information theory and statistical modeling. We develop a theory of symmetric matrices with equal corresponding principal minors based on a simple equivalent property due to Oeding (2011). We then use this theory to provide a method for choosing a canonical representative from the class of all symmetric matrices with specified principal minors. Finally, we provide an efficient algorithm for computing this canonical representative given its principal minors as input.

Keywords: Principal minor assignment problem
2010 MSC: 15A15

1. Introduction

Given an $n \times n$ symmetric complex matrix $\boldsymbol{H}$, it is trivial but time consuming to compute all of its principal minors. In the inverse problem, known as the *symmetric principal minor assignment problem*, we are given an oracle that produces any requested principal minor of a symmetric matrix $\boldsymbol{H}$ in constant time, and we are asked to compute $\boldsymbol{H}$. At first glance, this inverse problem appears significantly more difficult. Our aim in this paper is to show that it is not: the reconstruction may be performed in polynomial time with a graph-search based algorithm.

The assignment problem has attracted some attention in recent years. Griffin and Tsatsomeros (2006b) proposed an algorithm which is guaranteed to work if the matrix to be reconstructed is *off-diagonal full*. Among other conditions, this requires that no off-diagonal entry be equal to zero. Furthermore, their algorithm has a running time of $O(n^5)$ and involves multiple matrix inversions, which raises concerns about its numerical stability. Holtz and

*Corresponding Author. Current address: @WalmartLabs, 850 Cherry Ave, San Bruno, CA 94066
Email addresses: jkrising@gmail.com (Justin Rising), kulesza@umich.edu (Alex Kulesza)

Sturmfels (2007) approached the problem algebraically and showed that a vector of length 2^n, assuming it strictly satisfies the Hadamard-Fischer inequalities, is the list of principal minors of some symmetric matrix if and only if it satisfies a certain system of polynomial equations. Oeding (2011) later proved a more general conjecture of Holtz and Sturmfels (2007), removing the Hadamard-Fischer assumption and set-theoretically characterizing the variety of principal minors of symmetric matrices. Oeding (2011) also observed that this characterization can in principle be used to solve the assignment problem by considering all principal minors of order at most three. However, it is not obvious that this approach can be used to construct a polynomial-time algorithm, since it involves maintaining an exponentially-sized set of candidate matrices. It also requires $O(n^3)$ queries to the principal minor oracle.

Our algorithm solves the symmetric principal minor assignment problem in worst-case $O(n^3)$ time, and it can be significantly faster for sparse matrices: if there are m nonzero entries in $\boldsymbol{H}$, then the running time is $O(n^2 + mn)$. The algorithm involves no matrix inversions. It requires $O(n^2)$ queries to the principal minor oracle; since $\Omega(n^2)$ oracle queries are required to determine the magnitudes of the off-diagonal elements, this is asymptotically optimal. A MATLAB implementation of our algorithm is available to download[1]. In deriving our algorithm, we also address the question of when two symmetric matrices have equal corresponding principal minors, providing elementary proofs for some of the results of Oeding (2011).

We stress that our algorithm does not guarantee that the oracle is consistent with the reconstructed matrix; rather, we guarantee that if the oracle is consistent with any matrix, it must be the matrix output by the algorithm. Actually verifying the consistency of the oracle requires generating every principal minor of the matrix, which inherently requires exponential time (but no more; see Griffin and Tsatsomeros (2006a)). However, in certain applications we can be sure that the oracle is consistent with some symmetric matrix, and so our algorithm is sufficient for these problems.

We see two primary applications of a solution to the symmetric principal minor assigment problem. The first is deciding whether a vector of length 2^n is the entropic vector of some multivariate Gaussian distribution. Any solution to this problem has potential applications in information theory and multivariate statistics (Hassibi and Shadbakht, 2007).

The second is estimating the parameters of a determinantal point process over a finite set (see Borodin (2011); Hough et al. (2006); Kulesza and Taskar (2012) and references therein). In brief, a determinantal point process Y is a random subset of $[n]$ characterized by a matrix $\boldsymbol{K}$ with the property that $\mathbb{P}(A \subseteq Y) = \det(\boldsymbol{K}_A)$ for all $A \subseteq [n]$. Not all determinantal point processes are characterized by symmetric matrices, but many interesting classes are, including the class of structured determinantal processes defined in Kulesza and Taskar (2012). Our algorithm allows us to reconstruct this matrix $\boldsymbol{K}$ given knowledge of its principal minors.

[1] `http://www.eecs.umich.edu/~kulesza/code/sym_assignment.tgz`

Example: 3×3 *Matrices*

Consider the problem of reconstructing a 3×3 matrix from its principal minors. Suppose that we are given an oracle for the principal minors of

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} .$$

We have $\det\left(\boldsymbol{H}_{\{i\}}\right) = h_{ii}$ for each i, so we can reconstruct the diagonal of $\boldsymbol{H}$ exactly with three queries to the oracle. We also have $\det\left(\boldsymbol{H}_{\{i,j\}}\right) = h_{ii}h_{jj} - h_{ij}^2$ for each $i \neq j$, so we can find the squares of the off-diagonal elements with an additional three queries. However, we get no information about the *signs* of these elements. We must therefore examine one more principal minor—the full determinant:

$$\det\left(\boldsymbol{H}\right) = h_{11}h_{22}h_{33} + 2h_{12}h_{13}h_{23} - h_{11}h_{23}^2 - h_{22}h_{13}^2 - h_{33}h_{12}^2 .$$

We can now infer the value of the product $h_{12}h_{13}h_{23}$; if it is nonzero, we know from its sign whether an even or odd number of the off-diagonal elements are negative. By inspection, any assignment of signs that preserves this parity gives the same set of principal minors, so we have fully characterized the desired set of matrices. On the other hand, if the product of the off-diagonal elements is zero, then any combination of signs is consistent with the determinant.

For $n \times n$ matrices, we can again use $n + n(n-1)/2$ queries to find the diagonal elements and the magnitudes of the off-diagonal elements. However, we need a general technique for assigning signs to the off-diagonal elements. It turns out that this can be achieved using a graphical representation of the matrix $\boldsymbol{H}$ in which the natural generalization of the three-element product is a simple chordless cycle. Our algorithm uses a systematic exploration of cycles on a spanning tree of this graph to infer the signs of off-diagonal elements. This framework not only allows us to reconstruct a matrix $\boldsymbol{H'}$ with the same principal minors as $\boldsymbol{H}$, but also to fully characterize the set of all such matrices.

The remainder of this paper is organized as follows. In Section 2, we introduce our notation and terminology. In Section 3, we describe a necessary and sufficient condition for two symmetric matrices to have equal corresponding principal minors, and we show how to choose a canonical representative from the set of matrices with given principal minors. In Section 4, we give a general algorithm for the symmetric principal minor assignment problem. Finally, we develop a more efficient variant of our algorithm in Section 5. Appendix A shows how our algorithm operates on an example 6×6 matrix.

2. Notation and Terminology

2.1. Matrices

We will use n to denote the number of rows of a square matrix, $[n]$ to denote the set $\{1, 2, \ldots, n\}$, and $S^n(\mathbb{C})$ to denote the set of $n \times n$ complex symmetric matrices. The principal

submatrix of $\boldsymbol{H} \in S^n(\mathbb{C})$ corresponding to $\alpha \subseteq [n]$ will be denoted $\boldsymbol{H}_\alpha$, and we observe the convention that $\det(\boldsymbol{H}_\emptyset) = 1$. All matrices are assumed to be symmetric unless otherwise stated.

We say that $\boldsymbol{H}$ and $\boldsymbol{K}$ are *determinantally compatible* if $h_{ii} = k_{ii}$ for all i and $|h_{ij}| = |k_{ij}|$ for all $i \neq j$. We further say that $\boldsymbol{H}$ and $\boldsymbol{K}$ are *determinantally equivalent*, and write $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$, if $\det(\boldsymbol{H}_\alpha) = \det(\boldsymbol{K}_\alpha)$ for all $\alpha \subseteq [n]$. Any pair of determinantally equivalent matrices are determinantally compatible, but the converse is not true.

We also say that two matrices $\boldsymbol{H}$ and $\boldsymbol{K}$ are $\boldsymbol{D}$*-similar*, and write $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$, if there is some diagonal matrix $\boldsymbol{D}$ with nonzero entries in $\{-1, 1\}$ such that $\boldsymbol{H} = \boldsymbol{D}\boldsymbol{K}\boldsymbol{D}^{-1}$. The set of all such $n \times n$ matrices will be denoted as $D^n_\pm$. We observe that $D^n_\pm$ is an Abelian group under matrix multiplication, so its actions on $S^n(\mathbb{C})$ correspond to symmetries: $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$ if and only if $\boldsymbol{H}$ is in the orbit of $\boldsymbol{K}$ under the action of $D^n_\pm$ on $S^n(\mathbb{C})$ defined by $\boldsymbol{D} \cdot \boldsymbol{K} = \boldsymbol{D}\boldsymbol{K}\boldsymbol{D}^{-1}$. This symmetry is easy to describe: it is simply invariance with respect to the direction of the coordinate axes. As above, any pair of $\boldsymbol{D}$-similar matrices are determinantally compatible, but the converse is not true.

2.2. Graphs

While we can describe our algorithms and results purely in terms of matrices, there is an equivalent, cleaner description in graph-theoretic terms. We will use n to denote the number of vertices in a graph G, and m to denote the number of edges. w_{ij} denotes the weight of the edge between vertex i and vertex j.

A rooted graph is a graph G in which some vertex r has been designated as the root. While this does not imply any special properties of G, many graphical algorithms, including ours, assume a designated root node.

A path P in a graph G is a sequence of vertices $\left\{v_{i_j}\right\}_{j=1}^{|P|}$ such that G contains the edges $\left(v_{i_j}, v_{i_{j+1}}\right)$ for $j = 1, \ldots, |P| - 1$. The set of vertices along a path P is denoted $\operatorname{supp}(P)$. A graph is connected if there is a path between any two vertices, and is otherwise partitioned into a set of connected subgraphs referred to as connected components.

A tree is a graph T such that there is a unique path between any two vertices of T. If G is a connected graph, a spanning tree T is a subgraph of G such that T is a tree, each vertex of G is present in T, and each edge of T is present in G. If G is not connected, we can define a spanning forest, which is the union of a collection of spanning trees for each connected component of G. If T is a spanning tree of a rooted graph, then the depth of a vertex v is the length of the path in T from the root to v.

If G contains a path P and also contains the edge $\left(v_{i_{|P|}}, v_{i_1}\right)$, then we say that G contains the cycle $C = P$. If the vertices in a cycle C are distinct, C is called a simple cycle. If there is an edge between two vertices of a cycle which is not contained in the cycle itself, this edge is referred to as a chord. A cycle with no chords is called chordless. If C does contain a chord e, then there are two cycles C_1 and C_2 such that $\operatorname{supp}(C_1) \cup \operatorname{supp}(C_2) = \operatorname{supp}(C)$ and $\operatorname{supp}(C_1) \cap \operatorname{supp}(C_2) = \operatorname{supp}(e)$. In this case, we say that e separates C into the subcycles C_1 and C_2.

Finally, we will denote the product of edge weights along a cycle C as

$$p(C) = w_{i_{|C|},i_1} \prod_{j=1}^{|C|-1} w_{i_j,i_{j+1}} \ .$$

2.3. Graphical representations of matrices

For any $\boldsymbol{H} \in S^n(\mathbb{C})$, we define the graph $G(\boldsymbol{H})$ to have vertex set $[n]$ and edge set $\{(i,j) : i \neq j \text{ and } h_{ij} \neq 0\}$. As observed above, the signs of $\boldsymbol{H}$ contain essential information about its determinants, so we give $G(\boldsymbol{H})$ edge weights $w_{ij} = \operatorname{sgn}(h_{ij})$. The subgraph of $G(\boldsymbol{H})$ induced by $\alpha \subseteq [n]$ is simply $G(\boldsymbol{K}_\alpha)$. We will use $\operatorname{comp}(\boldsymbol{H})$ to denote the number of connected components of $G(\boldsymbol{H})$.

For determinantally compatible matrices $\boldsymbol{H}$ and $\boldsymbol{K}$, the only possible differences are the signs of the off-diagonal elements. We define the graph $G(\boldsymbol{H},\boldsymbol{K})$ to have the shared vertex and edge sets of $G(\boldsymbol{H})$ and $G(\boldsymbol{K})$, and edge weights $w_{ij} = \operatorname{sgn}(h_{ij}k_{ij})$. As above, we will use $\operatorname{comp}(\boldsymbol{H},\boldsymbol{K})$ to denote the number of connected components of $G(\boldsymbol{H},\boldsymbol{K})$.

A graph coloring is a mapping from the vertices of graph G to a discrete set of colors obeying constraints induced by the edge weights. In traditional graph coloring problems, the vertices i and j must be assigned different colors whenever the edge (i,j) is present. In our case, we will be interested in a coloring c of $G(\boldsymbol{H},\boldsymbol{K})$ taking values in $\{-1,1\}$ such that $c(j) = w_{ij}c(i)$. If such a mapping exists, it will be referred to as a valid coloring of $G(\boldsymbol{H},\boldsymbol{K})$.

We will be interested in determining whether the graphs we consider have unique colorings, but by the above definition, this is never true: if c is a valid coloring for G, then so is $-c$. We define a rooted coloring to be a valid coloring c_r with the property that $c(r) = 1$ for the root r. Any connected rooted graph $G(\boldsymbol{H},\boldsymbol{K})$ with a valid coloring possesses a unique rooted coloring.

Finally, let $G_\pm$ be the set of graphs with edge weights in $\{-1,1\}$. For $\boldsymbol{D} \in D_\pm^n$ and $G \in G_\pm$, $\boldsymbol{D}\cdot G$ is the graph with the same vertices as G, but with edge weights $w'_{ij} = d_{ii}d_{jj}w_{ij}$, where w_{ij} are the edge weights in G. We say that G_1 and G_2 are $\boldsymbol{D}$-similar, $G_1 \overset{D}{\sim} G_2$, if and only if $G_1 = \boldsymbol{D}\cdot G_2$ for some $\boldsymbol{D} \in D_\pm$. By construction, $G(\boldsymbol{H}) \overset{D}{\sim} G(\boldsymbol{K})$ if and only if $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$.

3. The Theory of $\boldsymbol{D}$-Similarity

Oeding (2011) showed that two complex symmetric matrices are determinantally equivalent if and only if they are $\boldsymbol{D}$-similar. Therefore, if we want to understand the properties of matrices that are determinantally equivalent, we should study the properties of matrices which are $\boldsymbol{D}$-similar. In this section, we pursue this study and develop a theory of $\boldsymbol{D}$-similarity which is sufficient for our reconstruction. In the course of doing so, we will give an elementary combinatorial proof of Oeding's theorem and show how to choose a canonical representative from the set of matrices with given principal minors.

3.1. Preliminary Results

To start, we show in Lemmas 3.1 and 3.2 that without loss of generality we can assume that our graphs are connected and that our vertices are labeled in any convenient order. Theorem 3.3, which is an interesting result in its own right, will allow us to make arbitrary choices in our algorithms without worrying that they affect the correctness of the result.

Lemma 3.1. *Let $\boldsymbol{H}, \boldsymbol{K} \in S^n(\mathbb{C})$, let σ be a permutation of $[n]$, and let $\boldsymbol{\Sigma}$ be the corresponding permutation matrix. $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$ if and only if $\boldsymbol{\Sigma H \Sigma}^T \overset{\det}{\equiv} \boldsymbol{\Sigma K \Sigma}^T$.*

Proof. We will show that $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$ if and only if $\boldsymbol{\Sigma H \Sigma}^T \overset{\det}{\equiv} \boldsymbol{\Sigma K \Sigma}^T$. If $\boldsymbol{H} = \boldsymbol{D K D}^{-1}$, then $\boldsymbol{\Sigma H \Sigma}^T = (\boldsymbol{\Sigma D \Sigma}^T)(\boldsymbol{\Sigma K \Sigma}^T)(\boldsymbol{\Sigma D}^{-1}\boldsymbol{\Sigma}^T)$. This follows from the observations that $\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Sigma}^T$ and that $\boldsymbol{\Sigma D \Sigma}^T = \boldsymbol{\Sigma}^T \boldsymbol{D \Sigma}$. The proof of the converse is similar and is omitted. □

Lemma 3.2. *Let $\boldsymbol{H}, \boldsymbol{K} \in S^n(\mathbb{C})$. $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$ if and only if $\boldsymbol{H}_C \overset{\det}{\equiv} \boldsymbol{K}_C$ for every connected component C of $G(\boldsymbol{H}, \boldsymbol{K})$.*

Proof. As before, we will show that $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$ if and only if $\boldsymbol{H}_C \overset{D}{\sim} \boldsymbol{K}_C$ for every connected component C of $G(\boldsymbol{H}, \boldsymbol{K})$. By Lemma 3.1, we can assume without loss of generality that $\boldsymbol{H} = \operatorname{diag}\left(\{\boldsymbol{H}_{C_i}\}_{i=1}^{\operatorname{comp}(\boldsymbol{H},\boldsymbol{K})}\right)$ and $\boldsymbol{K} = \operatorname{diag}\left(\{\boldsymbol{K}_{C_i}\}_{i=1}^{\operatorname{comp}(\boldsymbol{H},\boldsymbol{K})}\right)$. We write $\boldsymbol{D} = \operatorname{diag}\left(\{\boldsymbol{D}_{C_i}\}_{i=1}^{\operatorname{comp}(\boldsymbol{H},\boldsymbol{K})}\right)$, and simply observe that $\boldsymbol{H} = \boldsymbol{D} \cdot \boldsymbol{K}$ if and only if $\boldsymbol{H}_{C_i} = \boldsymbol{D}_{C_i} \cdot \boldsymbol{K}_{C_i}$ for all i, since all entries outside of the blocks corresponding to the connected components are zero. □

Theorem 3.3. *Let $\boldsymbol{H} \in S^n(\mathbb{C})$ and $\boldsymbol{D}_1$, $\boldsymbol{D}_2 \in D_{\pm}^n$. $\boldsymbol{D}_1 \cdot \boldsymbol{H} = \boldsymbol{D}_2 \cdot \boldsymbol{H}$ if and only if $[\boldsymbol{D}_1]_C = \pm[\boldsymbol{D}_2]_C$ for every connected component C of $G(\boldsymbol{H})$.*

Proof. We first assume that $[\boldsymbol{D}_1]_C = \pm[\boldsymbol{D}_2]_C$ for every connected component C of $G(\boldsymbol{H})$. Here we can directly apply Lemma 3.2 to conclude that $\boldsymbol{D}_1 \cdot \boldsymbol{H} = \boldsymbol{D}_2 \cdot \boldsymbol{H}$.

We now assume that $\boldsymbol{D}_1 \cdot \boldsymbol{H} = \boldsymbol{D}_2 \cdot \boldsymbol{H}$ and that $G(\boldsymbol{H})$ has a single connected component. Let $\theta = [D_1]_{11}[D_2]_{11}$, and assume that $\boldsymbol{D}_1 \neq \theta \boldsymbol{D}_2$. Then there is some smallest index $b > 1$ such that $[D_1]_{bb} \neq \theta[D_2]_{bb}$. This implies that $[\boldsymbol{D}_1 \cdot \boldsymbol{H}]_{bj} \neq [\boldsymbol{D}_2 \cdot \boldsymbol{H}]_{bj}$ for any $j < b$, and so we have that $\boldsymbol{D}_1 \cdot \boldsymbol{H} \neq \boldsymbol{D}_2 \cdot \boldsymbol{H}$. This contradicts our hypothesis, and so we can conclude that $\boldsymbol{D}_1 = \theta \boldsymbol{D}_2$. By Lemma 3.2, the argument above applies to each connected component of $G(\boldsymbol{H})$, and we have the desired result. □

3.2. Algorithms

In this section, we give a polynomial time algorithm that correctly decides whether two determinantally compatible complex symmetric matrices are determinantally equivalent. We then show how to extend it to be completely constructive: for any determinantally compatible $\boldsymbol{H}, \boldsymbol{K} \in S^n(\mathbb{C})$, if $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$, we will construct a $\boldsymbol{D}$ such that $\boldsymbol{H} = \boldsymbol{D} \cdot \boldsymbol{K}$; and if $\boldsymbol{H} \overset{\det}{\not\equiv} \boldsymbol{K}$, we will construct a $\beta \subseteq [n]$ such that $\boldsymbol{H}_\beta \overset{\det}{\not\equiv} \boldsymbol{K}_\beta$ and $\boldsymbol{H}_\gamma \overset{\det}{\equiv} \boldsymbol{K}_\gamma$ for any γ strictly contained in β. Throughout this section we assume that any pair $\boldsymbol{H}$ and $\boldsymbol{K}$ are determinantally compatible and that $G(\boldsymbol{H}, \boldsymbol{K})$ is connected, with Lemma 3.2 as justification.

Algorithm 1 Given determinantally compatible $\boldsymbol{H}$ and $\boldsymbol{K}$, either produce a $\boldsymbol{D}$ such that $\boldsymbol{H} = \boldsymbol{D}\boldsymbol{K}\boldsymbol{D}^{-1}$ or determine that none exists

Let T be a spanning tree of $G(\boldsymbol{H}, \boldsymbol{K})$ with root 1
Produce a coloring c of T with $c(1) = 1$ and construct the corresponding $\boldsymbol{D}$
for each edge (i, j) not in T **do**
    Verify that $w_{ij} = d_{ii}d_{jj}$
end for

Recall that in our setting c is a valid coloring of $G(\boldsymbol{H}, \boldsymbol{K})$ if and only if $c(j) = w_{ij}c(i)$ for all i and j, and a valid rooted coloring if $c(r) = 1$ for a fixed root r. We refer to these constraints collectively as the coloring equations.

Lemma 3.4. *$\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$ if and only if $G(\boldsymbol{H}, \boldsymbol{K})$ possesses a valid coloring.*

Proof. $\boldsymbol{H} = \boldsymbol{D}\boldsymbol{K}\boldsymbol{D}^{-1}$ if and only if $h_{ij} = d_{ii}d_{jj}k_{ij}$ for all i and j. By the construction of $G(\boldsymbol{H}, \boldsymbol{K})$, $w_{ij} = \operatorname{sgn}(h_{ij}k_{ij})$; therefore $\boldsymbol{H} = \boldsymbol{D}\boldsymbol{K}\boldsymbol{D}^{-1}$ implies $w_{ij} = d_{ii}d_{jj}$, and in that case the coloring defined by $c(i) = d_{ii}$ is valid. Conversely, if we are given a valid coloring c of $G(\boldsymbol{H}, \boldsymbol{K})$, we can set $d_{ii} = c(i)$, and by the same logic $\boldsymbol{H} = \boldsymbol{D}\boldsymbol{K}\boldsymbol{D}^{-1}$. □

In light of Lemma 3.4, we can decide whether $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$ by determining whether the coloring equations for $G(\boldsymbol{H}, \boldsymbol{K})$ have a solution. Any algorithm that produces a solution to a system of linear equations may be used to find valid colorings of $G(\boldsymbol{H}, \boldsymbol{K})$, but the coloring equations are so sparse that a specialized algorithm can find the solution much more quickly. We begin our discussion of such an algorithm with a result on trees and colorability.

Lemma 3.5. *If $G(\boldsymbol{H}, \boldsymbol{K})$ is a tree, there is a unique valid rooted coloring of $G(\boldsymbol{H}, \boldsymbol{K})$ for any set of edge weights. If $G(\boldsymbol{H}, \boldsymbol{K})$ is not a tree, there is some set of edge weights for which no valid coloring is possible.*

Proof. First assume that $G(\boldsymbol{H}, \boldsymbol{K})$ is a tree; then we can permute the indices of $G(\boldsymbol{H}, \boldsymbol{K})$ so that the matrix corresponding to the coloring equations is $n \times n$ upper triangular with nonzero diagonal entries. Thus, we are guaranteed that a unique solution exists.

Now assume that $G(\boldsymbol{H}, \boldsymbol{K})$ is not a tree. In this case, there is a pair of vertices u and v such that there is a path p_1 from u to v, and a disjoint path p_2 from v to u. If we write w_p for the product of the edge weights along the path p, we have that $c(u) = c(u)w_{p_1}w_{p_2}$. We can always choose edge weights so that this condition is not satisfied. □

The proof of Lemma 3.5 gives us a relationship between the colorability of a cycle and the product of its edge weights. We record this result as Corollary 3.6.

Corollary 3.6. *A cycle C of $G(\boldsymbol{H}, \boldsymbol{K})$ possesses a valid coloring if and only if $p(C) = 1$.*

By Lemma 3.5, the coloring equations for any tree have a solution. Furthermore, they are sufficiently sparse that any of the standard graph search algorithms can be used to solve them with slight modifications. Therefore, our algorithm to find a valid coloring of an arbitrary

graph is simple: we will color some spanning tree, and verify that the coloring produced is valid for the entire graph. This idea is captured in Algorithm 1 and proved correct in Theorem 3.7.

Theorem 3.7. *Let $\boldsymbol{H}, \boldsymbol{K} \in S^n(\mathbb{C})$. Algorithm 1 produces a $\boldsymbol{D} \in D^n_{\pm}$ such that $\boldsymbol{H} = \boldsymbol{D}\boldsymbol{K}\boldsymbol{D}^{-1}$ if any exists, and otherwise determines that no such $\boldsymbol{D}$ exists. Furthermore, its running time is $O(m+n)$, where m and n are the number of edges and vertices in $G(\boldsymbol{H}, \boldsymbol{K})$, respectively.*

Proof. By Lemma 3.4, if $\boldsymbol{H} = \boldsymbol{D}\boldsymbol{K}\boldsymbol{D}^{-1}$ for some $\boldsymbol{D} \in D^n_{\pm}$, then this $\boldsymbol{D}$ corresponds to a valid coloring of $G(\boldsymbol{H}, \boldsymbol{K})$. As discussed above, Algorithm 1 finds a valid rooted coloring of the spanning tree T and verifies that it holds for the entire graph. By Theorem 3.3 and the constraint $c(1) = 1$, the $\boldsymbol{D}$ produced is irrespective of the choice of spanning tree. If $\boldsymbol{H} \overset{D}{\not\sim} \boldsymbol{K}$, then the coloring equations have no solution, and Algorithm 1 will correctly verify this. The running time analysis is identical to that of breadth first search, and is omitted. □

We now consider the problem of verifying that $\boldsymbol{H} \overset{\det}{\not\equiv} \boldsymbol{K}$. While it is sufficient to show that there is no $\boldsymbol{D}$ such that $\boldsymbol{H} = \boldsymbol{D} \cdot \boldsymbol{K}$, we can actually do more: in polynomial time, we can produce a $\beta \subseteq [n]$ such that $\boldsymbol{H}_\beta \overset{\det}{\not\equiv} \boldsymbol{K}_\beta$, but $\boldsymbol{H}_\gamma \overset{\det}{\equiv} \boldsymbol{K}_\gamma$ for every γ strictly contained in β. We will refer to such β as a *minimal counterexample*, and to any $\alpha \supseteq \beta$ as a *counterexample*. In order to do this efficiently, we must examine the structure of any minimal counterexample.

Lemma 3.8. *If $\boldsymbol{H} \overset{D}{\not\sim} \boldsymbol{K}$, any minimal counterexample must be the support of some simple chordless cycle C.*

Proof. By Lemma 3.5 and Corollary 3.6, any minimal counterexample must be the support of a cycle C such that $p(C) = -1$. Suppose that C has some chord (u, v) that separates C into C_1 and C_2, both subsets of C. Then $p(C) = p(C_1)p(C_2)$, and either $p(C_1) = -1$ or $p(C_2) = -1$; in either case we have a smaller counterexample by Corollary 3.6. Thus a counterexample can be minimal only if it is the support of a simple chordless cycle. □

The proof of Lemma 3.8 suggests an algorithm for finding a minimal counterexample when $\boldsymbol{H} \overset{D}{\not\sim} \boldsymbol{K}$. The procedure is outlined in Algorithm 2, and its correctness is recorded in Theorem 3.9.

Theorem 3.9. *Let $\boldsymbol{H}, \boldsymbol{K} \in S^n(\mathbb{C})$. If $\boldsymbol{H} \overset{D}{\not\sim} \boldsymbol{K}$, then Algorithm 2 correctly discovers a minimal counterexample. Its running time is $O(m)$, where m is the number of edges in $G(\boldsymbol{H}, \boldsymbol{K})$.*

Proof. In the worst case, Algorithm 2 removes a single edge at each iteration. The number of iterations is then $O(m)$, and the cost of discovering all the chords of a given cycle is $O(m)$, so the overall running time of the algorithm is $O(m)$. □

Algorithm 2 Given determinantally compatible $\boldsymbol{H}$ and $\boldsymbol{K}$ with $\boldsymbol{H} \overset{D}{\not\sim} \boldsymbol{K}$, find a minimal counterexample α

```
Run Algorithm 1 until a contradiction is discovered along edge e
Let C be the cycle consisting of e and the path from i to j in the spanning tree T
while C contains a chord e = (u, v) do
    Let C_1 and C_2 be the subcycles of C separated by e
    if p(C_1) = −1 then
        C ← C_1
    else
        C ← C_2
    end if
end while
return C
```

3.3. Simple Chordless Cycles and Determinantal Equivalence

From the discussion regarding Algorithm 2, it is clear that simple chordless cycles have a privileged role in the theory of $\boldsymbol{D}$-similarity. In this subsection, we will specify exactly how the simple chordless cycles of $G(\boldsymbol{H})$ and $G(\boldsymbol{K})$ are related when $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$.

We begin with two lemmas on $\boldsymbol{D}$-similarity of graphs.

Lemma 3.10. *For any $G \in G_{\pm}$, there is some $\boldsymbol{H} \in S^n(\mathbb{C})$ such that $G = G(\boldsymbol{H})$.*

Proof. Define $\boldsymbol{H}$ by $h_{ii} = 1$ for all i, $h_{ij} = 0$ if (i, j) is not an edge of G, and $h_{ij} = w_{ij}$ if (i, j) is an edge of G. Then $G = G(\boldsymbol{H})$ by construction. □

Lemma 3.11. *Let C_1 and C_2 be simple chordless cycles of length n with weights in $\{-1, 1\}$. $C_1 \overset{D}{\sim} C_2$ if and only if $p(C_1) = p(C_2)$.*

Proof. By Lemma 3.10, we can choose $\boldsymbol{H}$ and $\boldsymbol{K}$ such that $C_1 = G(\boldsymbol{H})$ and $C_2 = G(\boldsymbol{K})$. Then $C_1 \overset{D}{\sim} C_2$ if and only if $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$. By Lemma 3.4, $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$ if and only if $G(\boldsymbol{H}, \boldsymbol{K})$ possesses a valid coloring. By Corollary 3.6, $G(\boldsymbol{H}, \boldsymbol{K})$ possesses a valid coloring if and only if $p(G(\boldsymbol{H}, \boldsymbol{K})) = 1$. By construction, $p(G(\boldsymbol{H}, \boldsymbol{K})) = 1$ if and only if $p(C_1) = p(C_2)$. □

Given a determinantally compatible pair of matrices $(\boldsymbol{H}, \boldsymbol{K})$ and a cycle C contained in $G(\boldsymbol{H}, \boldsymbol{K})$, we write $C_{\boldsymbol{H}}$ to denote C with the edge weights inherited from $G(\boldsymbol{H})$, $C_{\boldsymbol{K}}$ to denote C with the edge weights inherited from $G(\boldsymbol{K})$, and $C_{\boldsymbol{H},\boldsymbol{K}}$ to denote C with edge weights inherited from $G(\boldsymbol{H}, \boldsymbol{K})$. With this notation, we can now state and prove a necessary and sufficient condition for $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$ in terms of the simple chordless cycles of $G(\boldsymbol{H}, \boldsymbol{K})$.

Theorem 3.12. *Let $\boldsymbol{H}$, $\boldsymbol{K} \in S^n(\mathbb{C})$ be determinantally compatible. Then $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$ if and only if $p(C_{\boldsymbol{H}}) = p(C_{\boldsymbol{K}})$ for every simple chordless cycle C of $G(\boldsymbol{H}, \boldsymbol{K})$.*

Proof. We first assume that $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$. If C is any simple chordless cycle of $G(\boldsymbol{H}, \boldsymbol{K})$, we have $C_{\boldsymbol{H}} \overset{D}{\sim} C_{\boldsymbol{K}}$. Therefore $p(C_{\boldsymbol{H}}) = p(C_{\boldsymbol{K}})$ by Lemma 3.11.

We now assume that $\boldsymbol{H} \overset{D}{\not\sim} \boldsymbol{K}$. Then there is some minimal counterexample α. By Lemma 3.8, $G(\boldsymbol{H}_\alpha, \boldsymbol{K}_\alpha)$ must be a simple chordless cycle C. Lemma 3.11 allows us to conclude that $p(C_H) \neq p(C_K)$. □

The matrices whose graphs are simple chordless cycles are known as cyclic tridiagonal matrices (Engeln-Müllges and Uhlig, 1996). The following lemma gives an explicit expression for the determinant of a cyclic tridiagonal matrix $\boldsymbol{T}$.

Lemma 3.13. *Let $\boldsymbol{T}$ be a cyclic tridiagonal matrix. Then*

$$\det(\boldsymbol{T}) = t_{nn} \det\left(\boldsymbol{T}_{[n-1]}\right) - t_{n-1,n}^2 \det\left(\boldsymbol{T}_{[n-2]}\right) - t_{1,n}^2 \det\left(\boldsymbol{T}_{[n-1]\setminus[1]}\right) + (-1)^{n+1} t_{1n} \prod_{i=1}^{n-1} t_{i,i+1}$$

Proof. The expression is obtained by the Laplace expansion of the determinant of $\boldsymbol{T}$. The details are omitted. □

With Theorem 3.12 and Lemma 3.13 in hand, we have sufficient machinery to provide a simple combinatorial proof of Oeding's theorem equating $\boldsymbol{D}$-similarity and determinantal equivalence.

Theorem 3.14 (Oeding (2011))**.** *For $\boldsymbol{H}, \boldsymbol{K} \in S^n(\mathbb{C})$, $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$ if and only if $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$.*

Proof. If $\boldsymbol{H}$ and $\boldsymbol{K}$ are not determinantally compatible, then $\boldsymbol{H} \overset{D}{\not\sim} \boldsymbol{K}$ and $\boldsymbol{H} \overset{\det}{\not\equiv} \boldsymbol{K}$. We therefore assume that $\boldsymbol{H}$ and $\boldsymbol{K}$ are determinantally compatible. We first assume that $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$. Then $\boldsymbol{H}$ is diagonally similar to $\boldsymbol{K}$, and as diagonally similar matrices have equal corresponding principal minors, we can conclude that $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$.

We now assume that $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$. If $G(\boldsymbol{H}, \boldsymbol{K})$ is a tree, we have that $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$ by Lemma 3.5. Otherwise, let C be an arbitrary simple chordless cycle of $G(\boldsymbol{H}, \boldsymbol{K})$. Determinantal equivalence implies, applying Lemma 3.13 to the matrices $\boldsymbol{H}_{\mathrm{supp}(C)}$ and $\boldsymbol{K}_{\mathrm{supp}(C)}$ and canceling equal terms, that

$$h_{|C|1} \prod_{i=1}^{|C|-1} h_{i,i+1} = k_{|C|1} \prod_{i=1}^{|C|-1} k_{i,i+1} \ .$$

We know that $|h_{ij}| = |k_{ij}|$ for all i and j, so $p(C_{\boldsymbol{H}}) = p(C_{\boldsymbol{K}})$. Since this holds for every simple chordles cycle in $G(\boldsymbol{H}, \boldsymbol{K})$, Theorem 3.12 allows us to conclude that $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$. □

Two comments are in order. First, as observed in Engel and Schneider (1980), the hypotheses of Theorem 3.14 can be weakened considerably. Our argument would still go through if we could merely assume that $\boldsymbol{H}$ and $\boldsymbol{K}$ are determinantally compatible and the principal minors corresponding to the cycles of $G(\boldsymbol{H}, \boldsymbol{K})$ are equal. In particular, for any

determinantally compatible $\boldsymbol{H}$ and $\boldsymbol{K}$ with no zeros off the diagonal, we have that $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$ if and only if all the corresponding 3×3 principal minors are equal.

Second, we note that it was shown in Oeding (2011) that any two determinantally compatible complex symmetric matrices are determinantally equivalent if and only if these 3×3 principal minors are equal. Our machinery does not seem to be strong enough to recover this more general result.

We close this section by counting the number of matrices with equal corresponding principal minors to some fixed $\boldsymbol{H}$. The following result is an easy corollary of Theorems 3.3 and 3.14.

Corollary 3.15. *For any fixed* $\boldsymbol{H}$, $\left|\left\{\boldsymbol{K} : \boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}\right\}\right| = 2^{n-\mathrm{comp}(\boldsymbol{H})}$.

3.4. Canonicalization

Given that there exist many matrices determinantally equivalent to a fixed $\boldsymbol{H}$, how can we pick a canonical representative from this set? In this subsection, we will show that choosing a spanning tree T of $G(\boldsymbol{H})$ and requiring it to have positive edge weights uniquely determines an element of this set. We will then describe a method of canonicalization that takes a deterministic spanning tree algorithm $\mathcal{A}$ and a matrix $\boldsymbol{H}$ and produces the canonical representative corresponding to the output of $\mathcal{A}$. We begin with two lemmas regarding positive edge weight spanning trees and $\boldsymbol{D}$-similarity.

Lemma 3.16. *Let* $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$, *and assume that there is a spanning tree* T *of* $G(\boldsymbol{H}, \boldsymbol{K})$ *with weights identically equal to one. Then* $\boldsymbol{H} = \boldsymbol{K}$.

Proof. By Theorem 3.12, we must have $p(C) = 1$ for every chordless cycle C of $G(\boldsymbol{H}, \boldsymbol{K})$. Every edge in T has weight one, and so it follows that every edge of $G(\boldsymbol{H}, \boldsymbol{K})$ has weight one. Therefore $\boldsymbol{H} = \boldsymbol{K}$. □

Lemma 3.17. *Assume that* $G(\boldsymbol{H})$ *is a tree, and let* $\boldsymbol{D}$ *be the matrix corresponding to the solution of the coloring equations for* $G(\boldsymbol{H})$. *If* $\boldsymbol{K} = \boldsymbol{D}^{-1}\boldsymbol{H}\boldsymbol{D}$, *then* $\operatorname{sgn}(k_{ij}) = 1$ *whenever* $i \neq j$ *and* $k_{ij} \neq 0$.

Proof. Choose an arbitrary node r as the root of T, and consider the subgraph consisting of the path from r to any leaf l. Without loss of generality we assume every edge is of the form $(u, u+1)$. Then $d_{11} = 1$ and $d_{ii} = \prod_{j=1}^{i-1} w_{j,j+1}$ for $i > 1$. We have $h_{ij} = d_{ii}d_{jj}k_{ij}$ for all i and j. If $j \neq i+1$, $h_{ij} = 0$, so $k_{ij} = 0$ as well. If $j = i+1$, $d_{ii}d_{jj} = \operatorname{sgn}(h_{ij})$, which implies that $\operatorname{sgn}(k_{ij}) = 1$ as claimed. Since l was chosen arbitrarily, this holds for every path from the root to a leaf, and so it holds for the entire tree. □

The canonicalization procedure now follows naturally. Given $\boldsymbol{H}$, we take T to be a spanning tree of $G(\boldsymbol{H})$ generated by a spanning tree algorithm $\mathcal{A}$. We let $\boldsymbol{D}$ be the matrix corresponding to a valid coloring of T, and we take the canonical representation of $\boldsymbol{H}$ to be $\boldsymbol{D}^{-1}\boldsymbol{H}\boldsymbol{D}$. Algorithm 3 is a restatement of this procedure, and is shown to be correct in Theorem 3.18.

Algorithm 3 Canonicalize $\boldsymbol{H}$ with respect to a spanning tree algorithm $\mathcal{A}$

function CANONICALIZE($\boldsymbol{H}$, $\mathcal{A}$)
  Run $\mathcal{A}$ on $G(\boldsymbol{H})$ to produce a spanning tree T
  Mark 1 as the root of T
  Produce a $\boldsymbol{D}$ corresponding to a rooted coloring for T
  return $\boldsymbol{D}^{-1}\boldsymbol{H}\boldsymbol{D}$
end function

Theorem 3.18. *Let $\boldsymbol{H}, \boldsymbol{K} \in S^n(\mathbb{C})$, let $\mathcal{A}$ be a spanning tree algorithm, and let $\boldsymbol{H}_{\mathcal{A}}$ denote the output of Algorithm 3 with inputs $\boldsymbol{H}$ and $\mathcal{A}$. Then $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$ if and only if $\boldsymbol{H}_{\mathcal{A}} = \boldsymbol{K}_{\mathcal{A}}$.*

Proof. We first assume that $\boldsymbol{H}_{\mathcal{A}} = \boldsymbol{K}_{\mathcal{A}}$. $\boldsymbol{H}_{\mathcal{A}} \overset{D}{\sim} \boldsymbol{H}$ and $\boldsymbol{K}_{\mathcal{A}} \overset{D}{\sim} \boldsymbol{K}$, so it follows that $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$. By Theorem 3.14, we have that $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$.

We now assume that $\boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}$. By Theorem 3.14, $\boldsymbol{H} \overset{D}{\sim} \boldsymbol{K}$. This implies that $\boldsymbol{H}_{\mathcal{A}} \overset{D}{\sim} \boldsymbol{K}_{\mathcal{A}}$, and the weights of every edge of $T_{\boldsymbol{H}_{\mathcal{A}}}$ and $T_{\boldsymbol{K}_{\mathcal{A}}}$ are identically one by Lemma 3.17. We can therefore apply Lemma 3.16 to conclude that $\boldsymbol{H}_{\mathcal{A}} = \boldsymbol{K}_{\mathcal{A}}$. □

4. Solving the Symmetric Principal Minor Assignment Problem

In Section 3 we characterized the set $\{\boldsymbol{K} : \boldsymbol{H} \overset{\det}{\equiv} \boldsymbol{K}\}$ for any symmetric matrix $\boldsymbol{H}$ and described how to pick a canonical representative from this set. In this section, we give a naive algorithm that will reconstruct this canonical representative given a constant-time oracle for the principal minors of a symmetric matrix $\boldsymbol{H}$.

As before, chordless cycles will play a prominent role in our analysis. We start by showing that knowing the determinant of a cyclic tridiagonal matrix, its diagonal entries, and all but one of the off-diagonal entries is sufficient to infer the final entry.

Lemma 4.1. *Let $\boldsymbol{H}$ and $\boldsymbol{K}$ be determinantally compatible cyclic tridiagonal matrices where $h_{i,i+1} = k_{i,i+1}$ for all i between 1 and $n-1$. Then $h_{1n} = k_{1n}$.*

Proof. This follows immediately from Lemma 3.13. □

We now consider the problem of reconstructing a cyclic tridiagonal matrix $\boldsymbol{T}$ from its principal minors. As always, we can infer the diagonal entries from the one-element principal minors, and the magnitudes of the off-diagonal entries from the two-element principal minors. If we construct a spanning tree of $G(\boldsymbol{T})$, then there is exactly one edge whose sign is unknown. This satisfies the hypotheses of Lemma 4.1, and so we may infer the unknown sign. We refer to this process of inferring the sign of an edge as marking, and every edge not in T starts unmarked.

We move on from this simple case to reconstructing a matrix $\boldsymbol{H}$ such that $G(\boldsymbol{H})$ consists of a simple cycle C with at least one chord (u, v). In this case we can use Algorithm 4 to reconstruct $\boldsymbol{H}$. We will prove that this procedure is correct in Theorem 4.2.

Algorithm 4 Given a cycle C with exactly one unmarked edge e, infer the sign of e

procedure MARKCYCLE(C, e)
  if C contains a chord e' **then**
    Let C_1 be the subcycle of C not containing e, and C_2 the other subcycle
    if e' is unmarked **then**
      MARKCYCLE(C_1, e')
    end if
    MARKCYCLE(C_2, e)
  else
    Query the principal minor oracle for the determinant of the submatrix corresponing to C_1 and use Lemma 3.13 to determine the sign of e.
  end if
end procedure

Theorem 4.2. *Let $\boldsymbol{H} \in S^n(\mathbb{C})$ be such that $G(\boldsymbol{H})$ consists of a simple cycle with any number of chords. Given the diagonal entries of $\boldsymbol{H}$, the magnitudes of the off-diagonal entries, and a spanning tree T of $G(\boldsymbol{H})$ with each edge marked positive, Algorithm 4 will correctly infer the signs of the entries corresponding to the unmarked edges.*

Proof. The proof is by induction on the number of edges in $\{e \in G(\boldsymbol{H}) : e \notin T\}$, i.e., the unmarked edges. If there is exactly one unmarked edge e, then the sign may be inferred as argued above. We now assume that the algorithm works for any cycle with up to k unmarked edges, and we consider its operation on a cycle with $k+1$ unmarked edges.

Suppose that Algorithm 4 never chooses an unmarked chord e'; then the algorithm is singly recursive, and eventually C_2 will contain at most k unmarked edges and by induction the result will be correct. Otherwise, at some point e' will be unmarked. In this case C_1 contains at most k unmarked edges, so by induction it is marked correctly. Once e' is marked, C_2 contains at most k unmarked edges, so it is also marked correctly by induction. At this point the entire cycle is marked correctly. □

We finally consider the general symmetric principal minor assignment problem. In this case, the structure of $G(\boldsymbol{H})$ is arbitrary. Here we may use Algorithm 5 to perform the reconstruction. We will prove in Theorem 4.3 that this procedure is correct.

Theorem 4.3. *Given a principal minor oracle for $\boldsymbol{H} \in S^n(\mathbb{C})$ and a deterministic spanning tree algorithm $\mathcal{A}$, Algorithm 5 will correctly reconstruct the canonicalization $\boldsymbol{H}_{\mathcal{A}}$. Its running time is $O(m^2+n^2)$, where m and n are the numbers of edges and vertices in $G(\boldsymbol{H})$, respectively.*

Proof. Let $\tilde{\boldsymbol{H}}_{\mathcal{A}}$ denote the output of Algorithm 5 when given the oracle for $\boldsymbol{H}$ and $\mathcal{A}$ as input. By construction, $p\left(C_{\tilde{\boldsymbol{H}}_{\mathcal{A}}}\right) = p\left(C_{\boldsymbol{H}_{\mathcal{A}}}\right)$ for every simple chordless cycle C of $G\left(\tilde{\boldsymbol{H}}_{\mathcal{A}}, \boldsymbol{H}_{\mathcal{A}}\right)$. Therefore $\tilde{\boldsymbol{H}}_{\mathcal{A}} \overset{D}{\sim} \boldsymbol{H}_{\mathcal{A}}$ by Theorem 3.12, and $\tilde{\boldsymbol{H}}_{\mathcal{A}} \overset{\det}{\equiv} \boldsymbol{H}_{\mathcal{A}}$ by Theorem 3.14. $\tilde{\boldsymbol{H}}_{\mathcal{A}}$ and $\boldsymbol{H}_{\mathcal{A}}$ agree on the spanning tree T, so by Lemma 3.16, we can conclude that $\tilde{\boldsymbol{H}}_{\mathcal{A}} = \boldsymbol{H}_{\mathcal{A}}$.

Algorithm 5 Given a principal minor oracle for $\boldsymbol{H}$ and a deterministic spanning tree algorithm $\mathcal{A}$, output the canonicalization $\boldsymbol{H}_{\mathcal{A}}$

Use the first and second order principal minors of $\boldsymbol{H}$ to infer the diagonal elements and the magnitudes of the off-diagonal elements
Let T be a spanning tree of $G(\boldsymbol{H})$ generated by $\mathcal{A}$, and mark every edge in T as positive
while $G(\boldsymbol{H})$ contains an unmarked edge (i, j) **do**
    Let C be the cycle consisting of (i, j) and the path from i to j in T
    MarkCycle(C, (i, j))
end while

Computing the diagonal entries and the magnitudes of the off-diagonal entries is $O(n^2)$. The running time of Algorithm 4 is identical to that of Algorithm 2, and it must be called $O(m)$ times, so the overall running time of Algorithm 5 is $O(m^2 + n^2)$. □

We note that we have additionally produced a method for computing the determinant of a matrix $\boldsymbol{H}$ given its diagonal entries, the magnitude of its off-diagonal entries, and the value of $p(C)$ for every simple chordless cycle C of $G(\boldsymbol{H})$. In this case we can modify Algorithm 5 to assign signs to the entries of $\boldsymbol{H}$ consistent with the given sign products. From there, we may compute the determinant directly.

5. An Improved Algorithm

Algorithm 5 constructs the canonical matrix $\boldsymbol{H}_{\mathcal{A}}$ for an arbitrary spanning tree algorithm $\mathcal{A}$. However, a faster construction is possible when $\mathcal{A}$ adopts a breadth-first search (BFS) strategy. This is because BFS trees have the guarantee that the endpoints of non-tree edges are never ancestors, and differ in depth by at most one. We can exploit these facts to find chordless cycles in $G(\boldsymbol{H})$ efficiently, avoiding the potentially costly recursion into Algorithm 4.

Theorem 5.1. *Given a graph G on n nodes, a corresponding BFS tree T, and a pair of vertices i and j, we can construct in $O(n)$ time a simple path from i to j that has no chords.*

Proof. Let k be the lowest common ancestor (LCA) of i and j in T. (We can find k in $O(n)$ time using standard techniques.) Let P_{ik} denote the simple path obtained by iteratively following parent pointers from i until reaching k, and similarly for P_{jk}. Since T is a BFS tree, there are no non-tree edges between ancestors, and therefore P_{ik} and P_{jk} are chordless. The concatenation of these two paths (which intersect only at k, since k is the LCA) is thus a simple path from i to j that is chordless unless there exists an edge between a vertex on P_{ik} and a vertex on P_{jk}.

Consider an arbitrary vertex i' on P_{ik}. Because T is a BFS tree, non-tree edges always connect vertices whose depths differ by at most one, and, by construction, P_{jk} contains at most one vertex at any depth. This means that there are at most three vertices on P_{jk} that could be connected by an edge to i'. We can therefore check for chords to a given i' in constant time, or to all vertices on P_{ik} in $O(n)$ time.

Algorithm 6 Given a principal minor oracle for $\boldsymbol{H}$, output the canonicalization $\boldsymbol{H}_{\text{BFS}}$

Use the first and second order principal minors of $\boldsymbol{H}$ to infer the diagonal elements and the magnitudes of the off-diagonal elements
Let T be a BFS spanning tree of $G(\boldsymbol{H})$, and mark every edge in T as positive
Sort the non-tree edges in $G(\boldsymbol{H})$ using the partial ordering $(i,j) < (i',j')$ whenever $\text{depth}_T(i) < \text{depth}_T(i')$ or $(\text{depth}_T(i) = \text{depth}_T(i')$ and $\text{depth}_T(j) < \text{depth}_T(j'))$
for each non-tree edge (i,j) in sorted order **do**
    Compute a simple chordless path from i to j in $G(\boldsymbol{H}) - (i,j)$ using Theorem 5.1
    Call the oracle on the result (which is a simple chordless cycle in G) to mark (i,j)
end for

If we perform this search from the bottom up, then if we detect a chord (i',j'), we will be guaranteed that no other chord exists between $P_{ii'}$ and $P_{jj'}$; thus their concatenation will be a simple chordless path between i and j. □

Algorithm 6 shows how to use the results of Theorem 5.1 to implement the canonicalization procedure.

Theorem 5.2. *Given a principal minor oracle for $\boldsymbol{H} \in S^n(\mathbb{C})$, Algorithm 6 correctly reconstructs a matrix $\boldsymbol{H}_{\text{BFS}} \overset{D}{\sim} \boldsymbol{H}$. Its running time is $O(n^2 + mn)$, where m and n are the numbers of edges and vertices in $G(\boldsymbol{H})$, respectively.*

Proof. We address the running time first. Setting the diagonal elements and off-diagonal magnitudes requires $O(n^2)$ time, and constructing a BFS tree takes $O(m+n)$ time. The sort operation is $O(m\log(m))$, although sorting is not actually required in practice if the edges are simply expanded in BFS order. The for loop executes less than m times, and each iteration requires a call to the $O(n)$ path-finding procedure. Thus the total runtime of Algorithm 6 is $O(n^2+mn)$. If $G(\boldsymbol{H})$ is connected, then $m \geq n-1$, and the running time is simply $O(mn)$.

We now address correctness. By Theorem 3.18, there exists a canonicalization $\boldsymbol{H}_{\text{BFS}} \overset{D}{\sim} \boldsymbol{H}$ corresponding to the BFS tree found in the second step of Algorithm 6; to find it, we must successfully mark each edge in $G(\boldsymbol{H})$. The tree edges are marked immediately. Appealing to Lemma 4.1, the marking of a non-tree edge will be successful if the simple chordless path found in the first step of the for loop is completely marked, making (i,j) the only unmarked edge in the cycle passed to the oracle.

To see why this is always true, note that the path found using Theorem 5.1 has at most one non-tree edge (i',j'); since tree edges are marked, this is the only edge that could possibly be unmarked. By construction, $\text{depth}_T(i') \leq \text{depth}_T(i)$ and $\text{depth}_T(j') \leq \text{depth}_T(j)$, and since the edge (i,j) is removed from the graph, at least one of the two inequalities must hold strictly. Thus the edge (i',j') occurs before (i,j) in the sorted list, and has already been marked. □

A MATLAB implementation of Algorithm 6 (with some optimizations) is available from `http://www.eecs.umich.edu/~kulesza/code/sym_assignment.tgz`.

References

Borodin, A., 2011. Determinantal Point Processes. In: Akemann, G., Baik, J., Francesco, P. D. (Eds.), The Oxford Handbook of Random Matrix Theory. Oxford University Press, Ch. 11, pp. 231–249, arXiv:0911.1153.

Engel, G. M., Schneider, H., 1980. Matrices Diagonally Similar to a Symmetric Matrix. Linear Algebra and its Applications 29, 131–138.
URL http://dx.doi.org/10.1016/0024-3795(80)90234-7

Engeln-Müllges, G., Uhlig, F., 1996. Numerical Algorithms with C. Springer.

Griffin, K., Tsatsomeros, M. J., 2006a. Principal Minors, Part I: A Method for Computing All the Principal Minors of a Matrix. Linear Algebra and its Applications 419, 107–124.
URL http://dx.doi.org/10.1016/j.laa.2006.04.008

Griffin, K., Tsatsomeros, M. J., 2006b. Principal Minors, Part II: The Principal Minor Assignment Problem. Linear Algebra and its Applications 419, 125–171.
URL http://dx.doi.org/10.1016/j.laa.2006.04.009

Hassibi, B., Shadbakht, S., 2007. On a Construction of Entropic Vectors Using Lattice-Generated Distributions. In: International Symposium on Information Theory. pp. 501–505.
URL http://dx.doi.org/10.1109/ISIT.2007.4557096

Holtz, O., Sturmfels, B., 2007. Hyperdeterminantal Relations among Symmetric Principal Minors. Journal of Algebra 316, 634–648.
URL http://dx.doi.org/10.1016/j.jalgebra.2007.01.039

Hough, J. B., Krishnapur, M., Peres, Y., Virág, B., 2006. Determinantal Processes and Independence. Probability Surveys 3, 206–229.
URL http://dx.doi.org/10.1214/154957806000000078

Kulesza, A., Taskar, B., 2012. Determinantal Point Processes for Machine Learning. Foundations and Trends in Machine Learning 5 (2–3), 123–286, arXiv:1207.6083.
URL http://dx.doi.org/10.1561/2200000044

Oeding, L., 2011. Set Theoretic Defining Equations of the Variety of Principal Minors of Symmetric Matrices. Algebra and Number Theory 5 (1), 75–109, arXiv:0809.4236.
URL http://dx.doi.org/10.2140/ant.2011.5.75

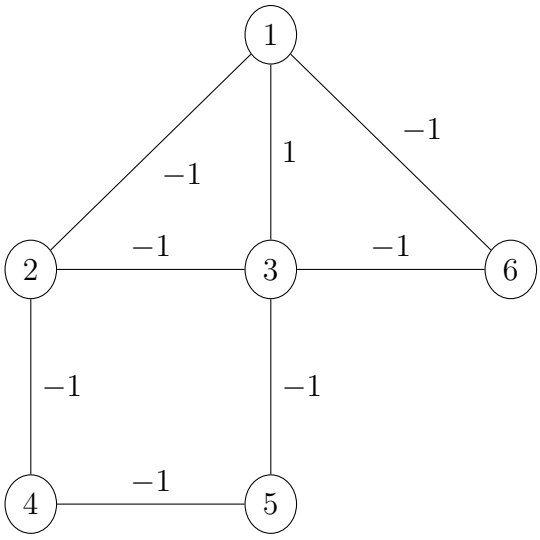

Figure A.1: The graph $G(\boldsymbol{H})$ of the matrix to be reconstructed.

Appendix A. Reconstruction of a Small Matrix

In this appendix, we demonstrate our reconstruction algorithm on the 6×6 matrix

$$\boldsymbol{H} = \begin{bmatrix} 3 & -1 & 1 & 0 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 & 0 \\ 1 & -1 & 3 & 0 & -1 & -1 \\ 0 & -1 & 0 & 3 & -1 & 0 \\ 0 & 0 & -1 & -1 & 3 & 0 \\ -1 & 0 & -1 & 0 & 0 & 3 \end{bmatrix}.$$

The corresponding graph $G(\boldsymbol{H})$ is shown in Figure A.1.

Our first step is to use the first order principal minors to read off the diagonal elements of $\boldsymbol{H}$ and the second order principal minors to read off the magnitudes of the off-diagonal elements. This yields the matrix

$$\tilde{\boldsymbol{H}} = \begin{bmatrix} 3 & 1 & 1 & 0 & 0 & 1 \\ 1 & 3 & 1 & 1 & 0 & 0 \\ 1 & 1 & 3 & 0 & 1 & 1 \\ 0 & 1 & 0 & 3 & 1 & 0 \\ 0 & 0 & 1 & 1 & 3 & 0 \\ 1 & 0 & 1 & 0 & 0 & 3 \end{bmatrix}.$$

We must now infer the signs of the off-diagonal elements. We begin by constructing a marked spanning tree T of $G(\tilde{\boldsymbol{H}})$, shown in Figure A.2(a). There are three edges that must be marked: $(2,3), (4,5)$ and $(3,6)$. Algorithm 6 selects umarked edges in an order consistent with the lexicographic ordering on the depths of their endpoints. As a result, we must mark the edges in the order $(2,3), (3,6), (4,5)$.

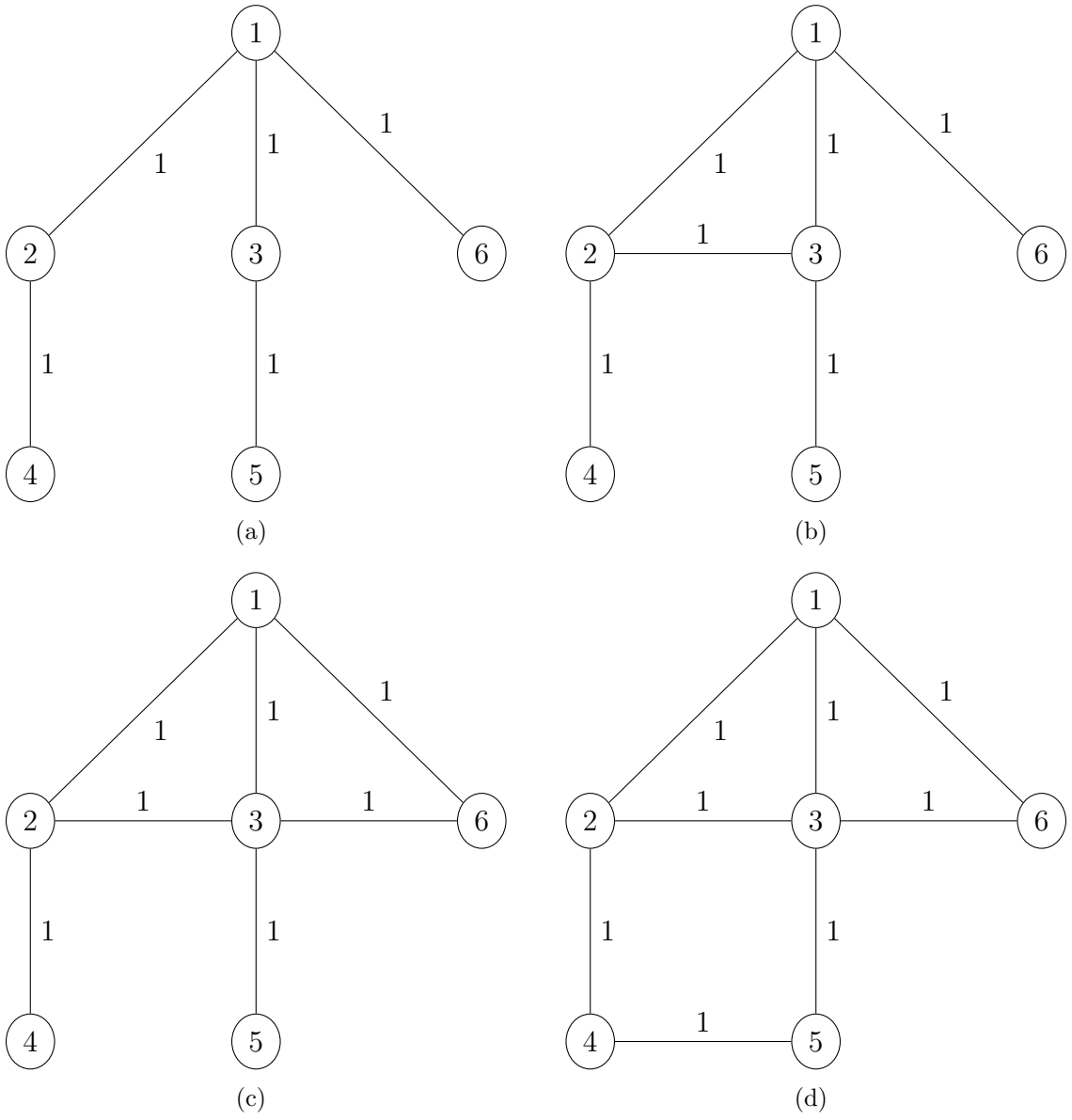

Figure A.2: The graph $G(\tilde{\boldsymbol{H}})$ constructed by Algorithm 6: (a) the initial spanning tree of $G(\tilde{\boldsymbol{H}})$; (b) the graph after the edge $(2, 3)$ is marked; (c) the graph after the edge $(3, 6)$ is marked; and, (d) the final reconstructed graph.

We begin by marking $(2, 3)$. The cycle $1, 2, 3$ has no chords, so we query the oracle for the value of $\det\left(\boldsymbol{H}_{\{1,2,3\}}\right)$. The result is 20, which implies that $\tilde{h}_{23} > 0$. We record this to the graph $G(\tilde{\boldsymbol{H}})$, which is shown in Figure A.2(b).

The second edge we mark is $(3, 6)$. This is contained in the simple chordless cycle $1, 3, 6$, so we query the oracle for $\det\left(\boldsymbol{H}_{\{1,3,6\}}\right)$. This is also equal to 20, implying that $\tilde{h}_{36} > 0$. We record this to the graph, whose state after this step is shown in Figure A.2(c).

Finally, we mark the edge $(4, 5)$. The relevant chordless cycle is $2, 3, 5, 4$. We query the oracle to find that $\det\left(\boldsymbol{H}_{\{2,3,4,5\}}\right) = 45$, and this implies that $\tilde{h}_{45} > 0$. We record this as the final edge in the graph, and we have completed the reconstruction. The final graph is shown in Figure A.2(d).

Of course, in general some edges will be marked as negative; however, as the reader may see from the structure of this example, when the matrix to be reconstructed is $\boldsymbol{D}$-similar to an entrywise nonnegative matrix $\boldsymbol{H}_+$, then Algorithm 6 will return $\boldsymbol{H}_+$.