The Fiedler Vector and Tree Decompositions of Graphs

Wayne Barrett

June 4, 2015

Brigham Young University
Collaborators

BYU Undergraduates:
Bryn Balls-Barker
Derek Sorensen
Adam Shumway
Ryan Echols

BYU Visiting Professor
Amanda Francis
Undirected Graph
Undirected Graph

6 vertices  9 edges
Undirected Graph

6 vertices  9 edges
no loops, no multiple edges
Adjacency matrix of a graph $G$

$$A(G) = \begin{bmatrix}
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix}.$$
Adjacency matrix of a graph $G$

$$A(G) = \begin{bmatrix}
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix}.$$
Adjacency matrix of a graph $G$

$$A(G) = \begin{bmatrix}
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix}.$$
Adjacency matrix of a graph $G$

\[ A(G) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \]
Definition

If $G$ is a graph on $n$ vertices, its adjacency matrix $A(G)$ is the $n \times n$ symmetric $(0, 1)$-matrix defined by

$$A(G) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$
Adjacency matrix of a graph $G$

$$A(G) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$ 

**Definition**

If $G$ is a graph on $n$ vertices, its adjacency matrix $A(G)$ is the $n \times n$ symmetric $(0, 1)$-matrix defined by

$$a_{ij} = \begin{cases} 1 & \text{if } \{i, j\} \text{ is an edge of } G \\ 0 & \text{otherwise} \end{cases}$$
Adjacency matrix of a graph $G$

\[ A(G) = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \]

**Definition**

If $G$ is a graph on $n$ vertices, its adjacency matrix $A(G)$ is the $n \times n$ symmetric $(0, 1)$-matrix defined by

\[ a_{ij} = \begin{cases} 1 & \text{if } \{i, j\} \text{ is an edge of } G \\ 0 & \text{otherwise}. \end{cases} \]
Laplacian matrix of a graph

\[ L(G) = \begin{pmatrix}
  2 & -1 & -1 & 0 \\
  -1 & 2 & -1 & 0 \\
  -1 & -1 & 3 & -1 \\
  0 & 0 & -1 & 1
\end{pmatrix}. \]

Definition

If \( G \) is a graph on \( n \) vertices, \( L(G) \) is the \( n \times n \) symmetric integer matrix defined by

\[ \ell_{ij} = \begin{cases} 
  \deg(i) & \text{if } i = j \\
  -1 & \text{if } i \neq j \text{ and } \{i, j\} \text{ is an edge of } G \\
  0 & \text{otherwise}.
\end{cases} \]
Laplacian matrix of a graph

\[ \mathbf{L}(G) = \begin{bmatrix}
2 & -1 & -1 & 0 \\
-1 & 2 & -1 & 0 \\
-1 & -1 & 3 & -1 \\
0 & 0 & -1 & 1
\end{bmatrix} \]

**Definition**

If \( G \) is a graph on \( n \) vertices, \( \mathbf{L}(G) \) is the \( n \times n \) symmetric integer matrix defined by

\[
\ell_{ij} = \begin{cases} 
\deg(i) & \text{if } i = j \\
-1 & \text{if } i \neq j \text{ and } \{i, j\} \text{ is an edge of } G \\
0 & \text{otherwise.}
\end{cases}
\]
Laplacian matrix of a graph

\[ L(G) = \begin{bmatrix}
2 & -1 & -1 & 0 \\
-1 & 2 & -1 & 0 \\
-1 & -1 & 3 & -1 \\
0 & 0 & -1 & 1
\end{bmatrix}. \]
Laplacian matrix of a graph

If $G$ is a graph on $n$ vertices, $L(G)$ is the $n \times n$ symmetric integer matrix defined by

$$L(G) = \begin{bmatrix}
2 & -1 & -1 & 0 \\
-1 & 2 & -1 & 0 \\
-1 & -1 & 3 & -1 \\
0 & 0 & -1 & 1
\end{bmatrix}.$$
Laplacian matrix of a graph

If $G$ is a graph on $n$ vertices, $L(G)$ is the $n \times n$ symmetric integer matrix defined by

$$L(G) = \begin{bmatrix}
2 & -1 & -1 & 0 \\
-1 & 2 & -1 & 0 \\
-1 & -1 & 3 & -1 \\
0 & 0 & -1 & 1
\end{bmatrix}.$$

Definition

If $G$ is a graph on $n$ vertices, $L(G)$ is the $n \times n$ symmetric integer matrix defined by

$$\ell_{ij} = \begin{cases}
\deg(i) & \text{if } i = j \\
-1 & \text{if } i \neq j \text{ and } \{i, j\} \text{ is an edge of } G. \\
0 & \text{otherwise}.
\end{cases}$$
- $L(G)$ is a singular positive semidefinite matrix
- $L(G)$ is a singular positive semidefinite matrix
  - The all ones vector is an eigenvector with eigenvalue $0$. 
Properties and uses of the Laplacian Matrix

- $L(G)$ is a singular positive semidefinite matrix
  - The all ones vector is an eigenvector with eigenvalue 0.
- The eigenvalues of $L(G)$ can be ordered $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. (Fiedler’s initial observation)
- $\lambda_2 > 0$ if and only if the graph $G$ is connected.
- Eigenvectors associated with $\lambda_2$ have been used to produce nice drawings of a graph and to partition a graph.
Properties and uses of the Laplacian Matrix

- $L(G)$ is a singular positive semidefinite matrix
  - The all ones vector is an eigenvector with eigenvalue 0.
- The eigenvalues of $L(G)$ can be ordered $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.
- (Fiedler’s initial observation)
  - $\lambda_2 > 0$ if and only if the graph $G$ is connected.

Eigenvectors associated with $\lambda_2$ have been used to produce nice drawings of a graph and to partition a graph.

Definition: Any eigenvector corresponding to $\lambda_2 (G)$ is called a Fiedler vector of $G$. 

Wayne Barrett (BYU)
- $L(G)$ is a singular positive semidefinite matrix
  - The all ones vector is an eigenvector with eigenvalue 0.
- The eigenvalues of $L(G)$ can be ordered $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.
- (Fiedler’s initial observation)
  - $\lambda_2 > 0$ if and only if the graph $G$ is connected.
- Eigenvectors associated with $\lambda_2$ have been used to produce nice drawings of a graph and to partition a graph.
Properties and uses of the Laplacian Matrix

- $L(G)$ is a singular positive semidefinite matrix
  - The all ones vector is an eigenvector with eigenvalue 0.
- The eigenvalues of $L(G)$ can be ordered $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.
- (Fiedler’s initial observation)
  \[ \lambda_2 > 0 \] if and only if the graph $G$ is connected.
- Eigenvectors associated with $\lambda_2$ have been used to produce nice drawings of a graph and to partition a graph.

**Definition:** Any eigenvector corresponding to $\lambda_2(G)$ is called a Fiedler vector of $G$. 
A cut vertex \( v \) of a connected graph \( G \) is a vertex whose removal disconnects \( G \).

The vertex 3 is a cut vertex.
A cut vertex $v$ of a connected graph $G$ is a vertex whose removal disconnects $G$. 

The vertex 3 is a cut vertex.

A block of a graph is a maximal subgraph with no cut vertex.
A cut vertex $v$ of a connected graph $G$ is a vertex whose removal disconnects $G$.

The vertex 3 is a cut vertex.
A **cut vertex** $v$ of a connected graph $G$ is a vertex whose removal disconnects $G$.

The vertex 3 is a cut vertex.

A **block** of a graph is a maximal subgraph with no cut vertex.
Labeling a graph by a Fiedler eigenvector

**Example** \( P_4 \)

\[
\begin{pmatrix}
1 & -1 & 0 & 0 \\
-1 & 2 & -1 & 0 \\
0 & -1 & 1 & -1 \\
& & & \\
\end{pmatrix} \approx \begin{pmatrix}
.65 & .27 & -.27 & -.65 \\
& & & \\
\end{pmatrix}
\]

Call a vertex positive (negative) (null) if its label is + (−) (0).

Wayne Barrett (BYU)

Fiedler Vector

June 4, 2015
Example $P_4$

Labeling a graph by a Fiedler eigenvector

\[
\begin{pmatrix}
  1 & -1 & 0 & 0 \\
  -1 & 2 & -1 & 0 \\
  0 & -1 & 2 & -1 \\
  0 & 0 & -1 & 1 \\
\end{pmatrix}
\begin{pmatrix}
  .65 \\
  .27 \\
  -.27 \\
  -.65 \\
\end{pmatrix}
\approx
\begin{pmatrix}
  .65 \\
  .27 \\
  -.27 \\
  -.65 \\
\end{pmatrix}
\]
Labeling a graph by a Fiedler eigenvector

Example \( P_4 \)

\[
\begin{bmatrix}
1 & -1 & 0 & 0 \\
-1 & 2 & -1 & 0 \\
0 & -1 & 2 & -1 \\
0 & 0 & -1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
.65 \\
.27 \\
-.27 \\
-.65 \\
\end{bmatrix}
\approx
\begin{bmatrix}
.65 \\
.27 \\
-.27 \\
-.65 \\
\end{bmatrix}
\]

Call a vertex positive (negative) (null) if its label is + (−) (0).
Example $P_4$

\[
\begin{bmatrix}
1 & -1 & 0 & 0 \\
-1 & 2 & -1 & 0 \\
0 & -1 & 2 & -1 \\
0 & 0 & -1 & 1 \\
\end{bmatrix}
\approx
\begin{bmatrix}
.65 \\
.27 \\
-.27 \\
-.65 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
.65 \\
.27 \\
-.27 \\
-.65 \\
\end{bmatrix}
\]

Call a vertex positive (negative) (null) if its label is $+ (-) (0)$.
Theorem on Fiedler Vectors

**Theorem**

Let $G$ be a connected graph with vertices labeled by a Fiedler vector $y$ of $G$, then exactly one of the following cases occurs.

1. **Case A.**
   - There is a single block $B_0$ with both positive and negative vertices.
   - Each other block has all $+$, all $-$, or all null vertices.
   - The labels of any path that contains at most 2 cut vertices per block $\&$ which begins with one vertex in $B_0$ form an increasing, decreasing, or zero sequence, according as the initial vertex is labeled $+$, $-$, or 0.
Theorem

Let $G$ be a connected graph with vertices labeled by a Fiedler vector $y$ of $G$, then exactly one of the following cases occurs.

Case A.

There is a single block $B_0$ with both positive and negative vertices. Each other block has all $+$, all $-$, or all null vertices. The labels of any path that contains at most 2 cut vertices per block and which begins with one vertex in $B_0$ form an increasing, decreasing, or zero sequence, according as the initial vertex is labeled $+$, $-$, or 0.
Theorem on Fiedler Vectors

Theorem

Let $G$ be a connected graph with vertices labeled by a Fiedler vector $y$ of $G$. Then exactly one of the following cases occurs.

Case A.

- There is a single block $B_0$ with both positive and negative vertices.
Theorem on Fiedler Vectors

**Theorem**

Let $G$ be a connected graph with vertices labeled by a Fiedler vector $y$ of $G$, Then exactly one of the following cases occurs.

**Case A.**

- There is a single block $B_0$ with both positive and negative vertices.
- Each other block has all $+$, all $-$, or all null vertices.
Theorem on Fiedler Vectors

Theorem

Let $G$ be a connected graph with vertices labeled by a Fiedler vector $y$ of $G$, Then exactly one of the following cases occurs.

Case A.

- There is a single block $B_0$ with both positive and negative vertices.
- Each other block has all $+$, all $-$, or all null vertices.
- The labels of any path that contains at most 2 cut vertices per block & which begins with one vertex in $B_0$ form an increasing, decreasing, or zero sequence, according as the initial vertex is labeled $+$, $-$, or 0.
Theorem on Fiedler Vectors

Case B.

No block of $G$ contains both positive and negative vertices. There is a unique null vertex $z$ which is adjacent to a non-null vertex. This vertex is a cut vertex. Each block that does not contain $z$ contains either only $+$ vertices, only $-$ vertices, or only null vertices. The labels of any path that contains at most 2 cut vertices per block & which begins at $z$ either increase at each cut vertex, decrease at each cut vertex, or consists entirely of null vertices. Every path containing positive and negative vertices passes through $z$. 
Case B.

- No block of $G$ contains both positive and negative vertices.
Case B.

- No block of $G$ contains both positive and negative vertices.
- There is a unique null vertex $z$ which is adjacent to a non-null vertex. This vertex is a cut vertex.
Theorem on Fiedler Vectors

Case B.

- No block of $G$ contains both positive and negative vertices.
- There is a unique null vertex $z$ which is adjacent to a non-null vertex. This vertex is a cut vertex.
- Each block that does not contain $z$ contains either only $+$ vertices, only $-$ vertices, or only null vertices.
No block of $G$ contains both positive and negative vertices.

There is a unique null vertex $z$ which is adjacent to a non-null vertex. This vertex is a cut vertex.

Each block that does not contain $z$ contains either only $+$ vertices, only $-$ vertices, or only null vertices.

The labels of any path that contains at most 2 cut vertices per block & which begins at $z$ either increase at each cut vertex, decrease at each cut vertex, or consists entirely of null vertices.
Case B.

- No block of $G$ contains both positive and negative vertices.
- There is a unique null vertex $z$ which is adjacent to a non-null vertex. This vertex is a cut vertex.
- Each block that does not contain $z$ contains either only $+$ vertices, only $-$ vertices, or only null vertices.
- The labels of any path that contains at most 2 cut vertices per block & which begins at $z$ either increase at each cut vertex, decrease at each cut vertex, or consists entirely of null vertices.
- Every path containing positive and negative vertices passes through $z$. 
If a graph is labeled by a Fiedler vector of the Laplacian matrix, vertices with large values (±) are on the “periphery” of the graph, while vertices with small values are in the “interior” of the graph.
If a graph is labeled by a Fiedler vector of the Laplacian matrix, vertices with large values (+ or −) are on the “periphery” of the graph, while vertices with small values are in the “interior” of the graph.

One consequence is that a Fiedler vector can be employed to construct a **tree decomposition** of a graph.
A *tree decomposition* of a graph $G = (V, E)$ is a tree $T = (\mathcal{W}, \mathcal{E})$, where the node set $\mathcal{W}$ is a collection of subsets $\mathcal{W}_t$ of $V(G)$ with the following properties:

1. The union of all $\mathcal{W}_t$ in $\mathcal{W}$ is $V(G)$. 

Example Failure of property 1. Collection does not include all the vertices.
A *tree decomposition* of a graph $G = (V, E)$ is a tree $T = (\mathcal{W}, \mathcal{E})$, where the node set $\mathcal{W}$ is a collection of subsets $W_t$ of $V(G)$ with the following properties:

1. The union of all $W_t$ in $\mathcal{W}$ is $V(G)$. 

Example: Failure of property 1. Collection does not include all the vertices.
A *tree decomposition* of a graph $G = (V, E)$, where the node set $\mathcal{W}$ is a collection of subsets $\mathcal{W}_t$ of $V(G)$ with the following properties:

1. The union of all $\mathcal{W}_t$ in $\mathcal{W}$ is $V(G)$.

**Example**
A tree decomposition of a graph $G = (V, E)$ is a tree $T = (W, E)$, where the node set $W$ is a collection of subsets $W_t$ of $V(G)$ with the following properties:

1. The union of all $W_t$ in $W$ is $V(G)$.

Example
A tree decomposition of a graph $G = (V, E)$ is a tree $T = (\mathcal{W}, \mathcal{E})$, where the node set $\mathcal{W}$ is a collection of subsets $W_t$ of $V(G)$ with the following properties:

1. The union of all $W_t$ in $\mathcal{W}$ is $V(G)$.

Example

[Diagram of a tree and a tree decomposition]

Failure of property 1. Collection does not include all the vertices.
A tree decomposition of a graph $G = (V, E)$ is a tree $T = (\mathcal{W}, \mathcal{E})$, where the node set $\mathcal{W}$ is a collection of subsets $W_t$ of $V(G)$ with the following properties:

1. The union of all $W_t$ in $\mathcal{W}$ is $V(G)$.
2. Every edge of $G$ lies in some $W_t$.
A tree decomposition of a graph $G$ is a tree $T = (\mathcal{W}, \mathcal{E})$, where the node set $\mathcal{W}$ is a collection of subsets $W_t$ of $V(G)$ with the following properties:

1. The union of all $W_t$ in $\mathcal{W}$ is $V(G)$.
2. Every edge of $G$ lies in some $W_t$. 
A *tree decomposition* of a graph $G = (V, E)$ is a tree $T = (W, E)$, where the node set $W$ is a collection of subsets $W_t$ of $V(G)$ with the following properties:

1. The union of all $W_t$ in $W$ is $V(G)$.
2. Every edge of $G$ lies in some $W_t$.

Failure of property 2. Collection does not include all the edges.
A tree decomposition of a graph $G = (V, E)$ is a tree $T = (W, E')$, where the node set $W$ is a collection of subsets $W_t$ of $V(G)$ with the following properties:

1. The union of all $W_t$ in $W$ is $V(G)$.
2. Every edge of $G$ lies in some $W_t$.
3. If $W_i, W_j, W_k \in W$ such that $W_k$ lies on the path from $W_i$ to $W_j$ in $T$, then $W_i \cap W_j \subseteq W_k$. 

Wayne Barrett (BYU)
A *tree decomposition* of a graph $G = (V, E)$, where the node set $V$ is a collection of subsets $W_t$ of $V(G)$ with the following properties:

1. The union of all $W_t$ in $W$ is $V(G)$.
2. Every edge of $G$ lies in some $W_t$.
3. If $W_i, W_j, W_k \in W$ such that $W_k$ lies on the path from $W_i$ to $W_j$ in $T$, then $W_i \cap W_j \subseteq W_k$. 

![Diagram of tree decomposition](image)
A tree decomposition of a graph $G = (V, E)$, where the node set $\mathcal{V}$ is a collection of subsets $W_t$ of $V(G)$ with the following properties:

1. The union of all $W_t$ in $\mathcal{W}$ is $V(G)$.
2. Every edge of $G$ lies in some $W_t$.
3. If $W_i, W_j, W_k \in \mathcal{W}$ such that $W_k$ lies on the path from $W_i$ to $W_j$ in $T$, then $W_i \cap W_j \subseteq W_k$. 
Tree Decomposition

A *tree decomposition* of a graph $G = (V, E)$ is a tree $T = (\mathcal{W}, \mathcal{E})$, where the node set $\mathcal{W}$ is a collection of subsets $W_t$ of $V(G)$ with the following properties:

1. The union of all $W_t$ in $\mathcal{W}$ is $V(G)$.
2. Every edge of $G$ lies in some $W_t$.
3. If $W_i, W_j, W_k \in \mathcal{W}$ such that $W_k$ lies on the path from $W_i$ to $W_j$ in $T$, then $W_i \cap W_j \subseteq W_k$.

Failure of property 3. Vertex 2 is in each terminal node set, but not in the middle one.
Valid Tree Decomposition:

- {0,2,3,4} 
- {3,5} 
- {1,2,3}
Valid Tree Decomposition:

A tree decomposition can yield a convenient representation of the various connections between vertices of a graph and might be employed in network science for community detection, node characterization, and anomaly identification.
Example
The corresponding Fiedler vector is

\[
\begin{bmatrix}
1 \\
-.77 \\
-.23 \\
-.77 \\
3.3 \\
-2.5
\end{bmatrix}
\]
Sort the entries of the Fiedler vector from least to greatest, keeping track of which entry corresponds to which vertex.

\[
\begin{bmatrix}
-2.5 \\
-0.77 \\
-0.77 \\
-0.23 \\
1 \\
3.3
\end{bmatrix}
\begin{array}{c}
5 \\
1 \\
3 \\
2 \\
0 \\
4
\end{array}
\]
The Algorithm

Identify the vertex corresponding to the entry of largest magnitude. Create a node set with this vertex and its neighbors.
The Algorithm

Identify the vertex corresponding to the entry of largest magnitude. Create a node set with this vertex and its neighbors.

\[
\begin{bmatrix}
-2.5 \\
-.77 \\
-.77 \\
-.23 \\
1 \\
3.3 \\
\end{bmatrix}
\begin{align*}
5 \\
1 \\
3 \\
2 \\
0 \\
4 \\
\end{align*}
\]

Then delete this vertex from the graph and its corresponding entry from the vector.
The Algorithm

Identify the vertex corresponding to the entry of largest magnitude. Create a node set with this vertex and its neighbors.

\[
\begin{bmatrix}
-2.5 \\
-0.77 \\
-0.77 \\
-0.23 \\
1 \\
3.3
\end{bmatrix}
\]

\[
\begin{bmatrix}
5 \\
1 \\
3 \\
2 \\
0 \\
4
\end{bmatrix}
\]

Wayne Barrett (BYU)
The Algorithm

Identify the vertex corresponding to the entry of largest magnitude. Create a node set with this vertex and its neighbors.

\[
\begin{bmatrix}
-2.5 \\
-0.77 \\
-0.77 \\
-0.23 \\
1 \\
3.3
\end{bmatrix}
\]

\[W_1 = \{0, 4\} \quad \text{label 3.3}\]
The Algorithm

Identify the vertex corresponding to the entry of largest magnitude. Create a node set with this vertex and its neighbors.

\[
\begin{bmatrix}
-2.5 \\
-.77 \\
-.77 \\
-.23 \\
1 \\
3.3
\end{bmatrix}
\]

\[
\begin{array}{c}
5 \\
1 \\
3 \\
2 \\
0 \\
4
\end{array}
\]

\[W_1 = \{0, 4\} \text{ label 3.3}\]

Then delete this vertex from the graph and its corresponding entry from the vector.
The Algorithm

Repeat this step with the entry of second-greatest magnitude.
The Algorithm

Repeat this step with the entry of second-greatest magnitude.

\[
\begin{bmatrix}
-2.5 \\
-0.77 \\
-0.77 \\
-0.23 \\
1 \\
*
\end{bmatrix}
\begin{bmatrix}
5 \\
1 \\
3 \\
2 \\
0 \\
*
\end{bmatrix}
\]

Continue to iterate until edges are included in the tree decomposition.
Repeat this step with the entry of second-greatest magnitude.

\[
\begin{bmatrix}
-2.5 \\
-.77 \\
-.77 \\
-.23 \\
1 \\
*
\end{bmatrix}
\begin{bmatrix}
5 \\
1 \\
3 \\
2 \\
0 \\
*
\end{bmatrix}
\]

Continue to iterate until edges are included in the tree decomposition.
The Algorithm

 Repeat this step with the entry of second-greatest magnitude.

$$\begin{bmatrix} -2.5 \\ -0.77 \\ -0.77 \\ -0.23 \\ 1 \\ * \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \\ 3 \\ 2 \\ 0 \\ * \end{bmatrix}$$

$$W_2 = \{3, 5\} \text{ label } -2.5$$
The Algorithm

Repeat this step with the entry of second-greatest magnitude.

\[
\begin{pmatrix}
-2.5 \\
-0.77 \\
-0.77 \\
-0.23 \\
1 \\
*
\end{pmatrix}
\begin{pmatrix}
5 \\
1 \\
3 \\
2 \\
0 \\
*
\end{pmatrix}
\]

\[W_2 = \{3, 5\} \quad \text{label} \ -2.5\]

Continue to iterate until edges are included in the tree decomposition.
The Algorithm

\[
\begin{bmatrix}
* & * \\
-.77 & 1 \\
-.77 & 3 \\
-.23 & 2 \\
1 & 0 \\
* & *
\end{bmatrix}
\]

\[W_3 = \{0, 2, 3\}\] label 1
The Algorithm

\[
\begin{bmatrix}
* & * \\
-.77 & 1 \\
-.77 & 3 \\
-.23 & 2 \\
1 & 0 \\
* & * \\
\end{bmatrix}
\]

\[W_3 = \{0, 2, 3\}\]

label 1
The Algorithm

\[
\begin{bmatrix}
  * & * \\
  -.77 & 1 \\
  -.77 & 3 \\
  -.23 & 2 \\
  1 & 0 \\
  * & * 
\end{bmatrix}
\]

\[W_3 = \{0, 2, 3\} \quad \text{label 1}\]
Case of a tie: Group entries of equal magnitude and sign together and take all of their neighbors.
**Case of a tie:** Group entries of equal magnitude and sign together and take all of their neighbors.

\[
\begin{bmatrix}
* & * \\
-.77 & 1 \\
-.77 & 3 \\
-.23 & 2 \\
* & * \\
* & *
\end{bmatrix}
\]
**Case of a tie:** Group entries of equal magnitude and sign together and take all of their neighbors.

\[
\begin{bmatrix}
  * & * \\
  -0.77 & 1 \\
  -0.77 & 3 \\
  -0.23 & 2 \\
  * & * \\
  * & *
\end{bmatrix}
\]
Case of a tie: Group entries of equal magnitude and sign together and take all of their neighbors.

\[
\begin{bmatrix}
  * & * \\
  -0.77 & 1 \\
  -0.77 & 3 \\
  -0.23 & 2 \\
  * & * \\
  * & *
\end{bmatrix}
\]

\[W_4 = \{1, 2, 3\} \quad \text{label } -0.77\]
All edges are covered by the four node sets $W_1, W_2, W_3, W_4$.
The Algorithm

All edges are covered by the four node sets $W_1, W_2, W_3, W_4$

Arrange the node sets in the order of their labels:
The Algorithm

All edges are covered by the four node sets $W_1, W_2, W_3, W_4$

Arrange the node sets in the order of their labels:

$$-2.5 \quad -.77 \quad 1 \quad 3.3$$

Property 3 of tree decompositions is satisfied.

And the tree looks optimal.
The Algorithm

All edges are covered by the four node sets $W_1, W_2, W_3, W_4$

Arrange the node sets in the order of their labels:

$-2.5 \quad -.77 \quad 1 \quad 3.3$

$\{3,5\} \rightarrow \{1,2,3\} \rightarrow \{0,2,3\} \rightarrow \{0,4\}$
The Algorithm

All edges are covered by the four node sets \( W_1, W_2, W_3, W_4 \)

Arrange the node sets in the order of their labels:

\[
-2.5 \quad -.77 \quad 1 \quad 3.3
\]

\[
\{3,5\} \quad \{1,2,3\} \quad \{0,2,3\} \quad \{0,4\}
\]

Property 3 of tree decompositions is satisfied.
The Algorithm

All edges are covered by the four node sets $W_1, W_2, W_3, W_4$

Arrange the node sets in the order of their labels:

$-2.5 \quad -0.77 \quad 1 \quad 3.3$

$\{3, 5\} \quad \{1, 2, 3\} \quad \{0, 2, 3\} \quad \{0, 4\}$

Property 3 of tree decompositions is satisfied. And the tree looks optimal.
```python
def G(graph_number):
    print ('Graph'), graph_number, (' :')
    import networkx, networkx
    graph_number = graph_number
    import graph
    G = networkx.Graph(graph_number)
    L = G.laplacian_matrix()
    eigs = L.eigenvalues()
    n = L.nrows()
    connected = 1
    null_vector = matrix(1,n)
    eigs.sort()
    s0 = eigs[0]
    for s in range(n):
        if s[0] == 1:
            for i in range(n):
                if L[i, j] == null_vector:
                    connected = 0
            if connected == 1:
                print ('')
                print ('LaPlacian matrix is')
                print ('')
                n_copy = n
                digit_accuracy = 6
                # Sort the set to get lambda 2
                eigs.sort()
                # Get lambda 2 by first extracting the 2nd slot of Eigs which contains the eigenvalue, vector and multiplicity
                s = eigs[1]
                print ('')
                print ('Lambda2 is'), s[0]
                G.show()
                if s[2] == 1:
                    fvec_shell = s[1]
                    fvec = fvec_shell[0]
                    fvec1 = matrix(1,n)
                    for y in range(n):
                        print ('y is'), y
                        fvec[0,y] = round(fvec[0,y],4)
                        print ('Fiedler vector is'), fvec[0,n(digits = 5)]
                        print ('')
                # We make each magnitude an ordered pair with its node and sort it
```
Some Limitations of the Algorithm

Although the algorithm will always produce a tree satisfying properties 1 and 2, it will not always satisfy property 3.
Some Limitations of the Algorithm

Although the algorithm will always produce a tree satisfying properties 1 and 2, it will not always satisfy property 3.

Outcome for a 10-vertex example:

\{0,3,7\} \rightarrow \{0,4\} \rightarrow \{4,7\} \rightarrow \{2,4,5\} \rightarrow \{2,5,9\} \rightarrow \{5,8,9\} \rightarrow \{2,6,7,8\} \rightarrow \{1,6\}
Some Limitations of the Algorithm

Although the algorithm will always produce a tree satisfying properties 1 and 2, it will not always satisfy property 3.

Outcome for a 10-vertex example:

\{0,3,7\} \rightarrow \{0,4\} \rightarrow \{4,7\} \rightarrow \{2,4,5\} \rightarrow \{2,5,9\} \rightarrow \{5,8,9\} \rightarrow \{2,6,7,8\} \rightarrow \{1,6\}

2 and 7 are problem vertices.
Some Limitations of the Algorithm

Although the algorithm will always produce a tree satisfying properties 1 and 2, it will not always satisfy property 3.

Outcome for a 10-vertex example:

\{0,3,7\} — \{0,4\} — \{4,7\} — \{2,4,5\} — \{2,5,9\} — \{5,8,9\}—\{2,6,7,8\}—\{1,6\}  

2 and 7 are problem vertices.

So we throw them in to produce a tree satisfying property 3.

\{0,3,7\} — \{0,4,7\} — \{4,7\} — \{2,4,5,7\} — \{2,5,7,9\} — \{2,5,7,8,9\}—\{2,6,7,8\}—\{1,6\}
Some Limitations of the Algorithm

Although the algorithm will always produce a tree satisfying properties 1 and 2, it will not always satisfy property 3.

Outcome for a 10-vertex example:

\[
\{0,3,7\} \rightarrow \{0,4\} \rightarrow \{4,7\} \rightarrow \{2,4,5\} \rightarrow \{2,5,9\} \rightarrow \{5,8,9\} \rightarrow \{2,6,7,8\} \rightarrow \{1,6\}
\]

2 and 7 are problem vertices.

So we throw them in to produce a tree satisfying property 3.

\[
\{0,3,7\} \rightarrow \{0,4,7\} \rightarrow \{4,7\} \rightarrow \{2,4,5,7\} \rightarrow \{2,5,7,9\} \rightarrow \{2,5,7,8,9\} \rightarrow \{2,6,7,8\} \rightarrow \{1,6\}
\]

Two of these sets are now subsets of other sets in the chain.
Some Limitations of the Algorithm

Although the algorithm will always produce a tree satisfying properties 1 and 2, it will not always satisfy property 3.

Outcome for a 10-vertex example:
{0,3,7} — {0,4} — {4,7} — {2,4,5} — {2,5,9} — {5,8,9} — {2,6,7,8} — {1,6}

2 and 7 are problem vertices.

So we throw them in to produce a tree satisfying property 3.
{0,3,7} — {0,4,7} — {4,7} — {2,4,5,7} — {2,5,7,9} — {2,5,7,8,9} — {2,6,7,8} — {1,6}

Two of these sets are now subsets of other sets in the chain.

So we throw them out.
{0,3,7} — {0,4,7} — {2,4,5,7} — {2,5,7,8,9} — {2,6,7,8} — {1,6}
Some Limitations of the Algorithm

Although the algorithm will always produce a tree satisfying properties 1 and 2, it will not always satisfy property 3.

Outcome for a 10-vertex example:

\{0,3,7\} — \{0,4\} — \{4,7\} — \{2,4,5\} — \{2,5,9\} — \{5,8,9\} — \{2,6,7,8\} — \{1,6\}

2 and 7 are problem vertices.

So we throw them in to produce a tree satisfying property 3.

\{0,3,7\} — \{0,4,7\} — \{4,7\} — \{2,4,5,7\} — \{2,5,7,9\} — \{2,5,7,8,9\} — \{2,6,7,8\} — \{1,6\}

Two of these sets are now subsets of other sets in the chain.

So we throw them out.

\{0,3,7\} — \{0,4,7\} — \{2,4,5,7\} — \{2,5,7,8,9\} — \{2,6,7,8\} — \{1,6\}

This turns out to be a very reasonable tree decomposition.
Some Limitations of the Algorithm

Although the algorithm will always produce a tree satisfying properties 1 and 2, it will not always satisfy property 3.

Outcome for a 10-vertex example:

\{0,3,7\} — \{0,4\} — \{4,7\} — \{2,4,5\} — \{2,5,9\} — \{5,8,9\} — \{2,6,7,8\} — \{1,6\}

2 and 7 are problem vertices.

So we throw them in to produce a tree satisfying property 3.

\{0,3,7\} — \{0,4,7\} — \{4,7\} — \{2,4,5,7\} — \{2,5,7,9\} — \{2,5,7,8,9\} — \{2,6,7,8\} — \{1,6\}

Two of these sets are now subsets of other sets in the chain.

So we throw them out.

\{0,3,7\} — \{0,4,7\} — \{2,4,5,7\} — \{2,5,7,8,9\} — \{2,6,7,8\} — \{1,6\}

This turns out to be a very reasonable tree decomposition.

But we have no theorem that says this is the case for other graphs.
The only kind of tree this algorithm can produce is a path.
The only kind of tree this algorithm can produce is a path. But this is not as bad as it seems.
Some Limitations of the Algorithm

The only kind of tree this algorithm can produce is a path. But this is not as bad as it seems. If the multiplicity of $\lambda_2 = k > 1$, then one could use $k$ eigenvectors to produce a tree decomposition.
Some Limitations of the Algorithm

The only kind of tree this algorithm can produce is a path.
But this is not as bad as it seems.
If the multiplicity of $\lambda_2 = k > 1$, then one could use $k$ eigenvectors to produce a tree decomposition.
We have not yet implemented the algorithm in the higher multiplicity cases.
Some Limitations of the Algorithm

The only kind of tree this algorithm can produce is a path.
But this is not as bad as it seems.
If the multiplicity of $\lambda_2 = k > 1$, then one could use $k$ eigenvectors to produce a tree decomposition.
We have not yet implemented the algorithm in the higher multiplicity cases.
Remarkably, it is frequently the case that a graph that “looks” 2–dimensional - so that a path would be an inappropriate representation - has multiplicity($\lambda_2$) > 1.
Some Limitations of the Algorithm

The only kind of tree this algorithm can produce is a path. But this is not as bad as it seems. If the multiplicity of $\lambda_2 = k > 1$, then one could use $k$ eigenvectors to produce a tree decomposition. We have not yet implemented the algorithm in the higher multiplicity cases.

Remarkably, it is frequently the case that a graph that “looks” 2–dimensional - so that a path would be an inappropriate representation - has multiplicity($\lambda_2$) > 1.

Example
Some Limitations of the Algorithm

The only kind of tree this algorithm can produce is a path. But this is not as bad as it seems.
If the multiplicity of $\lambda_2 = k > 1$, then one could use $k$ eigenvectors to produce a tree decomposition.
We have not yet implemented the algorithm in the higher multiplicity cases.
Remarkably, it is frequently the case that a graph that “looks” 2–dimensional - so that a path would be an inappropriate representation - has multiplicity($\lambda_2$) > 1.

Example

```
    ∗
   /|
  /  |
 ∗   ∗
 /|
/  |
 ∗   ∗
```

multiplicity($\lambda_2$) = 2.
Extremely interesting topic for me.
Extremely interesting topic for me. An instance is the intriguing result

**Theorem (Mowshowitz; Petersdorf and Sachs)**

> If $G$ is a graph with an automorphism of order greater than 2, then the adjacency matrix of $G$ has a multiple eigenvalue.
Extremely interesting topic for me. An instance is the intriguing result

Theorem (Mowshowitz; Petersdorf and Sachs)

If $G$ is a graph with an automorphism of order greater than 2, then the adjacency matrix of $G$ has a multiple eigenvalue.

Same proof works for the Laplacian matrix.
Extremely interesting topic for me. An instance is the intriguing result

**Theorem (Mowshowitz; Petersdorf and Sachs)**

*If* $G$ *is a graph with an automorphism of order greater than 2, then the adjacency matrix of* $G$ *has a multiple eigenvalue.*

Same proof works for the Laplacian matrix.

We are interested in the eigenvalue $\lambda_2$ of the Laplacian matrix, but the proof gives no clue as to which eigenvalue might be multiple.
Extremely interesting topic for me. An instance is the intriguing result

**Theorem (Mowshowitz; Petersdorf and Sachs)**

*If G is a graph with an automorphism of order greater than 2, then the adjacency matrix of G has a multiple eigenvalue.*

Same proof works for the Laplacian matrix.

We are interested in the eigenvalue $\lambda_2$ of the Laplacian matrix, but the proof gives no clue as to which eigenvalue might be multiple.

The proof assumes that all eigenvalues are simple and proceeds to a contradiction.
Example: The Icosahedral Graph

By the theorem, there must be a multiple eigenvalue. We stumbled on a way to find all the eigenvalues of any graph with an automorphism via smaller matrices, sometimes much smaller.
Example: The Icosahedral Graph

By the theorem, there must be a multiple eigenvalue.
By the theorem, there must be a multiple eigenvalue.

We stumbled on a way to find all the eigenvalues of any graph with an automorphism via smaller matrices, sometimes much smaller.
The icosahedral graph

The key to seeing it is to number the vertices in an optimal order, beginning with one vertex from each orbit.
The icosahedral graph

The key to seeing it is to number the vertices in an optimal order, beginning with one vertex from each orbit.

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{bmatrix}
\]
The icosahedral graph

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
\end{bmatrix}
\]
The icosahedral graph

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{bmatrix}
\]

is a block circulant matrix.

The list of eigenvalues of this matrix are the union of the lists of eigenvalues of the three $4 \times 4$ matrices:

\[
B + C + C^T,
\]

\[
B + \omega C + \omega^2 C^T,
\]

\[
B + \omega^2 C + \omega C^T,
\]

where $\omega = e^{2\pi i/3}$.

of the form

\[
\begin{bmatrix}
B & C & C^T \\
C^T & B & C \\
C & C^T & B
\end{bmatrix}
\]
The icosahedral graph

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
\end{bmatrix}
\]

of the form

\[
\begin{bmatrix}
B & C & C^T \\
C^T & B & C \\
C & C^T & B \\
\end{bmatrix}
\]

a block circulant matrix.

The list of eigenvalues of this matrix are the union of the lists of eigenvalues of the three \(4 \times 4\) matrices:
The icosahedral graph

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
\end{bmatrix}
\]

of the form

\[
\begin{bmatrix}
B & C & C^T \\
C^T & B & C \\
C & C^T & B \\
\end{bmatrix},
\]

a block circulant matrix.

The list of eigenvalues of this matrix are the union of the lists of eigenvalues of the three \(4 \times 4\) matrices:

\[B + C + C^T,\]
The icosahedral graph

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 
\end{bmatrix}
\]

is a block circulant matrix.

The list of eigenvalues of this matrix are the union of the lists of eigenvalues of the three \(4 \times 4\) matrices:

\[
B + C + C^T, \quad B + \omega C + \omega^2 C^T
\]
The icosahedral graph of the form

\[
\begin{bmatrix}
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
\end{bmatrix}
\]

of the form

\[
\begin{bmatrix}
B & C & C^T \\
C^T & B & C \\
C & C^T & B \\
\end{bmatrix}
\]

a block circulant matrix.

The list of eigenvalues of this matrix are the union of the lists of eigenvalues of the three $4 \times 4$ matrices:

\[B + C + C^T, \quad B + \omega C + \omega^2 C^T, \quad B + \omega^2 C + \omega C^T,\]

where $\omega = e^{2\pi i/3}$. 

Wayne Barrett (BYU)  
Fiedler Vector  
June 4, 2015  
32 / 34
The icosahedral graph

\[ B_0 = B + C + C^T = \begin{bmatrix} 2 & 1 & 2 & 0 \\ 1 & 0 & 2 & 2 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 1 & 2 \end{bmatrix} \text{ with eigenvalues } 5, \sqrt{5}, -1, -\sqrt{5}, \]
The icosahedral graph

\[ B_0 = B + C + C^T = \begin{bmatrix} 2 & 1 & 2 & 0 \\ 1 & 0 & 2 & 2 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 1 & 2 \end{bmatrix} \]

with eigenvalues 5, \( \sqrt{5} \), \(-1\), \(-\sqrt{5}\).

\[ B_1 = B + \omega C + \omega^2 C^T \text{ with eigenvalues } \sqrt{5}, -1, -1, -\sqrt{5}, \]

Explains why the graph has many multiple eigenvalues, not just some.

It is easy to generalize this to any automorphism of any graph all of whose orbits have the same size.

The longer the orbits, the smaller the matrices that yield the eigenvalues of the original.
The icosahedral graph

\[ B_0 = B + C + C^T = \begin{bmatrix} 2 & 1 & 2 & 0 \\ 1 & 0 & 2 & 2 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 1 & 2 \end{bmatrix} \] with eigenvalues 5, \( \sqrt{5} \), \(-1\), \(-\sqrt{5}\),

\[ B_1 = B + \omega C + \omega^2 C^T \] with eigenvalues \( \sqrt{5}, -1, -1, -\sqrt{5} \),

and \( B_2 = B + \omega^2 C + \omega C^T = B_1^T \) with the same eigenvalues.
The icosahedral graph

\[
B_0 = B + C + C^T = \begin{bmatrix}
2 & 1 & 2 & 0 \\
1 & 0 & 2 & 2 \\
2 & 2 & 0 & 1 \\
0 & 2 & 1 & 2
\end{bmatrix}
\text{ with eigenvalues } 5, \sqrt{5}, -1, -\sqrt{5},
\]

\[
B_1 = B + \omega C + \omega^2 C^T \text{ with eigenvalues } \sqrt{5}, -1, -1, -\sqrt{5},
\]

and \( B_2 = B + \omega^2 C + \omega C^T = B_1^T \) with the same eigenvalues.

Explains why the graph has many multiple eigenvalues, not just some.
The icosahedral graph

\[ B_0 = B + C + C^T = \begin{bmatrix} 2 & 1 & 2 & 0 \\ 1 & 0 & 2 & 2 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 1 & 2 \end{bmatrix} \] with eigenvalues 5, \( \sqrt{5} \), \(-1\), \(-\sqrt{5}\),

\[ B_1 = B + \omega C + \omega^2 C^T \] with eigenvalues \( \sqrt{5} \), \(-1\), \(-1\), \(-\sqrt{5}\),

and \( B_2 = B + \omega^2 C + \omega C^T = B_1^T \) with the same eigenvalues.

Explains why the graph has many multiple eigenvalues, not just some.

It is easy to generalize this to any automorphism of any graph all of whose orbits have the same size.
The icosahedral graph

\[ B_0 = B + C + C^T = \begin{bmatrix} 2 & 1 & 2 & 0 \\ 1 & 0 & 2 & 2 \\ 2 & 2 & 0 & 1 \\ 0 & 2 & 1 & 2 \end{bmatrix} \] with eigenvalues 5, \( \sqrt{5} \), \(-1\), \(-\sqrt{5}\),

\[ B_1 = B + \omega C + \omega^2 C^T \] with eigenvalues \( \sqrt{5} \), \(-1\), \(-1\), \(-\sqrt{5}\),

and \( B_2 = B + \omega^2 C + \omega C^T = B_1^T \) with the same eigenvalues.

Explains why the graph has many multiple eigenvalues, not just some.

It is easy to generalize this to any automorphism of any graph all of whose orbits have the same size.

The longer the orbits, the smaller the matrices that yield the eigenvalues of the original.
The icosahedral graph

Since this talk was about $\lambda_2$ (and its Fiedler vector) we ought to end by mentioning its multiplicity for the icosahedral graph.

Since this graph is 5-regular, the eigenvalues of its Laplacian matrix are just 5 minus the eigenvalues of the adjacency matrix:

$$0, (5 - \sqrt{5})^3, 6, (5 + \sqrt{5})^3$$

$\lambda_2 = 5 - \sqrt{5}$ has multiplicity 3.

Makes sense, because the icosahedral graph is really the icosahedron, which is 3-dimensional.
The icosahedral graph

Since this talk was about $\lambda_2$ (and its Fiedler vector) we ought to end by mentioning its multiplicity for the icosahedral graph.

Since this graph is 5-regular, the eigenvalues of its Laplacian matrix are just 5 minus the eigenvalues of the adjacency matrix:

$$\lambda_2 = 5 - \sqrt{5}$$ has multiplicity 3.

Makes sense, because the icosahedral graph is really the icosahedron, which is 3-dimensional.
The icosahedral graph

Since this talk was about $\lambda_2$ (and its Fiedler vector) we ought to end by mentioning its multiplicity for the icosahedral graph.

Since this graph is 5-regular, the eigenvalues of its Laplacian matrix are just 5 minus the eigenvalues of the adjacency matrix:

$$0, (5 - \sqrt{5})^3, 6^5, (5 + \sqrt{5})^3$$
The icosahedral graph

Since this talk was about $\lambda_2$ (and its Fiedler vector) we ought to end by mentioning its multiplicity for the icosahedral graph.

Since this graph is 5-regular, the eigenvalues of its Laplacian matrix are just 5 minus the eigenvalues of the adjacency matrix:

$$0, (5 - \sqrt{5})^3, 6^5, (5 + \sqrt{5})^3$$

$\lambda_2 = 5 - \sqrt{5}$ has multiplicity 3.

Makes sense, because the icosahedral graph is really the icosahedron, which is 3-dimensional.
The icosahedral graph

Since this talk was about $\lambda_2$ (and its Fiedler vector) we ought to end by mentioning its multiplicity for the icosahedral graph.

Since this graph is 5-regular, the eigenvalues of its Laplacian matrix are just 5 minus the eigenvalues of the adjacency matrix:

$$0, (5 - \sqrt{5})^3, 6^5, (5 + \sqrt{5})^3$$

$\lambda_2 = 5 - \sqrt{5}$ has multiplicity 3.

Makes sense, because the icosahedral graph is really the icosahedron, which is 3-dimensional.
The icosahedral graph

Since this talk was about $\lambda_2$ (and its Fiedler vector) we ought to end by mentioning its multiplicity for the icosahedral graph.

Since this graph is 5-regular, the eigenvalues of its Laplacian matrix are just 5 minus the eigenvalues of the adjacency matrix:

$$0, (5 - \sqrt{5})^3, 6^5, (5 + \sqrt{5})^3$$

$\lambda_2 = 5 - \sqrt{5}$ has multiplicity 3.

Makes sense, because the icosahedral graph is really the icosahedron, which is 3-dimensional.