# Fiedler Vector Method

Sagnik Dutta, Nilava Metya, Sagnik Mukherjee
**Chennai Mathematical Institute**

May 24, 2022

## Contents

# 1 Introduction

The Fiedler vector of a graph, namely the eigenvector corresponding to the second smallest eigenvalue of a graph's Laplacian matrix, plays an important role in spectral graph theory with applications in problems such as graph bi-partitioning and envelope reduction. The problem we want to discuss is as follows:

**Statement of the Problem:** Given a finite simple connected graph $G = (V, E)$, partition the vertex set as $V = V_1 \sqcup V_2$ such that the number of cut edges is minimized while keeping $|V_1| \simeq |V_2|$.

Here a "cut" edge simply means an edge which has one endpoint in $V_1$ and the other in $V_2$.
In the presentation, we discussed the Fielder Vector method, which was introduced by Fiedler in [1] as an efficient way of solving this problem. Our presentation is based on the survey [3] written by Brian Slininger where he discussed in detail about the Fiedler vector method and its shortcomings, which we also have discussed in our presentation.

First, let us recall some basic facts from Spectral Graph theory.

## 1.1 Basic Facts

Given a graph $G = (V, E)$, we can define several matrices to record the data of the graph (after labeling each vertex by some integer between 1 and $n := |V|$), for example

- *Adjacency Matrix*: A matrix $A$ of size $|V| \times |V|$, such that $A_{ij} = \mathbf{1}_{\{i,j\}} \in E$
- *Degree Matrix*: A matrix $D$ of size $|V| \times |V|$ such that $D_{ij} = \delta_i^j \cdot \deg(i)$.
- *Incidence Matrix*: A matrix $H$ of size $|V| \times |E|$ such that

$$H_{v,e} = \begin{cases} 1 & \text{if } v \text{ is adjacent to } e \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

- *Laplacian*: The Laplacian is defined as $\mathcal{L} := D - A$

Laplacian of a given graph is going to be the most important matrix for us in this presentation. Let us recall some basic facts about the Laplacian too:

- $\mathcal{L}$ is symmetric and hence its all eigenvalues are real.
- $\mathcal{L} = HH^t$.
- The eigenvalues of $\mathcal{L}$ are non-negative. Hence $\mathcal{L}$ is positive semi-definite (PSD).

*Proof.* Let $\lambda$ be an eigenvalue of $\mathcal{L}$ with corresponding eigenvector to be $v$. Then $\mathcal{L}v = \lambda v \implies \langle \mathcal{L}v, v \rangle = \langle \lambda v, v \rangle = \lambda \|v\|^2$. But $\langle \mathcal{L}v, v \rangle = v^t \mathcal{L} v = v^t HH^t v = \left\|H^t v\right\|^2$. This means $\lambda \|v\|^2 = \left\|H^t v\right\|^2 \implies \lambda \geq 0$. ■

- 0 is an eigenvalue of $\mathcal{L}$ with corresponding eigenvector $(1, 1, \cdots, 1)$.

  *Proof.* The $i^{\text{th}}$ entry of $\mathcal{L} \cdot (1, \cdots, 1)$ is simply $\deg(i) - \sum_{j:\{i,j\}\in E} 1 = 0$. ■

# 2 Developing the solution

We now try to develop the solution. Our first goal is to mathematically formulate the problem.

## 2.1 Mathematically modelling the problem

Let the graph be $G := (V, E)$. Then the problem at hand is to define a partition of $V$ in to $V = V_1 \sqcup V_2$ such that the number of cut edges is minimum. But also we have a constraint, i.e. $|V_1| = |V_2|$. So we have to mathematically model two things;

1. Find some mathematical formula for the number of cut edges for a given partition of the vertex set.
2. (The constraint) Find some mathematical formula to describe the equipartition of the vertex set.

First let us try to mathematically describe the equipartition of the vertex set.
Suppose the vertex set $V$ has a partition $P = (V_1, V_2)$. Then assign the value $1$ to each vertex in $V_1$ and $-1$ to each vertex in $V_2$. This way we assign to the $i_{\text{th}}$ vertex a value $x_i \in \{\pm 1\}$. This gives us a vector $\boldsymbol{x} := (x_i)_{i=1}^n$. Now note the following:

**Lemma 2.1.** *$P$ is a equipartition of $V$ iff* $\sum_{i=1}^{n} x_i = 0 \equiv \boldsymbol{x} \perp (1, 1, \cdots, 1)$.

*Proof.* We have an equipartition *iff* $\big|\{i : x_i = 1\}\big| = \big|\{i : x_i = -1\}\big|$ *iff* $\sum_{i=1}^{n} x_i = 0$. ■

Thus we have the **mathematical formulation of the constraint** (1)

$$\text{An equipartition of the vertex set} \equiv \sum_{i=1}^{n} x_i = 0 \equiv \boldsymbol{x} \perp (1, 1, \cdots, 1).$$

We can have an additional constraint:

$$\sum_{i=1}^{n} x_i^2 = n$$

Putting these together, we have the following set of constraints:

$$\sum_{i=1}^{n} x_i = 0 \equiv \boldsymbol{x} \perp (1, 1, \cdots, 1)$$

$$\text{and} \sum_{i=1}^{n} x_i^2 = n$$

Now let us find the number of cut edges for a given partition of the vertex set.
So let $P = (V_1, V_2)$ be a partition of $V$. Then note that the number of cut edges (technical name: cut value)

$$\mathfrak{C}(P) = \sum_{\substack{\{i,j\}\in E \\ x_i \neq x_j}} 1$$

We shall find an alternate and more useful version of $\mathfrak{C}(P)$.

$$\begin{aligned} 4 \cdot \mathfrak{C}(P) = 4 \cdot \sum_{\substack{\{i,j\}\in E \\ x_i \neq x_j}} 1 &= \sum_{\substack{\{i,j\}\in E \\ x_i \neq x_j}} (\pm 2)^2 + \sum_{\substack{\{i,j\}\in E \\ x_i = x_j}} 0^2 \\ &= \sum_{\substack{\{i,j\}\in E \\ x_i \neq x_j}} (x_i - x_j)^2 + \sum_{\substack{\{i,j\}\in E \\ x_i = x_j}} (x_i - x_j)^2 \\ &= \sum_{\{i,j\}\in E} (x_i - x_j)^2 \end{aligned}$$

With this formulation our problem is now as follows.

**Problem:** Minimize $\sum_{\{i,j\}\in E} (x_i - x_j)^2$ subjected to the constraints $\sum_{i=1}^{n} x_i = 0 \equiv \boldsymbol{x} \perp (1, \cdots, 1)$ and $\sum_{i=1}^{n} x_i^2 = n$.
Let's reformulate the above statement using the language of spectral graph theory.

## 2.2 Spectral Graph Theory

We can rewrite the expression $\sum_{\{i,j\}\in E}(x_i - x_j)^2$ as follows:

$$\begin{aligned} \sum_{\{i,j\}\in E} (x_i - x_j)^2 = \sum_{\{i,j\}\in E} x_i^2 + x_j^2 - 2x_i x_j &= \sum_i \deg(i) x_i^2 - 2 \sum_{\{i,j\}\in E} x_i x_j \\ &= \boldsymbol{x}^t D \boldsymbol{x} - \boldsymbol{x}^t A \boldsymbol{x} = \boldsymbol{x}^t (D - A) \boldsymbol{x} = \boldsymbol{x}^t \mathcal{L} \boldsymbol{x} \end{aligned}$$

Thus our new version of the problem is

**Problem:** (1st Version) Find $\boldsymbol{x}$ which minimizes $\boldsymbol{x}^t \mathcal{L} \boldsymbol{x}$ subjected to the constraints $\|\boldsymbol{x}\|^2 = n$ and $\langle \boldsymbol{x}, (1, \cdots, 1) \rangle = 0$.
We can have an alternate and more useful version of the above.

**Problem:** (2nd Version) Find $\underset{\substack{\boldsymbol{x}\perp(1,1,\cdots,1) \\ \|\boldsymbol{x}\|\neq 0}}{\operatorname{argmin}} \dfrac{\boldsymbol{x}^t \mathcal{L} \boldsymbol{x}}{\|\boldsymbol{x}\|^2} = \underset{\substack{\boldsymbol{x}\perp(1,\cdots,1) \\ \|\boldsymbol{x}\|\neq 0}}{\operatorname{argmin}} R_{\mathcal{L}}(\boldsymbol{x})$
where $R_{\mathcal{L}}(\boldsymbol{x})$ is the Rayleigh Quotient.

Now recall the following fact from the lecture.

**Theorem 2.2.** *If $A_{n\times n}$ is a real PSD matrix with eigenvalues $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ with eigenbasis $(v_1, \cdots, v_n)$ then*

$$\min_{\substack{\boldsymbol{x}\perp\langle v_1, v_2, \cdots, v_k\rangle \\ \|\boldsymbol{x}\|\neq 0}} R_{\mathcal{L}}(\boldsymbol{x}) = \lambda_{k+1}$$

Note that for $\mathcal{L}$, $\lambda_1 = 0$ and $v_1 = (1, \cdots, 1)$, so that

$$\min_{\substack{\boldsymbol{x}\perp(1,1,\cdots,1) \\ \|\boldsymbol{x}\|\neq 0}} R_{\mathcal{L}}(\boldsymbol{x}) = \lambda_2$$

That is how the second eigenvalue and hence the second eignevector (called the Fiedler Vector) of the Laplacian comes to picture. Based on this, we design the following algorithm of finding $\boldsymbol{x}$ and hence the desired partition;

**Algorithm:** Look at the $i^{\text{th}}$ component of the Fiedler vector. If it is $< 0$ put $i^{\text{th}}$ in $V_1$, otherwise put it in $V_2$.

# 3 The Main Theorem

First we state a few lemmas and theorems that we will need to prove the main theorem of [3]:

**Theorem 3.1** (Cauchy Interlacing theorem)**.** *Let $A_{n\times n}$ be a symmetric matrix and $B_{m\times m}$ be a principal submatrix of $A$. Further, let the eigenvalues of $A$ be $\lambda_1 \leq \cdots \leq \lambda_n$ and the eigenvalues of $B$ be $\beta_1 \leq \cdots \leq \beta_m$. Then, for all $k \leq m$, the matrix $A$ has at least $k$ eigenvalues less than or equal to $\beta_k$.*

A proof of the above theorem can be found in [2].

**Lemma 3.2.** *Let $A_{n\times n}$ be a matrix and $X_{n\times n}$ be a nonsingular matrix. Then, $X^TAX$ is symmetric positive-definite if and only if $A$ is symmetric positive-definite.*

*Proof.*

$$\begin{aligned}
A \text{ is positive-definite} &\implies \forall v \in \mathbb{R}^n (v \neq 0), (Xv)^T A(Xv) > 0 \\
&\implies \forall v \in \mathbb{R}^n (v \neq 0), v^T(X^TAX)v > 0 \\
&\implies X^TAX \text{ is positive-definite} \\
X^TAX \text{ is positive-definite} &\implies \forall v \in \mathbb{R}^n (v \neq 0), (X^{-1}v)^T(X^TAX)(X^{-1}v) > 0 \\
&\implies \forall v \in \mathbb{R}^n (v \neq 0), v^TAv > 0 \\
&\implies A \text{ is positive-definite}
\end{aligned}$$

■

**Lemma 3.3.** *Let $A$ be an $n \times n$ symmetric matrix. If $\rho(A) < 1$ i.e. the spectral radius of $A$ is less than 1, then $(I_n - A)$ is nonsingular and $(I_n - A)^{-1} = \sum_{l=0}^{\infty} A^l$.*

*Proof.* As $A$ is symmetric, from spectral theorem it follows that $A$ has an eigenvalue decomposition $A = Q\Lambda Q^T$ where $\Lambda$ is a diagonal matrix containing the eigenvalues of $A$ as diagonal entries and $QQ^T = I$. Then $\sum\limits_{l=0}^{\infty} A^l = Q(\sum\limits_{l=0}^{\infty} \Lambda^l)Q^T$ is convergent as the entries of $\Lambda$ have absolute value less than 1. Now notice that $(I-A)\cdot\sum\limits_{l=0}^{m} A^l = I - A^{m+1}$ since it is a telescoping series. As $m$ tends to infinity, the matrix $A^{m+1} = Q\Lambda^{m+1}Q^T$ tends to the zero matrix, again because of the fact that the entries of $\Lambda$ have absolute value less than 1. Hence $(I-A)\cdot\sum\limits_{l=0}^{\infty} A^l = \cdot\sum\limits_{l=0}^{\infty} A^l\cdot(I-A) = \lim\limits_{m\to\infty}(I-A^{m+1}) = I$ which implies $(I_n - A)^{-1} = \sum_{l=0}^{\infty} A^l$. ■

Now we are ready to prove the main theorem.

**Theorem 3.4** (Fiedler's theorem of connectivity of spectral graph partitions)**.** *Let $G = (V, E)$ be a connected graph and let $\boldsymbol{x} = (x_1, \cdots, x_n)$ is the Fiedler vector for the Laplacian of this graph. Let $V_1 = \{\, i : x_i > 0 \,\}$ and $V_2 = V \setminus V_1$. Let $G_i$ be the subgraph induced by $V_i$. Then $G_i$ are both connected.*

*Proof.* Relabel the vertices in a way such that $V_1 = \{1, \cdots, k\}, V_2 = \{k+1, \cdots, n\}$. For the sake of contradiction, suppose $G_1$ is not connected. Then we can partition $V_1$ into two subsets $\{1, \cdots, t\}$ and $\{t+1, \cdots, k\}$ such that there are no edges between these two subsets of vertices. Hence the Laplacian looks like

$$\mathcal{L} = \begin{bmatrix} L_{11} & \mathbf{0} & L_{13} \\ \mathbf{0} & L_{22} & L_{23} \\ L_{13}^T & L_{23}^T & L_{33} \end{bmatrix}.$$

By nature of $\mathcal{L}$, the entries of $L_{13}, L_{23}$ are non-positive. Write a similar block form for

$$\boldsymbol{x} = \begin{bmatrix} (\boldsymbol{x}_1)_{t\times 1} \\ (\boldsymbol{x}_2)_{(k-t)\times 1} \\ (\boldsymbol{x}_3)_{(n-k)\times 1} \end{bmatrix}.$$

where components of $\boldsymbol{x}_1, \boldsymbol{x}_2$ are positive and those of $\boldsymbol{x}_3$ are negative.
As $\boldsymbol{x}$ is the eigenvector of $\mathcal{L}$ corresponding to the eigenvalue $\lambda_2$, we have $\mathcal{L}\boldsymbol{x} = \lambda_2\boldsymbol{x}$
$\implies L_{11}\boldsymbol{x}_1 + L_{13}\boldsymbol{x}_3 = \lambda_2\boldsymbol{x}_1$
Fix any positive real $\epsilon > 0$. Then adding $\epsilon\boldsymbol{x}_1$ to both sides of the above equation, we get

$$(\epsilon I + L_{11})\boldsymbol{x}_1 + L_{13}\boldsymbol{x}_3 = (\epsilon + \lambda_2)\boldsymbol{x}_1$$

We will need the following lemma which we will prove later:

**Lemma 3.5.** *$(\epsilon I + L_{11})$ is invertible and its inverse $Y$ is a positive matrix.*

Assuming this lemma and multiplying both sides of the above equation by $Y = (\epsilon I + L_{11})^{-1}$, we get

$$\begin{aligned} \boldsymbol{x}_1 + YL_{13}\boldsymbol{x}_3 &= (\epsilon + \lambda_2)Y\boldsymbol{x}_1 \\ \implies \boldsymbol{x}_1^T\boldsymbol{x}_1 + \boldsymbol{x}_1^TYL_{13}\boldsymbol{x}_3 &= (\epsilon + \lambda_2)\boldsymbol{x}_1^TY\boldsymbol{x}_1 \\ \implies (\epsilon + \lambda_2)\frac{\boldsymbol{x}_1^TY\boldsymbol{x}_1}{\boldsymbol{x}_1^T\boldsymbol{x}_1} &= 1 + \frac{\boldsymbol{x}_1^TYL_{13}\boldsymbol{x}_3}{\boldsymbol{x}_1^T\boldsymbol{x}_1} > 1 \end{aligned}$$

where the last inequality follows from the facts that $x_1, Y$ are positive matrices, $x_3$ is a negative vector and $L_{13}$ is a non-positive matrix, but not the zero matrix (because if $L_{13}$ is the zero matrix, then the graph $G$ itself is disconnected, which is a contradiction). Let $\lambda_t(Y)$ denote the $t$-th eigenvalue i.e. the largest eigenvalue of $Y$ and let $\lambda_1(\epsilon I + L_{11})$ denote the smallest eigenvalue of $\epsilon I + L_{11}$. Then we have

$$\begin{aligned} (\epsilon + \lambda_2)\lambda_t(Y) = (\epsilon + \lambda_2)\max_{v\neq 0}\frac{v^TYv}{v^Tv} &\geq (\epsilon + \lambda_2)\frac{\boldsymbol{x}_1^TY\boldsymbol{x}_1}{\boldsymbol{x}_1^T\boldsymbol{x}_1} > 1 \\ \implies \lambda_1(\epsilon I + L_{11}) = 1/\lambda_t(Y) &< \epsilon + \lambda_2 \\ \implies \epsilon + \lambda_1(L_{11}) &< \epsilon + \lambda_2 \end{aligned}$$

Hence $\lambda_1(L_{11}) < \lambda_2$. Similarly, $\lambda_1(L_{22}) < \lambda_2$. If the eigenvectors corresponding to $\lambda_1(L_{11})$ and $\lambda_1(L_{22})$ are $v_1$ and $v_2$ respectively, then

$$\begin{bmatrix} L_{11} & 0 \\ 0 & L_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ 0 \end{bmatrix} = \lambda_1(L_{11}) \begin{bmatrix} v_1 \\ 0 \end{bmatrix} \qquad \text{and} \qquad \begin{bmatrix} L_{11} & 0 \\ 0 & L_{22} \end{bmatrix} \begin{bmatrix} 0 \\ v_2 \end{bmatrix} = \lambda_1(L_{22}) \begin{bmatrix} 0 \\ v_2 \end{bmatrix} .$$

Thus the principal submatrix $\begin{bmatrix} L_{11} & 0 \\ 0 & L_{22} \end{bmatrix}$ of $\mathcal{L}$ has two eigenvalues less than $\lambda_2$. Then by Theorem 3.1, $\mathcal{L}$ has two eigenvalues less than $\lambda_2$, which is a contradiction! ■

*Proof of Lemma 3.5.* Write $\epsilon I + L_{11} = D - N$ where $D$ is a non-negative diagonal matrix containing the diagonal entries of $\epsilon I + L_{11}$ and $N$ contains all the off-diagonal entries with zeros along the diagonal. Therefore we have

$$\begin{aligned} &\epsilon I + L_{11} \\ &= D^{1/2}(I - D^{-1/2}ND^{-1/2})D^{1/2} \\ &= D^{1/2}(I - M)D^{1/2} \qquad \text{where } M = D^{-1/2}ND^{-1/2} \end{aligned}$$

Using Lemma 3.2 and the identity $(D^{1/2})^T = D^{1/2}$, we can see that $I - M = D^{1/2}(\epsilon I + L_{11})D^{1/2}$ is positive-definite. Thus $(I - M)$ has positive eigenvalues. If $\lambda$ is an eigenvalue of $M$, then $1 - \lambda$ is an eigenvalue of $I - M$, thereby implying that the eigenvalues of $M$ are less than 1. The eigenvalues of $M$ are also greater than $-1$ by the following reasoning:

$$\begin{aligned} \lambda_1(M) &= \min_{v \neq 0} \frac{v^T M v}{v^T v} \\ &\geq \min_{v \neq 0} \frac{-|v|^T M |v|}{v^T v} \\ &= -\max_{v \neq 0} \frac{|v|^T M |v|}{v^T v} \\ &\geq -\lambda_t(M) \qquad \lambda_t(M) \text{ is the largest eigenvalue of } M \\ &> -1 \end{aligned}$$

Hence $\rho(M) < 1$. Then it directly follows from Lemma 3.3 that $(I - M)^{-1} = \sum_{l=0}^{\infty} M^l$.

$$\begin{aligned} \therefore Y &= (\epsilon I + L_{11})^{-1} \\ &= D^{-1/2}(I - M)^{-1}D^{-1/2} \\ &= D^{-1/2} \cdot \sum_{l=0}^{\infty} M^l \cdot D^{-1/2} \end{aligned}$$

$M$ describes the adjacency matrix of a weighted graph. As $M_{ij} < 0 \iff N_{ij} < 0 \iff (L_{11})_{ij} > 0$ for all $i \neq j$ and $L_{11}$ describes the adjacency matrix of a connected weighted graph, it follows that $M$ also describes the adjacency matrix of a connected graph. Therefore a sufficiently high power of $M$ contains all positive entries, which implies that $Y$ is a positive matrix. ■

# 4 Examples

Below are the graphs as processed by the above algorithm, unless otherwise stated. The SageMath code was created as a separate PDF file and has been appended at the end. For a couple of examples:

1. In this graph, the above algorithm gives a cut different from what would be optimal.

5. This example shows a highly unbalanced cut.

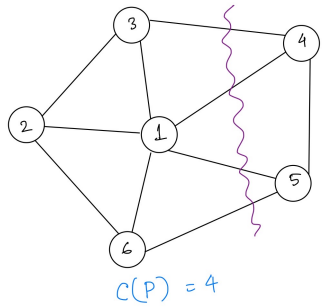



Figure 1: Example 1 as processed by the algorithm

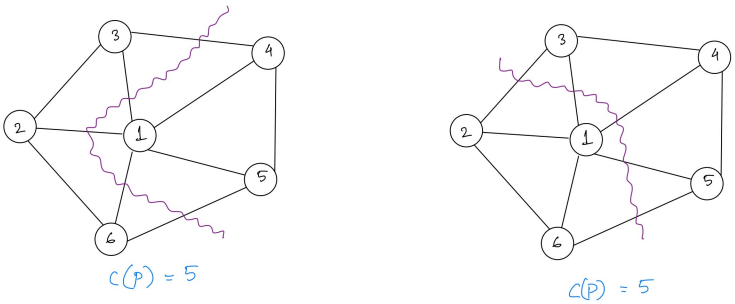



Figure 2: Ideal cuts for Example 1

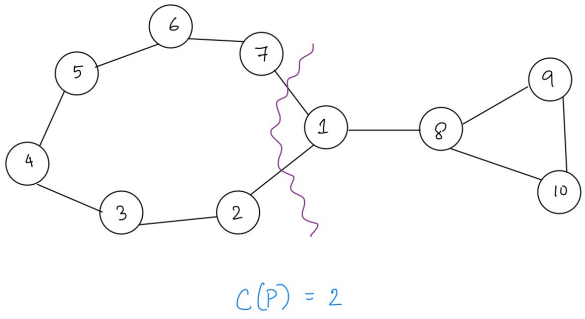



Figure 3: Example 2 as processed by the algorithm

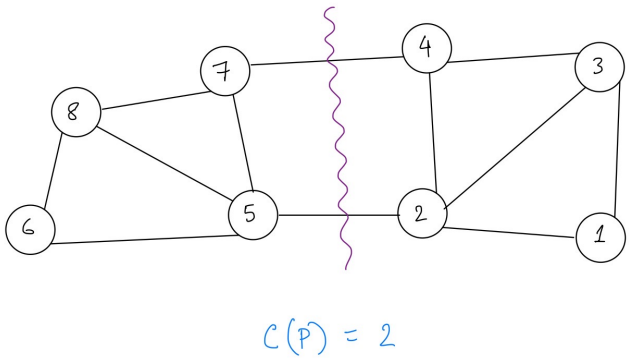



Figure 4: Example 3 as processed by the algorithm

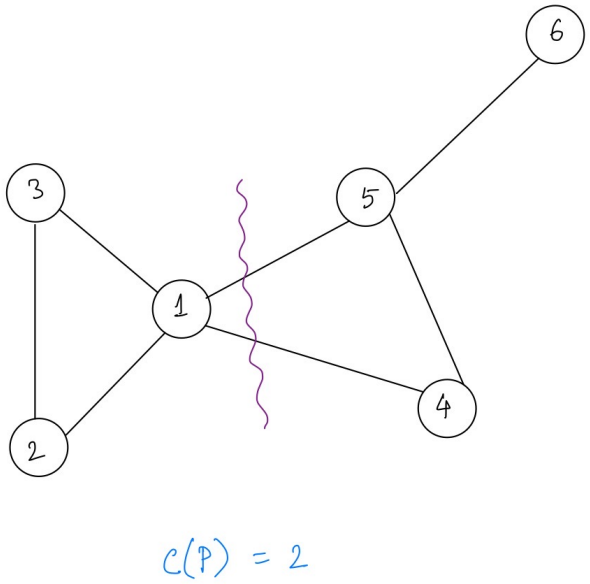



Figure 5: Example 4 as processed by the algorithm

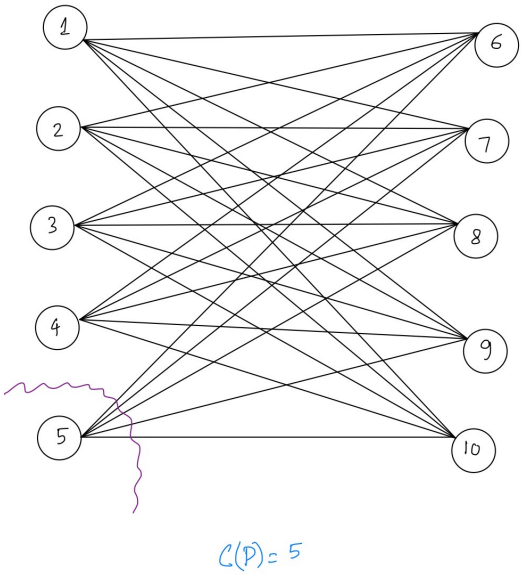



Figure 6: Example 5 as processed by the algorithm

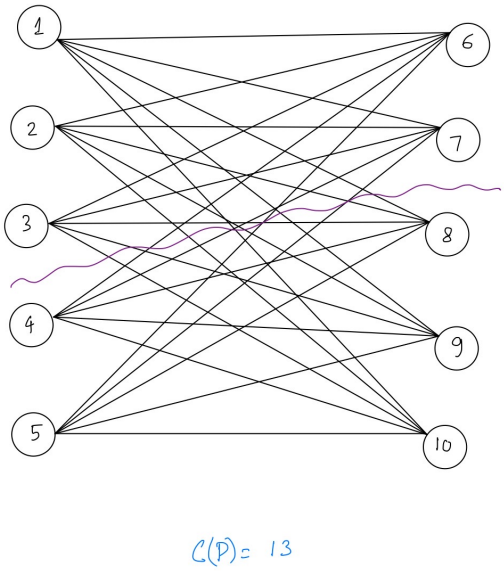



Figure 7: Some better balanced cuts for Example 5

# 5 Some natural questions

## 5.1 What effect does connectedness have on $\lambda_2$ (of $\mathcal{L}$)?

The following lemma answers this questions. It also justifies why $\lambda_2$ is called the *algebraic connectivity* of the graph.

**Lemma 5.1.** $\lambda_2 > 0 \iff$ *the graph is connected.*

*Proof.* ( $\implies$ ) If $G$ has two connected components, then $(1 \cdots, 1, 0 \cdots, 0)$ and $(0 \cdots, 0, 1 \cdots, 1)$ are linearly independent eigenvectors for 0. This means $(\lambda_1 =)\lambda_2 = 0$.
( $\impliedby$ ) Suppose $G$ is connected. If $\boldsymbol{v}$ is an eigenvector corresponding to 0 then $0 = \boldsymbol{v}^t \mathcal{L} \boldsymbol{v} = \sum_{\{i,j\} \in E} (v_i - v_j)^2 \implies v_i = v_j$ for every edge $\{i, j\}$. Due to connectedness, $\boldsymbol{v} \in \langle (1, \cdots, 1) \rangle$. So, geometric (= algebraic) multiplicity of $(1, \cdots, 1)$, which means that $\lambda_2 \neq 0$. ■

## 5.2 What if all entries in the Fiedler vector are positive?

Say $\mathcal{L}\boldsymbol{v} = \lambda \boldsymbol{v}$ with $\lambda > 0$.

$$\begin{aligned} l_{11}v_1 + \cdots + l_{1n}v_n &= \lambda v_1 \\ &\vdots \\ l_{n1}v_1 + \cdots + l_{nn}v_n &= \lambda v_n \end{aligned}$$

Adding these gives $\sum_{i=1}^{n} l_{i1}v_1 + \cdots + \sum_{i=1}^{n} l_{in}v_n = \lambda \sum_{i=1}^{n} v_i$. All the red sums are 0 because of the way $\mathcal{L}$ is defined. Indeed, the $i^{\text{th}}$ red sum has $+\deg_i$ contribution from $i$ and $-1$ contribution from each neighbour of $i$, which makes the total 0. It follows that $\sum_i v_i = 0$ because $\lambda > 0$.

## 5.3 How to balance the 0's (if at all) in the Fiedler vector?

As we saw in the examples, there can be cases where the presence of 0's causes problems in balancing the sizes of $V_{1,2}$. What do we do if there are only a few nonzero components of the Fiedler vector and a bunch of 0's?
We do not have a solution to this yet, and this is indeed a drawback of the Fiedler vector method because it gives only an approximate solution. By observation, we had conjectured that

> For a connected graph $G$ with Laplacian $\mathcal{L}$, let $\boldsymbol{v}$ be a Fiedler vector.
> Look at $S = \{ i : v_i = 0 \}$. Then for each $i \in S, \exists j, k$ such that
> $v_j > 0, v_k < 0$ and both $j$ and $k$ are neighbours of $i$.

However, upon exploring further examples, we discovered that the conjecture is false. A counterexample is $K_{n,n}$, the complete bipartite graph on $2n$ vertices.

# References

[1] Miroslav Fiedler. “A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory”. eng. In: *Czechoslovak Mathematical Journal* 25.4 (1975), pp. 619–633. URL: http://eudml.org/doc/12900.

[2] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Matrix Analysis. Cambridge University Press, 2013.

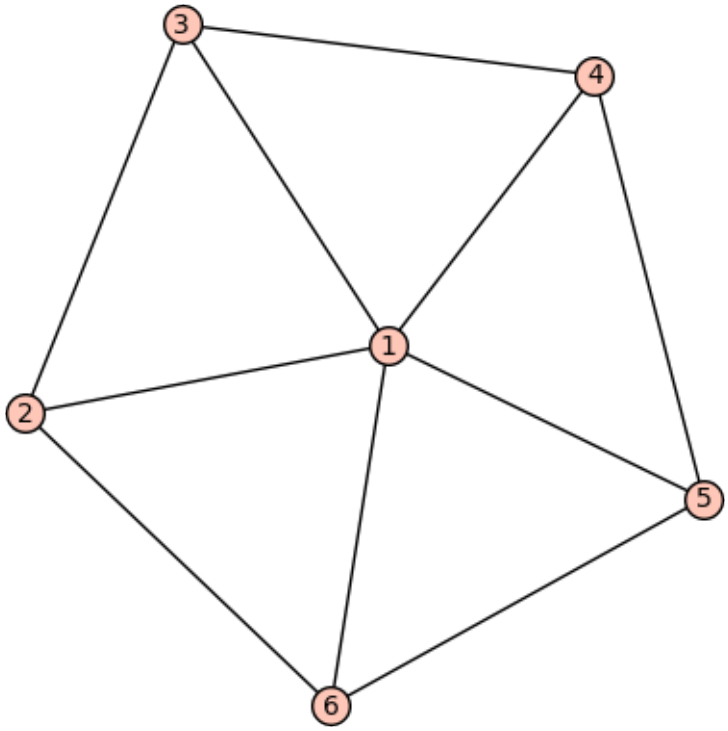
[3] Brian Slininger. *Fiedler's Theory of Spectral Graph Partitioning*.

[4] Justin Wyss-Gallifent. *Graph theory*. https://www.math.umd.edu/~immortal/MATH401/book/ch_graph_theory.pdf.

# Examples for the Fiedler-vector method on SageMath

May 24, 2022

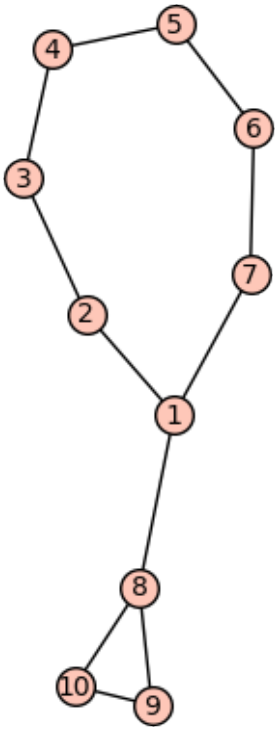
[1]:
```
e = [(1,2),(2,3), (3,4), (4,5), (5,6), (6,2), (1,3), (1,4), (1,5), (1,6)]
H = Graph()
H.add_edges(e)
L = H.laplacian_matrix()
u = L.eigenvalues()
u.sort()
n = L.nrows()
I = matrix.identity(n)
v = (L-u[1]*I).kernel().basis()
H.show()
fv = []
for i in v[0]:
    fv.append(round(i, ndigits=7))
print("Eigenvalues:\n ",u, "\n")
print("Fiedler vector: \n",fv)
pos = []
neg = []
ze = []
for i in range(n):
    #print(fv[i])
    if (fv[i]>0):
        pos.append(i+1)
    elif (fv[i]<0):
        neg.append(i+1)
    else:
        ze.append(i+1)
print("+: ",pos,"\n-: ",neg,"\n0: ",ze)
```




```
Eigenvalues:
  [0, 2.3819660112501106?, 2.3819660112501106?, 4.6180339887498895?,
4.6180339887498895?, 6]

Fiedler vector:
 [0.0, 1.0, -0.0, -1.0, -0.618034, 0.618034]
+:  [2, 6]
-:  [4, 5]
0:  [1, 3]
```

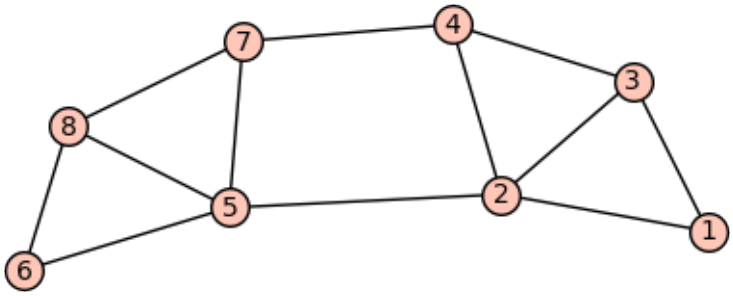
[2]:
```
e = [(1,8), (1,7), (9,10), (8,9), (8,10), (7,6), (6,5), (5,4), (4,3), (3,2),␣
 ↪(2,1)]
H = Graph()
H.add_edges(e)
L = H.laplacian_matrix()
u = L.eigenvalues()
u.sort()
n = L.nrows()
I = matrix.identity(n)
v = (L-u[1]*I).kernel().basis()
H.show()
fv = []
for i in v[0]:
    fv.append(round(i, ndigits=7))
print("Eigenvalues:\n ",u, "\n")
print("Fiedler vector: \n",fv)
pos = []
neg = []
ze = []
for i in range(n):
    #print(fv[i])
    if (fv[i]>0):
        pos.append(i+1)
    elif (fv[i]<0):
        neg.append(i+1)
    else:
        ze.append(i+1)
print("+: ",pos,"\n-: ",neg,"\n0: ",ze)
```




```
Eigenvalues:
  [0, 0.2374872911701323?, 0.7530203962825330?, 1, 2.445041867912629?,
2.563397064749295?, 3, 3.483178635798579?, 3.801937735804839?,
4.715937008281994?]

Fiedler vector:
 [1.0, -2.2002617, -4.8779892, -6.3972563, -6.3972563, -4.8779892, -2.2002617,
7.1630361, 9.3939892, 9.3939892]
+:  [1, 8, 9, 10]
-:  [2, 3, 4, 5, 6, 7]
0:  []
```

[3]:
```
e = [(1,2), (1,3), (3,2), (3,4), (2,4), (2,5), (5,7), (5,6), (5,8), (7,8),␣
 ↪(8,6), (8,7), (4,7)]
H = Graph()
H.add_edges(e)
L = H.laplacian_matrix()
u = L.eigenvalues()
u.sort()
n = L.nrows()
I = matrix.identity(n)
v = (L-u[1]*I).kernel().basis()
H.show()
fv = []
for i in v[0]:
    fv.append(round(i, ndigits=7))
print("Eigenvalues:\n ",u, "\n")
print("Fiedler vector: \n",fv)
pos = []
neg = []
ze = []
for i in range(n):
    #print(fv[i])
    if (fv[i]>0):
        pos.append(i+1)
    elif (fv[i]<0):
        neg.append(i+1)
    else:
        ze.append(i+1)
print("+: ",pos,"\n-: ",neg,"\n0: ",ze)
```

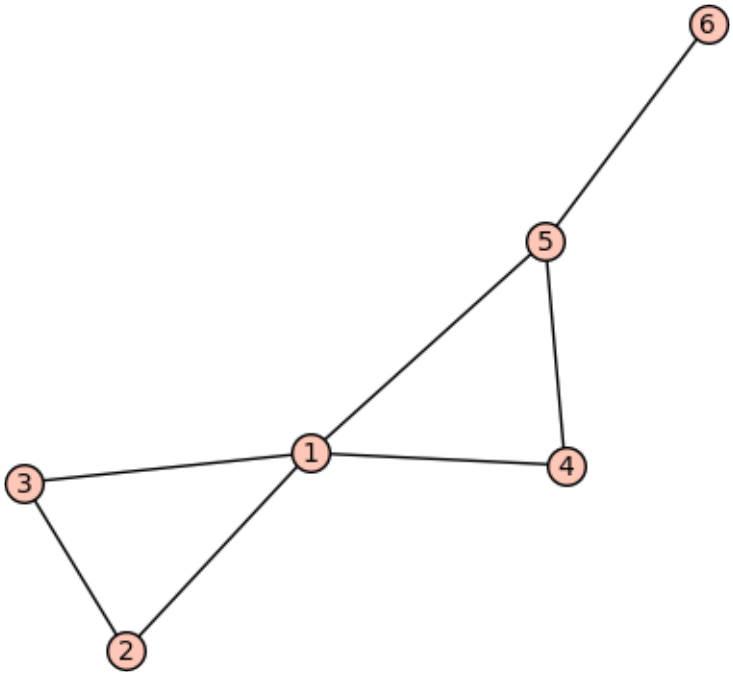



```
Eigenvalues:
  [0, 0.6677534988349950?, 2, 2.905212412256926?, 4, 4, 4.600195693596128?,
5.826838395311951?]

Fiedler vector:
 [1.0, 0.5123639, 0.8198826, 0.3998043, -0.5123639, -1.0, -0.3998043,
-0.8198826]
```

```
+:  [1, 2, 3, 4]
-:  [5, 6, 7, 8]
0:  []
```

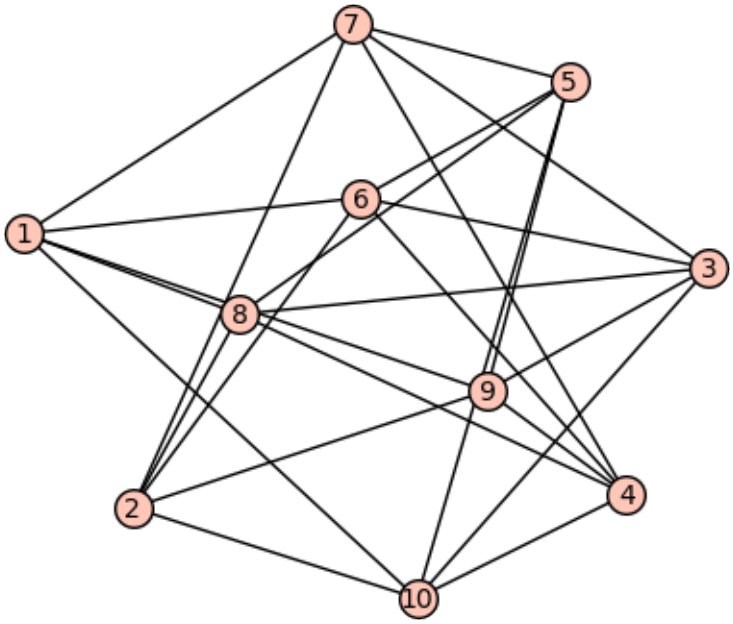
[4]:
```
e = [(1,2), (1,3), (2,3), (1,4), (1,5), (4,5), (6,5)]
H = Graph()
H.add_edges(e)
L = H.laplacian_matrix()
u = L.eigenvalues()
u.sort()
n = L.nrows()
I = matrix.identity(n)
v = (L-u[1]*I).kernel().basis()
H.show()
fv = []
for i in v[0]:
    fv.append(round(i, ndigits=7))
print("Eigenvalues:\n ",u, "\n")
print("Fiedler vector: \n",fv)
pos = []
neg = []
ze = []
for i in range(n):
    #print(fv[i])
    if (fv[i]>0):
        pos.append(i+1)
    elif (fv[i]<0):
        neg.append(i+1)
    else:
        ze.append(i+1)
print("+: ",pos,"\n-: ",neg,"\n0: ",ze)
```



```
Eigenvalues:
  [0, 0.6313506281127615?, 1.473842397735243?, 3, 3.787710597426518?,
5.107096376725478?]

Fiedler vector:
 [1.0, 2.7126046, 2.7126046, -0.4460601, -1.6104998, -4.3686494]
+:  [1, 2, 3]
-:  [4, 5, 6]
0:  []
```

[5]:
```
e = [(1,6), (1,7), (1,8), (1,9), (1,10), (2,6), (2,7), (2,8), (2,9), (2,10),␣
 ↪(3,6), (3,7), (3,8), (3,9), (3,10), (4,6), (4,7), (4,8), (4,9), (4,10), (5,6),␣
 ↪(5,7), (5,8), (5,9), (5,10)]
H = Graph()
H.add_edges(e)
L = H.laplacian_matrix()
u = L.eigenvalues()
u.sort()
n = L.nrows()
I = matrix.identity(n)
v = (L-u[1]*I).kernel().basis()
H.show()
fv = []
for i in v[0]:
    fv.append(round(i, ndigits=7))
print("Eigenvalues:\n ",u, "\n")
print("Fiedler vector: \n",fv)
pos = []
neg = []
ze = []
for i in range(n):
    #print(fv[i])
    if (fv[i]>0):
        pos.append(i+1)
    elif (fv[i]<0):
        neg.append(i+1)
    else:
        ze.append(i+1)
print("+: ",pos,"\n-: ",neg,"\n0: ",ze)
```



```
Eigenvalues:
  [0, 5, 5, 5, 5, 5, 5, 5, 5, 10]

Fiedler vector:
 [1.0, 0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
+:  [1]
-:  [5]
0:  [2, 3, 4, 6, 7, 8, 9, 10]
```