EIGENVALUE LOCATION IN COGRAPHS

DAVID P. JACOBS, VILMAR TREVISAN, AND FERNANDO COLMAN TURA

Abstract. We give an $O(n)$ time and space algorithm for constructing a diagonal matrix congruent to $A + xI$, where $A$ is the adjacency matrix of a cograph and $x \in \mathbb{R}$. Applications include determining the number of eigenvalues of a cograph’s adjacency matrix that lie in any interval, obtaining a formula for the inertia of a cograph, and exhibiting infinitely many pairs of equienergetic cographs with integer energy.

1. Introduction

Let $G = (V, E)$ be an undirected graph with vertex set $V$ and edge set $E$. For $v \in V$, $N(v)$ denotes the open neighborhood of $v$, that is, $\{w|\{v, w\} \in E\}$. The closed neighborhood $N[v] = N(v) \cup \{v\}$. If $|V| = n$, the adjacency matrix $A = [a_{ij}]$ is the $n \times n$ matrix of zeros and ones such that $a_{ij} = 1$ if and only if $v_i$ is adjacent to $v_j$ (that is, there is an edge between $v_i$ and $v_j$). A value $\lambda$ is an eigenvalue if $\det(A - \lambda I) = 0$, and since $A$ is real symmetric its eigenvalues are real. In this paper, a graph’s eigenvalues are the eigenvalues of its adjacency matrix.

This paper is concerned with cographs. This class of graphs has been discovered independently by several authors in many equivalent ways since the 1970’s. Corneil, Lerchs and Burlingham [6] define cographs recursively:

1. A graph on a single vertex is a cograph;
2. A finite union of cographs is a cograph;
3. The complement of a cograph is a cograph.

A graph is a cograph if and only it has no induced path of length four [6]. They are often simply called $P_4$ free graphs in the literature. Linear time algorithms for recognizing cographs are given in [7] and more recently in [10].

While recognition algorithms for cographs is an interesting problem, our motivation for considering cographs comes from spectral graph theory [4, 8]. Spectral properties of cographs were studied by Royle in [16] where the surprising result was obtained that the rank of a cograph is the number of non-zero rows in the adjacency matrix. An elementary proof of this property was later given in [5]. More recently, in [2] Bıyıköğlu, Simić and Stanić obtained the multiplicity of $-1$ and $0$ for cographs.

The purpose of this paper is to extend to cographs eigenvalue location algorithms that exist for trees [11], threshold graphs [12] and generalized lollipop graphs [9]. Recall that two real symmetric matrices $R$ and $S$ are congruent if there exists a nonsingular matrix $P$ for which $R = P^TSP$. Our main focus is an algorithm that uses $O(n)$ time and space for constructing a diagonal matrix congruent to $A + xI$, where $A$ is adjacency matrix of a cograph, and $x \in \mathbb{R}$. Our paper is similar in spirit to the papers [11, 12] which describe $O(n)$ diagonalization algorithms for trees and for threshold graphs. Threshold

1991 Mathematics Subject Classification. 05C50, 05C85, 15A18.
Key words and phrases. cograph, adjacency matrix, eigenvalue.
Work supported by Science without Borders CNPq - Grant 400122/2014-6, Brazil.
graphs are $P_4$, $C_4$, and $2K_2$ free, and therefore are a subclass of cographs. Hence our algorithm is an extension of the algorithm in [12].

Several points are worth noting. First, while one might expect linear time algorithms for graphs with sparse adjacency matrices such as trees, the adjacency matrix of a cograph can be dense. Next, while our algorithm’s correctness is based on elementary matrix operations, its implementation operates directly on the cotree and uses only $O(n)$ space. Finally, the analysis of algorithms for trees and threshold graphs has led to interesting theoretical results. For example, in [15] conditions were determined for the index (largest eigenvalue) in trees to be integer. In [12] the authors showed that all eigenvalues of threshold graphs, except $-1$ and $0$, are simple. In [13] the algorithm was used to show that no threshold graphs have eigenvalues in $(-1, 0)$.

If $G$ is a graph having eigenvalues $\lambda_1, \ldots, \lambda_n$, its energy, denoted $E(G)$ is defined to be $\sum_{i=1}^{n} |\lambda_i|$. Two non-cospectral graphs with the same energy are called equienergetic. Finding non-cospectral equienergetic graphs is a relevant problem. In [13] the authors presented infinite sequences of connected, equienergetic pairs of non-cospectral threshold graphs with integer energy. In this paper, we continue this investigation.

Here is an outline of the remainder of this paper. In Section 2 we describe cotrees, and present some known facts. In Section 3 we give the elementary matrix operations used in our algorithm. In Section 4 we give the complete diagonalization algorithm. In Section 5 using Sylvester’s Law of Inertia, we show how to efficiently determine how many eigenvalues of a cograph lie in a given interval. The inertia of a graph $G$ is the triple $(n_+, n_0, n_-)$ giving the number of eigenvalues of $G$ that are positive, zero, and negative, and in Section 6 we give a formula for cograph inertia. Finally in Section 7 we exhibit infinitely many non-threshold cographs equienergetic to a complete graph.

2. Cotrees and adjacency matrix

Cographs have been represented in various ways, and it is useful to recall the representation given in [6]. The unique normalizend form of a cograph $G$ is defined recursively: If $G$ is connected, then it is in normalized form if it is expressed as a single vertex, or the complemented union of $k \geq 2$

$$G = G_1 \cup G_2 \cup \ldots \cup G_k$$

connected cographs $G_i$ in normalized form. If $G$ disconnected its normalized form is the complement of a connected cograph in normalized form. The unique rooted tree $T_G$ representing the parse structure of the cograph’s normalized form is called the cotree. The leaves or terminal vertices of $T_G$ correspond to vertices in the cograph. The interior nodes represent $\cup$ operations.

It is not difficult to show that the class of cographs is also the smallest class of graphs containing $K_1$, and closed under the union $\cup$ and join $\otimes$ operators. In fact one can transform the cotree of Corneil, Lerchs and Burlingham into an equivalent tree $T_G$ using $\cup$ and $\otimes$. In the connected case, we simply place a $\otimes$ at the tree’s root, placing $\cup$ on interior nodes with odd depth, and placing $\otimes$ on interior nodes with even depth. To build a cotree for a disconnected cograph, we place $\cup$ at the root, and place $\otimes$’s at odd depths, and $\cup$’s at even depths. It will be convenient for us to use this unique alternating representation. In [2] this structure is called a minimal cotree, but throughout this paper we call it simply a cotree. All interior nodes of cotrees have at least two children. Figure 1 shows a cograph and cotree. The following is well known.
Lemma 1. If $G$ is a cograph with cotree $T_G$, vertices $u$ and $v$ are adjacent in $G$ if and only if their least common ancestor in $T_G$ is $\otimes$.

Two vertices $u$ and $v$ are duplicates if $N(u) = N(v)$ and coduplicates if $N[u] = N[v]$. We call $u$ and $v$ siblings if they are either duplicates or coduplicates. Siblings play an important role in the structure of cographs, as well as in this paper.

Lemma 2. Two vertices $v$ and $u$ in a cograph are siblings if and only if they share the same parent $w$ node in the cotree. Moreover, if $w = \cup$, they are duplicates. If $w = \otimes$ they are coduplicates.

Lemma 3. A cograph $G$ of order $n \geq 2$ has a pair of siblings.

Proof. The cotree of $G$ must have an interior vertex adjacent to two leaves. □

Let $G$ be a cograph with cotree $T_G$. Let $G - v$ denote the subgraph obtained by removing $v$. It is known that $G - v$ is a cograph, so we shall use $T - v$ to denote the cotree of $G - v$. There is a general method for constructing $T - v$ [6, Lem. 1]. However, it somewhat simplifies the process if $v$ has maximum depth. The following lemma can be proved with Lemma 1.

Lemma 4. Let $T_G$ be a cotree, and let $\{v_k, v_l\}$ be siblings of greatest depth, whose parent $w$ has $k$ children. If $k > 2$ we obtain $T - v$ by removing $v$. If $k = 2$ and $w$ is not the root, we obtain $T - v$ by moving $u$ to the parent of $w$, and removing $v$ and $w$. If $k = 2$ and $w$ is the root, the cotree is $u$.

We end this section by making an important observation.

Lemma 5. Let $G$ be a cograph with adjacency matrix $A$ and cotree $T_G$. Let $\{v_k, v_l\}$ be siblings with parent $w$. If $w = \cup$ (they are duplicates), then rows(columns) $k$ and $l$ in $A$ are equal. If $w = \otimes$ (they are coduplicates), then the rows (columns) are equal except in two positions, namely $A[k, k] = A[l, l] = 0$ and $A[k, l] = A[l, k] = 1$.

3. Diagonalizing a row and column

Given a cograph $G$ with adjacency matrix $A$, and $x \in \mathbb{R}$, we will transform $B = A + xI$ into a congruent diagonal matrix in stages. In this section, we illustrate a stage, given
a partially transformed matrix. Recall that matrices are congruent if one can obtain the other by a sequence of \textit{pairs} of elementary operations, each pair consisting of a row operation followed by the same column operation.

Let $T_G$ be the cotree of $G$, and let $\{v_k, v_l\}$ be a pair of siblings. We assume the diagonal values $d_k$ and $d_l$ of rows $k$ and $l$, respectively, may have been modified by previous computations, but only diagonal values. The goal is to annihilate off-diagonal 1’s in the row and column corresponding to $v_k$, maintaining congruence to $B$. Let $w$ be the parent of the siblings in $T_G$. There are two cases.

\textbf{Case 1:} $w = \otimes$. By Lemma 2 we know $\{v_k, v_l\}$ are coduplicates. By Lemma 5 rows (columns) $l$ and $k$ of the matrix $B$ have the form

\[
\begin{bmatrix}
a_1 & a_1 \\
\vdots & \vdots \\
a_i & a_i \\
\vdots & \vdots \\
a_1 \ldots a_i \ldots d_l & 1 \ldots a_n \\
\vdots & \vdots \\
a_1 \ldots a_i \ldots 1 & d_k \ldots a_n \\
\vdots & \vdots \\
a_n & a_n
\end{bmatrix},
\]

where $a_i \in \{0, 1\}$. The row and column operations

\[
R_k \leftarrow R_k - R_l \\
C_k \leftarrow C_k - C_l
\]

give:

\[
\begin{bmatrix}
a_1 & 0 \\
\vdots & \vdots \\
a_i & 0 \\
\vdots & \vdots \\
a_1 \ldots a_i \ldots d_l & 1 - d_l \ldots a_n \\
\vdots & \vdots \\
0 \ldots 0 \ldots 1 - d_l & d_k + d_l - 2 \ldots 0 \\
\vdots & \vdots \\
a_n & 0
\end{bmatrix}
\]

Most of the non-zero elements in row and column $k$ have been removed. But we must now remove the two entries $1 - d_l$. There are three subcases, depending on whether $d_k + d_l - 2 \neq 0$ and whether $d_l = 1$.

\textbf{Subcase 1a:} $d_k + d_l - 2 \neq 0$. Then we may perform the operations

\[
R_l \leftarrow R_l - \frac{1 - d_l}{d_k + d_l - 2} R_k \\
C_l \leftarrow C_l - \frac{1 - d_l}{d_k + d_l - 2} C_k
\]

obtaining:
EIGENVALUE LOCATION IN COGRAPHS

\[
\begin{bmatrix}
  a_1 & 0 \\
  \vdots & \vdots \\
  a_i & 0 \\
  \vdots & \vdots \\
  a_1 & \ldots & a_i & \ldots & \gamma & 0 & \ldots & a_n \\
  0 & \ldots & 0 & \ldots & d_k + d_l - 2 & \ldots & 0 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  a_n & 0 \\
\end{bmatrix},
\]

where

\[
\gamma = d_l - \frac{(1 - d_l)^2}{d_k + d_l - 2} = \frac{d_k d_l - 1}{d_k + d_l - 2}.
\]

The following assignments are made

\[
d_k \leftarrow d_k + d_l - 2 \quad d_l \leftarrow \frac{d_k d_l - 1}{d_k + d_l - 2}. \tag{1}
\]

Since row (and column) \(k\) is diagonalized, the value \(d_k\) becomes permanent value and we remove \(v_k\) from the cotree:

\[
T_G \leftarrow T_G - v_k.
\]

The assignments in (1) are technically incorrect since \(d_k\) is modified in the first assignment and used in the second. In our algorithm’s pseudo-code, we will use temporary variables and assign \(\alpha \leftarrow d_k\) and \(\beta \leftarrow d_l\), and then assign \(d_k \leftarrow \alpha + \beta - 2\) and \(d_l \leftarrow \frac{\alpha \beta - 1}{\alpha + \beta - 2}\). To keep notation simple, in the remainder of this section, we will not do this.

**subcase 1b:** \(d_k + d_l = 2\) and \(d_l = 1\). Then the matrix looks like

\[
\begin{bmatrix}
  a_1 & 0 \\
  \vdots & \vdots \\
  a_i & 0 \\
  \vdots & \vdots \\
  a_1 & \ldots & a_i & \ldots & 1 & 0 & \ldots & a_n \\
  0 & \ldots & 0 & \ldots & 0 & \ldots & 0 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  a_n & 0 \\
\end{bmatrix},
\]

and we are done. We make the assignments

\[
d_k \leftarrow 0 \quad d_l \leftarrow 1 \quad T_G \leftarrow T_G - v_k
\]

as \(d_k\) becomes permanent, and \(v_k\) is removed.

**subcase 1c:** \(d_k + d_l - 2 = 0\) and \(d_l \neq 1\). Then our matrix looks like
We note that \(a_1, \ldots, a_n \in \{0, 1\}\) and since \(1 - d_l \neq 0\), for each \(i \neq k, l\) such that \(a_{il} = 1\), we perform:

\[
R_i \leftarrow R_i - \frac{1}{1 - d_l}R_k
\]

\[
C_i \leftarrow C_i - \frac{1}{1 - d_l}C_k
\]

This annihilates most of row (column) \(l\), without changing any other values:

\[
\begin{bmatrix}
0 & 0 \\
\vdots & \vdots \\
0 & 0 \\
\vdots & \vdots \\
0 \ldots 0 \ldots d_l & 1 - d_l \ldots 0 \\
0 \ldots 0 \ldots 1 - d_l & 0 \ldots 0 \\
\vdots & \vdots \\
0 & 0
\end{bmatrix}
\]

The operations

\[
R_l \leftarrow R_l + \frac{1}{2}R_k
\]

\[
C_l \leftarrow C_l + \frac{1}{2}C_k
\]

replace the diagonal \(d_l\) with one, while the operations

\[
R_k \leftarrow R_k - (1 - d_l)R_l
\]

\[
C_k \leftarrow C_k - (1 - d_l)C_l
\]

eliminate the two off-diagonal elements, finally giving:
What is different about this subcase is that both rows $k$ and $l$ have been diagonalized. We make the following assignments

\[ d_k \leftarrow -(1 - d_l)^2 \quad d_l \leftarrow 1 \quad T_G \leftarrow T_G - v_k \quad T_G \leftarrow T_G - v_l \]

removing both vertices from $T_G$ and making both variables permanent.

**Case 2:** $w = \cup$. By Lemma 5, row $k$ and $l$ of the matrix $B$ look like:

\[
\begin{bmatrix}
0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 & -(1 - d_l)^2 & \cdots & 0 \\
0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 0 \\
\end{bmatrix}
\]

Similar to Case 1, the row and column operations

\[
R_k \leftarrow R_k - R_l \\
C_k \leftarrow C_k - C_l
\]

give:

\[
\begin{bmatrix}
a_1 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
a_i & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
a_1 \cdots a_i \cdots d_l & -d_l & \cdots & a_n \\
0 \cdots 0 \cdots -d_l & d_k + d_l & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\
a_n & 0 & \cdots & 0 \\
\end{bmatrix}
\]

**Subcase 2a:** $d_k + d_l \neq 0$. Then the matrix operations

\[
R_l \leftarrow R_l + \frac{d_l}{d_k + d_l} R_k \\
C_l \leftarrow C_l + \frac{d_l}{d_k + d_l} C_k
\]
diagonalize the matrix and the following assignments are made:

\[ d_k \leftarrow d_k + d_l \quad T_G \leftarrow T_G - v_k \quad d_l \leftarrow \frac{d_k d_l}{d_k + d_l} \]

**subcase 2b:** \( d_k + d_l = 0 \) and \( d_l = 0 \). Similar to **subcase 1b**, the matrix is in diagonal form and the following assignments are made:

\[ d_k \leftarrow 0 \quad T_G \leftarrow T_G - v_k \quad d_l \leftarrow 0 \]

**subcase 2c:** \( d_k + d_l = 0 \) and \( d_l \neq 0 \). Since \( d_l \neq 0 \), we use operations similar to (2) to annihilate most of rows and columns:

\[
\begin{bmatrix}
0 & 0 \\
\vdots & \vdots \\
0 & 0 \\
\vdots & \vdots \\
0 & \ldots & 0 & d_l & -d_l & \ldots & 0 \\
0 & \ldots & 0 & -d_l & 0 & \ldots & 0 \\
\vdots & \vdots \\
0 & 0
\end{bmatrix}
\]

The operations

\[ R_k \leftarrow R_k + R_l \\
C_k \leftarrow C_k + C_l \]

complete the diagonalization. The following assignments are made:

\[ d_k \leftarrow -d_l \quad d_l \leftarrow d_l \quad T_G \leftarrow T_G - v_k \quad T_G \leftarrow T_G - v_l \]

### 4. Diagonalizing \( A + xI \)

The algorithm in Figure 2 constructs a diagonal matrix \( D \), congruent to \( B = A + xI \), where \( A \) is the adjacency matrix of a cograph \( G \), and \( x \in \mathbb{R} \). The algorithm’s input is the cotree \( T_G \) and \( x \). Note the algorithm does not store the matrix, but rather only records changes on the diagonal values \( d_i \) of \( B \), allowing only \( O(n) \) space.

It initializes all entries of \( D \) with \( x \). At the beginning of each iteration, the cotree represents the subgraph induced by vertices whose rows and columns have not yet been diagonalized. During each iteration, a pair of siblings \( \{v_k, v_l\} \) from the cotree is selected, whose existence is guaranteed by Lemma 3. Note that the algorithm can’t assume that either of the diagonal elements \( d_k \) and \( d_l \) are still \( x \).

Each iteration of the loop annihilates either one or two rows (columns), updating diagonal values with arithmetic from Section 3. Once a row (column) is diagonalized, those entries never participate again in row and column operations, and so their values remain unchanged. When a row (column) corresponding to vertex \( v \) has been diagonalized, the subgraph induced by those vertices whose rows (columns) are undiagonalized, has been reduced. Thus \( v \) is removed from the cotree. It is slightly easier to reconstruct the cotree when leaves of maximum depth are removed, using the method of Lemma 4. Hence we always choose sibling pairs of maximum depth. The algorithm terminates when all rows and columns have been diagonalized. Each iteration of \textbf{Diagonalize} takes constant time, so its running time is \( O(n) \).
INPUT: cotree $T_G$, scalar $x$

OUTPUT: diagonal matrix $D = [d_1, d_2, \ldots, d_n]$ congruent to $A(G) + xI$

Algorithm Diagonalize $(T_G, x)$

1. initialize $d_i := x$, for $1 \leq i \leq n$
2. while $T_G$ has $\geq 2$ leaves
   1. select siblings $\{v_k, v_l\}$ of maximum depth with parent $w$
   2. $\alpha \leftarrow d_k$; $\beta \leftarrow d_l$
   3. if $w = \otimes$
      1. if $\alpha + \beta \neq 2$ // subcase 1a
         $d_k \leftarrow \alpha + \beta - 2$; $d_l \leftarrow \frac{\alpha \beta - 1}{\alpha + \beta - 2}$; $T_G = T_G - v_k$
      2. else if $\beta = 1$ // subcase 1b
         $d_k \leftarrow 0$; $d_l \leftarrow 1$; $T_G = T_G - v_k$
      3. else // subcase 1c
         $d_k \leftarrow -(1 - \beta)^2$; $d_l \leftarrow 1$; $T_G = T_G - v_k$; $T_G = T_G - v_l$
   4. else if $w = \cup$
      1. if $\alpha + \beta \neq 0$ // subcase 2a
         $d_k \leftarrow \alpha + \beta$; $d_l \leftarrow \frac{\alpha \beta}{\alpha + \beta}$; $T_G = T_G - v_k$
      2. else if $\beta = 0$ // subcase 2b
         $d_k \leftarrow 0$; $d_l \leftarrow 0$; $T_G = T_G - v_k$
      3. else // subcase 2c
         $d_k \leftarrow -\beta$; $d_l \leftarrow \beta$; $T_G = T_G - v_k$; $T_G = T_G - v_l$
   5. end loop

Figure 2. Diagonalization algorithm

Theorem 1. For inputs $T_G$ and $x$, where $T_G$ is the cotree of cograph $G$ having adjacency matrix $A$, algorithm Diagonalize computes a diagonal matrix $D$, which is congruent to $A + xI$ using $O(n)$ time and space.

It is interesting that for threshold graphs, the cotree is a caterpillar, as shown in [17, Cor. 3.2]. When Diagonalize is given such a cotree, it performs essentially the same arithmetic as the threshold graph algorithm in [12].

Example 1. In the remainder of this section we apply Algorithm Diagonalize to the cograph in Figure 1 with $x = 0$. In our figures, diagonal values $d_i$ appear under the vertex $v_i$ in the cotree. Initially, all $d_i$ will be $x = 0$. Sibling pairs appear in red. Dashed edges indicate vertices about to be removed from the cotree. To follow this example, the reader need not be concerned with labels of vertices, but simply check that each cotree produces the next one.

In the first iteration of the algorithm, siblings $\{v_k, v_l\}$ are chosen of maximum depth. Since $d_k = \alpha = 0$ and $d_l = \beta = 0$, subcase 2b occurs. The assignments

$$d_k \leftarrow 0 \quad d_l \leftarrow 0$$

are made, represented in Figure 3. Figure 4 depicts the cotree after vertex $v_k$ is removed and $v_l$ is relocated under its parent’s parent using the rules in Lemma 4.

In the next step, a sibling pair $\{v_k, v_l\}$ of depth three is chosen. Since their parent is $\otimes$, and $d_k = \alpha = 0$ and $d_l = \beta = 0$, subcase 1a is executed and the following assignments are made:

$$d_k \leftarrow -2 \quad d_l \leftarrow 1$$
Figure 3.

Figure 4.

Figure 5 shows the cotree after these assignments, and Figure 6 shows the cotree with $v_k$ removed and $v_l$ relocated.

Next, another depth three sibling pair $\{v_k, v_l\}$ is chosen. Since $d_k = \alpha = 0$ and $d_l = \beta = 0$, subcase 1a is taken and assignments

$$d_k \leftarrow -2 \quad d_l \leftarrow \frac{1}{2}$$

made. Vertex $v_k$ is removed from the cotree as shown in Figure 7, making the value $d_k = -2$ permanent. Note $v_l$ does not move per the rules in Lemma 4.

Depth three siblings $\{v_k, v_l\}$ are selected again. Since $d_k = \alpha = \frac{1}{2}$ and $d_l = \beta = 0$, subcase 1a is again executed and the assignments

$$d_k \leftarrow -\frac{3}{2} \quad d_l \leftarrow \frac{2}{3}$$

are made as shown in Figure 8. Vertex $v_k$ will be removed from the cotree, and $v_l$ relocated to its parent’s parent, as Figure 9 shows.

Next, a depth two pair $\{v_k, v_l\}$ is chosen. Since $d_k = \alpha = \frac{1}{2}$, and $d_l = \beta = 0$, subcase 2a is applied, and the following assignments are made:

$$d_k \leftarrow \frac{1}{2} \quad d_l \leftarrow 0$$
Their values don’t change, as indicated in Figure 10. Vertex $v_k$ is removed from the cotree, and vertex $v_l$ is relocated as shown in Figure 11. Next, a depth two pair $\{v_k, v_l\}$ is selected. Since $d_k = \alpha = \frac{2}{3}$ and $d_l = \beta = 0$, the subcase 2a is applied again.

The assignments are made

$$d_k \leftarrow \frac{2}{3} \quad d_l \leftarrow 0.$$
In the penultimate step, a depth one pair \( \{ v_k, v_l \} \) is chosen. Since \( d_k = \alpha = 0 \) and \( d_l = \beta = 0 \), subcase 1a is applied, and assignments
\[
d_k \leftarrow -2 \quad d_l \leftarrow \frac{1}{2}
\]
are made, as shown in Figure 14.

The final sibling pair \( \{ v_k, v_l \} \) is chosen. Since \( d_k = \alpha = \frac{1}{2} \) and \( d_l = \beta = 0 \), subcase 1a is applied and
\[
d_k \leftarrow -\frac{3}{2} \quad d_l \leftarrow \frac{2}{3}
\]
as shown in Figure 15. When we remove \( v_k \) using Lemma 4, the remaining cotree is \( v_l \). The algorithm stops, and the final diagonal is shown in Figure 16.
5. Locating eigenvalues

As an application, we can compute in $O(n)$ time the number of eigenvalues of a cograph in a given interval, as was done in [11] for trees and in [12] for threshold graphs. The following theorem is called Sylvester’s Law of Inertia [3, p. 336].

**Theorem 2.** Two $n \times n$ real symmetric matrices are congruent if and only if they have the same number of positive eigenvalues and the same number of negative eigenvalues.

The proof of the following theorem is similar to Theorem 3 in [12], and is based on Theorem 2.

**Theorem 3.** Let $D = [d_1, d_2, \ldots, d_n]$ be the diagonal returned by Diagonalize($T_G, -a$), and assume $D$ has $k_+$ positive values, $k_0$ zeros, and $k_-$ negative values.

- **i:** The number of eigenvalues of $G$ that are greater than $a$ is exactly $k_+$.
- **ii:** The number of eigenvalues of $G$ that are less than $a$ is exactly $k_-$.  
- **iii:** The multiplicity of $a$ is $k_0$.

Note that when we use Diagonalize($T_G, x$) and obtain $D = [d_1, d_2, \ldots, d_n]$, the order of the $d_i$’s depends on the order in which maximum depth sibling pairs are selected. It is not clear whether the values in $D$ are invariant, but their signs certainly are.

**Example 2.** As Figure 16 indicates, Diagonalize($T_G, 0$), produces three positive entries, five negative entries and one zero. Therefore $G$ has 3 positive eigenvalues, 5 negative eigenvalues and 0 is an eigenvalue with multiplicity one.

**Example 3.** One can check that Diagonalize($T_G, 1$), produces a diagonal with multiset \{2, -\frac{1}{2}, 0, 0, 2, 2, -1, -\frac{1}{4}, 1\}. Since there are four positive entries, three negative entries and two zeros, it follows that there are four eigenvalues greater than -1, three less than -1, and -1 has multiplicity two.

Assume $a < b$, and let $k_+$ be the number of positive values in the diagonal of Diagonalize($T_G, -a$), and let $j_+$ be number of positive values in the diagonal of Diagonalize($T_G, -b$). Then $(a, b]$ must contain exactly $k_+ - j_+$ eigenvalues. The number of eigenvalues in $(a, b)$ is $k_+ - j_+ - j_0$, where $j_0$ is the multiplicity of $b$. Thus we can find the number of eigenvalues in an interval by making two calls to the algorithm.

**Example 4.** From Example 2 and Example 3, it follows that $G$ has exactly $4 - 3 = 1$ eigenvalue in $(-1, 0]$. However, from Example 2 we know that zero is an eigenvalue, so
there must be no eigenvalues in \((-1, 0)\). This is not an accident. Recently in \([14]\), the striking result was obtained that no cograph has eigenvalues in \((-1, 0)\).

Any eigenvalue \(\lambda\) can be approximated by first finding an interval for which

\[a < \lambda \leq b.\]

By using a divide-and-conquer approach in which the interval length is successively cut in half, one can find an interval \([a', b']\) of arbitrarily small size for which

\[a' < \lambda \leq b'.\]

See \([12, \text{Sec. 4}]\) for more detail.

6. Inertia of cographs

The inertia of a graph \(G\) is the triple \((n_+, n_0, n_-)\), where \(n_+, n_0, n_-\) denote respectively the number of positive, zero, and negative eigenvalues of \(G\). We wish to compute the inertia of any cograph \(G\). If \(G\) is a single vertex, its inertia is \((0, 1, 0)\), so we assume that \(G\) has order \(n \geq 2\) with cotree \(T_G\). Note also that since \(n = n_+ + n_0 + n_-\), it suffices to compute only two of these components. In what follows, we will obtain formulas for \(n_-\) and \(n_0\) by using our diagonalization algorithm. Our formula for \(n_-\) given in Theorem 4 below appears to be new. The formulas for \(n_0\) (Theorem 5) and the multiplicity of \(-1\) (Theorem 6) can be found in \([2]\). Formulas relating inertia to the representation of threshold graphs can be found in the papers \([1, 13]\).

By Theorem 3, we can obtain the inertia of a cograph by applying \texttt{Diagonalize} to \(T_G\) with \(x = 0\), and then counting the number of entries in the diagonal that are positive, zero and negative. The following technical result appears in Lemma 3 of \([5]\).

Lemma 6. Suppose that \(0 \leq \alpha, \beta < 1\). Then

(a) \(0 < \frac{\alpha \beta - 1}{\alpha + \beta - 2} < 1\)

(b) \(0 \leq \frac{\alpha \beta}{\alpha + \beta} < 1\), provided \(\alpha + \beta \neq 0\).

Lemma 7. If \(\{v_k, v_l\}\) is a sibling pair processed by \texttt{Diagonalize}, with parent \(w = \otimes\) for which \(0 \leq d_k, d_l < 1\), then \(d_k\) becomes permanently negative, and \(d_l\) is assigned a value in \((0, 1)\).

Proof. By our assumption, subcase 1a is executed. Hence

\[d_k \leftarrow \alpha + \beta - 2 \quad d_l \leftarrow \frac{\alpha \beta - 1}{\alpha + \beta - 2}\]

where \(\alpha, \beta\) are the old values of \(d_k, d_l\). Clearly \(d_k < 0\). By Lemma 6(a), \(d_l \in (0, 1)\). \(\square\)

Lemma 8. If \(\{v_k, v_l\}\) is a sibling pair processed by \texttt{Diagonalize}, with parent \(w = \cup\) for which \(0 \leq d_k, d_l < 1\), then \(d_k\) becomes permanently nonnegative and \(d_l\) is assigned a value in \([0, 1)\).

Proof. If \(d_k > 0\) or \(d_l > 0\), then subcase 2a is executed which means

\[d_k \leftarrow \alpha + \beta \quad d_l \leftarrow \frac{\alpha \beta}{\alpha + \beta}\]

Clearly \(d_k > 0\) and by Lemma 6(b), \(d_l \in [0, 1)\). If \(d_k = d_l = 0\), then subcase 2b is executed meaning that both \(d_k\) and \(d_l\) are assigned 0, \(d_k\) permanently so. \(\square\)
Lemma 9. During the execution of Diagonalize\((T_G, 0)\), all diagonal values of vertices remaining on the cotree are in \([0, 1)\).

Proof. Initially all values on \(T_G\) are zero. Suppose after \(m\) iterations of Diagonalize all diagonal values of the cotree are in \([0, 1)\), and consider iteration \(m + 1\) with sibling pair \(\{v_k, v_l\}\) and parent \(w\). By assumption, \(0 \leq d_k, d_l < 1\). If \(w = \otimes\) then Lemma 7 guarantees the vertex \(d_l\) remaining on the tree is assigned a value in \((0, 1)\). If \(w = \cup\), Lemma 8 guarantees \(d_l \in [0, 1)\). This means after \(m + 1\) iterations the cotree \(T_G - v_k\) satisfies the desired property, completing the proof. □

Remark 1. Observe that if \(w\) is an interior node in \(T_G\) having \(t\) children, as the algorithm progresses bottom up through the rules of Lemma 4, each interior child of \(w\) eventually is replaced by a leaf. Thus when \(w\) is ready to be processed it will have \(t - 1\) sibling pairs. To simplify our analysis, without loss of generality we can assume that all \(t - 1\) sibling pairs are processed consecutively.

The following theorem shows that the quantity \(n_\_ (G)\) can be computed in linear time from the cotree.

Theorem 4. Let \(G\) be a cograph with cotree \(T_G\) having \(\otimes\)-nodes \(\{w_1, \ldots, w_j\}\), and assume each \(w_i\) has \(t_i\) children in \(T_G\). Then

\[
n_\_ (G) = \sum_{i=1}^{j} (t_i - 1).
\]

Proof. In executing Diagonalize\((T_G, 0)\), consider an interior node \(w_i\) in \(T_G\) of type \(\otimes\) with \(t_i\) children. By Remark 1 when it becomes eligible to be processed, it will have \(t_i\) leaves, and the algorithm will process \(t_i - 1\) sibling pairs. By Lemma 7 all diagonal values on the cotree remain in \([0, 1)\). By Lemma 7 each of the \(t_i - 1\) sibling pairs will produce a permanent negative value before \(w_i\) is removed. This shows

\[
n_\_ (G) \geq \sum_{i=1}^{j} (t_i - 1).
\]

However Lemma 8 shows that processing a sibling pair with parent \(\cup\) can only produce nonnegative permanent values. Hence the inequality is tight, completing the proof. □

We now consider \(n_0(G)\). A formula by Bıyıkoglu, Simić and Stanić is known [2, Cor. 3.2] but we give an alternate proof using our algorithm.

Remark 2. Consider an interior node \(w\) of type \(\cup\) with \(k = s + t\) children, where \(s\) children are interior (of type \(\otimes\)) and \(t\) children are terminal. In the execution of Diagonalize\((T_G, 0)\), from Lemmas 7 and 9 each \(\otimes\) node will become positive. Thus when \(w\) is processed, it will have \(s\) leaves with positive values and \(t\) leaves with zero.

Theorem 5. Let \(G\) be a cograph with cotree \(T_G\) having \(\cup\)-nodes \(\{w_1, \ldots, w_m\}\), where \(w_i\) has \(t_i\) terminal children. If \(G\) has \(j \geq 0\) isolated vertices, then

\[
n_0(G) = j + \sum_{i=1}^{m} (t_i - 1).
\]

Proof. Consider the execution of Diagonalize\((T_G, 0)\), and first consider the case when \(j = 0\). Let \(w_i\) be an interior node of type \(\cup\) having \(k_i\) children where \(s_i\) children are interior nodes and \(t_i\) are terminal. By Remark 2 when \(w_i\) is ready to be processed
it will have $s_i$ positive children and $t_i$ zeros. From Lemma 8 we see that subcase 2a
will be executed $s_i$ times and subcase 2b $t_i - 1$ times. Each execution of subcase 2b
produces a permanent zero on the diagonal, and so $w_i$ contributes to $t_i - 1$ zeros. This
shows $n_0(G) \geq \sum_{i=1}^{m}(t_i - 1)$. To obtain equality, we note that no zero can be created
when processing a sibling pair whose parent is $\otimes$. If $G$ has $j > 0$ isolates, then the root
of $T_G$ has type $\cup$ with $j$ children as leaves. We claim $j$ additional zeros are created.
Indeed, $j - 1$ are created through subcase 2b. The last iteration, either subcase 2a
or subcase 2b, creates an additional zero. 

It is possible to frame Theorem 5 in terms of duplicate vertices. Let $\{V_i\}$ be a
partition over the set of all vertices that are in duplicate pairs, such that each $V_i$
contains mutually pairwise duplicate vertices, and for $i \neq j$, $v \in V_i$ and $w \in V_j$ imply
$N(v) \neq N(w)$. Then $|V_i| = t_i$.

We mention two other known theorems that can be obtained through our algorithm.
While we omit the details, their proofs involve algorithm analysis when $x = 1$. One
such result [14] is that no cograph has an eigenvalue in $(-1, 0)$. Another [2] involves
the multiplicity of $-1$:

**Theorem 6.** Let $G$ be a cograph with cotree $T_G$ having $\otimes$-nodes \{$w_1, \ldots, w_m$\}, where
$w_i$ has $t_i$ terminal children. Then the multiplicity of $-1$ is $\sum_{i=1}^{m}(t_i - 1)$.

We apply the theorems in this section to the graph of Figure 1. By Theorem 5, we
have $n_0(G) = 2 - 1 = 1$. Applying Theorem 5, $n_-(G) = (3 - 1) + (3 - 1) + (2 - 1) = 5$.
Therefore $n_+(G) = 9 - 5 - 1 = 4$. By Theorem 6, the multiplicity of $-1$ in $G$ is two.
These numbers agree with those given in Section 5.

7. Equienergetic Cographs

Recall that the energy $E(G)$ of a graph $G$ is defined to be $\sum_{i=1}^{n} |\lambda_i|$, where $\lambda_1, \ldots, \lambda_n$
are its eigenvalues. If $E(G) = E(H)$, we say $G$ and $H$ are equienergetic. It is known
that the complete graph $K_n$ has energy $2(n - 1)$. We finish this paper by exhibiting
an infinite class $G = \{G_1, G_2, \ldots, G_r, \ldots\}$ where $G_r$ is a cograph of order $n = 3r + 4$
and $E(G_r) = E(K_{3r+4})$. In [13] the authors gave similar examples of threshold graphs,
however our present example involves non-threshold graphs. This is of interest because
equienergetic examples seem to often involve non-integer energy.

The following two technical lemmas describe the diagonalization algorithm when
multiple leaves of the same parent have the same diagonal value $d_i = y$. In particular,
when an interior vertex $w$ of the cograph has $m$ terminal children, $v_1, \ldots, v_m$, the
algorithm will generally make $m - 1$ iterations. If all $d_i$ are equal, then under certain
conditions, can say exactly what assignments are made at each iteration. We can
assume sibling pairs are chosen to be the leftmost pair. In Lemma 10 we prove the
case when $w = \otimes$, and the case $w = \cup$ in Lemma 11 is handled in a similar way.

**Lemma 10.** If $v_1, \ldots, v_m$ have parent $w = \otimes$, each with diagonal value $y > 1$, then the
algorithm performs $m - 1$ iterations of subcase 1a assigning, during iteration $j$:

$$
d_k \leftarrow \frac{j+1}{j}(y-1) \quad (3)
$$

$$
d_l \leftarrow \frac{y+j}{j+1} \quad (4)
$$

**Proof.** It is easy to check that when $j = 1$, since $\alpha = \beta = y > 1$, subcase 1a will be
chosen and perform assignments $d_k \leftarrow 2(y-1)$ and $d_l \leftarrow \frac{y+1}{2}$. Now suppose iteration $j$
makes assignments (3) and (4). Then during iteration \( j + 1 \), we will have \( \alpha = \frac{y + j}{j + 1} > 1 \) and \( \beta = y > 1 \). Therefore \( \alpha + \beta \neq 2 \) and subcase 1a will execute again. From the rules of the algorithm we get

\[
\begin{align*}
d_k &\leftarrow \alpha + \beta - 2 = \frac{y + j}{j + 1} + y - 2 = \frac{j + 2}{j + 1}(y - 1) \\
d_l &\leftarrow \frac{\alpha \beta - 1}{\alpha + \beta - 2} = \frac{j + 1}{(j + 2)(y - 1)}
\end{align*}
\]

which completes the induction. \( \square \)

Lemma 11. If \( v_1, \ldots, v_m \) have parent \( w = \cup \), each with diagonal value \( y > 0 \), then the algorithm performs \( m - 1 \) iterations of subcase 2a, assigning during iteration \( j \):

\[
\begin{align*}
d_k &\leftarrow \frac{(j + 1)y}{j} \\
d_l &\leftarrow \frac{y}{j + 1}
\end{align*}
\]

Proof. Similar to Lemma 10. \( \square \)

For each integer \( r \geq 1 \), we now define the cograph \( G_r \) to be the join of \( K_{r+2} \) with the disjoint union of \( r + 1 \) copies of \( K_2 \), that is

\[
G_r = (K_2 \cup K_2 \cup \ldots \cup K_2) \otimes K_{r+2}.
\]

\( G_r \) has order \( n = 3r + 4 \) and its cotree is shown in Figure 17. Let \( m(\lambda;G) \) denote the multiplicity of an eigenvalue in \( G \).

Lemma 12. \( G_r \) has inertia \((r + 1, 0, 2r + 3)\) and \( m(-1;G_r) = 2(r + 1) \).

Proof. This follows from Theorem 3, Theorem 5 and Theorem 6. \( \square \)

Lemma 13. The eigenvalue 1 in \( G_r \) has multiplicity \( r \).

Proof. By Theorem 3 it suffices to show that \( \text{Diagonalize}(T_{G_r}, -1) \) creates exactly \( r \) zeros. After initializing vertices with \(-1\), the algorithm performs subcase 1a on each of the \( r + 1 \) sibling pairs \( \{v_k, v_l\} \) of depth three. Each operation

\[
\begin{align*}
d_k &\leftarrow -4 \\
d_l &\leftarrow 0
\end{align*}
\]

leaves a zero on the tree at depth two. After all depth three pairs have been processed, the algorithm applies subcase 2b to \( r \) sibling pairs \( \{v_k, v_l\} \) where \( d_k = d_l = 0 \). This creates \( r \) permanent zeros, and so \( m(1;G_r) \geq r \). By Lemma 12, \( G_r \) has exactly \( r + 1 \) zeros.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{Figure17.png}
\caption{The cotree \( T_{G_r} \).}
\end{figure}
positive eigenvalues. However \( m(1; G_r) = r + 1 \) would contradict the well-known fact that a graph's largest eigenvalue is simple. So we have \( m(1; G_r) = r \). □

Since \( n = 3r + 4 \), Lemmas 12 and 13 account for all but two eigenvalues.

**Lemma 14.** The largest and smallest eigenvalues of \( G_r \) are

\[
\begin{align*}
\lambda_1 &= 2r + 3 \\
\lambda_n &= -(r + 1).
\end{align*}
\]

**Proof.** We will prove (8), and (7) will follow since a graph's eigenvalues sum to zero. To show (8), it suffices to prove that Diagonalize\( (T_{G_r}, r + 1) \) creates a non-negative diagonal with a single zero. Consider the \( r + 1 \) sibling pairs at depth three. Since \( \alpha = \beta = r + 1 \), for each pair subcase 1a is executed. The assignments

\[
\begin{align*}
d_k &\leftarrow 2r \\
d_l &\leftarrow \frac{r + 2}{2}
\end{align*}
\]

are made, leaving \( r + 1 \) vertices with a permanent positive value of \( 2r \).

At depth two, there are \( r + 1 \) leaves all with positive value \( y = \frac{r + 2}{2} \), and \( r \) iterations are performed. By Lemma 11, each iteration generates the positive permanent diagonal value of (3). On iteration \( r \), the assignment in (6)

\[
d_l \leftarrow \frac{y}{j + 1} = \frac{r + 2}{2(r + 1)}
\]

is made to a vertex \( u \) which then gets moved under the root.

At depth one, there are \( r + 3 \) leaves: \( r + 2 \) with diagonal value \( r + 1 \), and a single leaf \( u \) whose diagonal value is \( \frac{r + 2}{2(r + 1)} \). Assume the algorithm processes the leaves with identical value first. Letting \( y = r + 1 \) in Lemma 10 we see that on each of the \( r + 1 \) iterations the algorithm generates the permanent positive diagonal value in (3). Letting \( j = r + 1 \) in (4) we see that the last iteration leaves the value \( \frac{2(r + 1)}{r + 2} \) on the tree. For the last step of the algorithm, we process the two remaining vertices whose diagonal values are \( \alpha = \frac{2(r + 1)}{r + 2} \) and \( \beta = \frac{r + 2}{2(r + 1)} \). Since \( \alpha > 2 \) and \( \beta > 0 \), on the last iteration subcase 1a assigns

\[
\begin{align*}
d_k &\leftarrow \alpha + \beta - 2 > 0 \\
d_l &\leftarrow \frac{\alpha \beta - 1}{\alpha + \beta - 2} = 0
\end{align*}
\]

creating a positive value and zero for the last two diagonal entries. □

From Lemma 12, Lemma 13, and Lemma 14 it follows that

\[
E(G_r) = 2(r + 1) + r + (2r + 3) + (r + 1) = 2(3r + 4) - 2 = 2n - 2
\]

so we have:

**Theorem 7.** For each \( r \geq 1 \), \( G_r \) and \( K_{3r+4} \) are equienergetic.

Clearly \( G_r \) and \( K_n \) are noncospectral. Note \( G_r \) is not a threshold graph since its cotree is not a caterpillar. Three other infinite classes of cographs, equienergetic to complete graphs, were discovered. In all cases, the cotrees had depth three. Since the cotree structure and proofs are similar to the above example, we omit them.
References

1. R. B. Bapat, *On the adjacency matrix of a threshold graph*, Linear Algebra Appl. 439 (2013), no. 10, 3008–3015.
2. Türker Büyüköngütürk, Slobodan K. Simić, and Zoran Stanić, *Some notes on spectra of cographs*, Ars Combin. 100 (2011), 421–434.
3. Gerald L. Bradley, *A primer of linear algebra*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.
4. Andries E. Brouwer and Willem H. Haemers, *Spectra of graphs*, Universitext, Springer, New York, 2012.
5. Gerard J. Chang, Liang-Hao Huang, and Hong-Gwa Yeh, *On the rank of a cograph*, Linear Algebra Appl. 429 (2008), no. 2-3, 601–605.
6. D. G. Corneil, H. Lerchs, and L. Stewart Burlingham, *Complement reducible graphs*, Discrete Appl. Math. 3 (1981), no. 3, 163–174.
7. D. G. Corneil, Y. Perl, and L. K. Stewart, *A linear recognition algorithm for cographs*, SIAM J. Comput. 14 (1985), no. 4, 926–934.
8. Dragoş M. Cvetković, Michael Doob, and Horst Sachs, *Spectra of graphs*, third ed., Johann Ambrosius Barth, Heidelberg, 1995, Theory and applications.
9. Renata R. Del-Vecchio, David P. Jacobs, Vilmar Trevisan, and Cybele T. M. Vinagre, *Diagonalization of generalized lollipop graphs*, Proc. 8th Latin-American Algorithms, Graphs, and Optimization Symposium, Electron. Notes Discrete Math., LAGOS 2015, Beberibe, to appear.
10. Michel Habib and Christophe Paul, *A simple linear time algorithm for cograph recognition*, Discrete Appl. Math. 145 (2005), no. 2, 183–197.
11. David P. Jacobs and Vilmar Trevisan, *Locating the eigenvalues of trees*, Linear Algebra Appl. 434 (2011), no. 1, 81–88.
12. David P. Jacobs, Vilmar Trevisan, and Fernando Tura, *Eigenvalue location in threshold graphs*, Linear Algebra Appl. 439 (2013), no. 10, 2762–2773.
13. ______, *Eigenvalues and energy in threshold graphs*, Linear Algebra Appl. 465 (2015), 412–425.
14. Ali Mohammadian and Vilmar Trevisan, *Some spectral properties of cographs*, Manuscript submitted for publication, 2015.
15. Laura Patuzzi, Maria Aguiar A. de Freitas, and Renata R. Del-Vecchio, *Indices for special classes of trees*, Linear Algebra Appl. 442 (2014), 106–114.
16. Gordon F. Royle, *The rank of a cograph*, Electron. J. Combin. 10 (2003), Note 11, 7 pp. (electronic).
17. Irene Sciriha and Stephanie Farrugia, *On the spectrum of threshold graphs*, ISRN Discrete Mathematics 2011, 1–29.

School of Computing, Clemson University Clemson, SC 29634 USA
E-mail address: dpj@clemson.edu

Instituto de Matemática, UFRGS, 91509–900 Porto Alegre, RS, Brazil
E-mail address: trevisan@mat.ufrgs.br

Departamento de Matemática, UFSM, 97105–900 Santa Maria, RS, Brazil
E-mail address: ftura@smail.ufsm.br