Università della Svizzera italiana

Institute of Computing
CI

Introduction to Graph Partitioning

Bachelor Course – Numerical Computing

Pietro Miotti,
PhD Student,
Scientific Computing Group, Institute of Computing

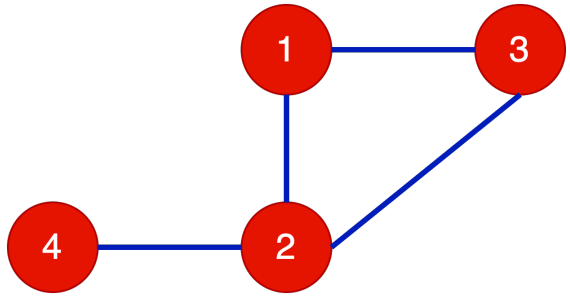
Quick Recap

- What is a Graph?
- Why Graphs?
- Understanding Graphs: analysis and algorithms

What is a Graph?

Mathematical rep. Graphical representation Matrix rep.

$G = (V, E)$



$$A_G = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

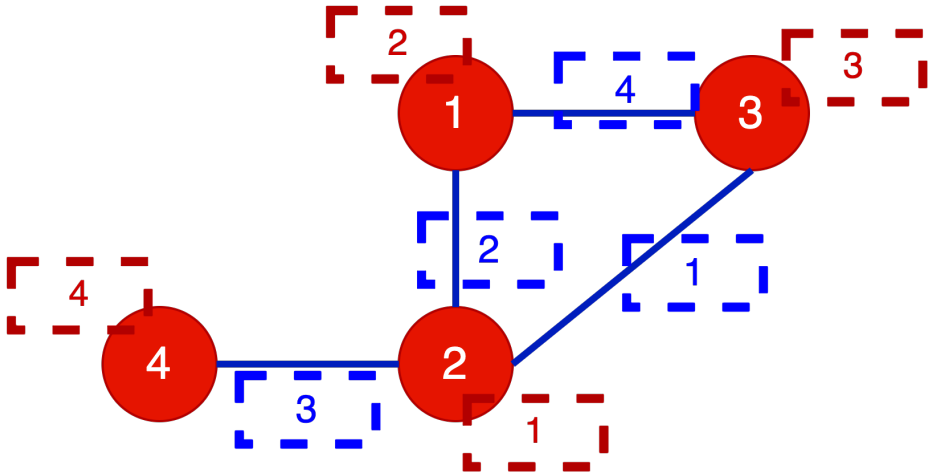
$$D_G = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$V = \{1, 2, 3, 4\} = \{ v_i \mid i=1,\ldots,n\}$

$E = \{ (1,2), (1,3), (2,3), (2,4)\} = \{ e_{ij} \mid v_i \text{ and } v_j \text{ are connected}\}$

Weighted graph

$G = (V, E, W_V, W_E)$



$V = \{1, 2, 3, 4\} = \{ v_i \mid i=1,\ldots,n\}$

$E = \{ (1,2), (1,3), (2,3), (2,4)\} = \{ e_{ij} \mid v_i \text{ and } v_j \text{ are connected}\}$

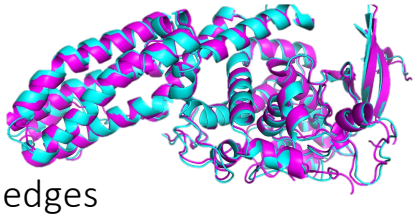
$W_v = \{2, 1, 3, 4\} = \{ w_v(v_i) \mid v_i \epsilon V\}$ (weight of vertices).

$W_E = \{2, 4, 1, 3\} = \{ w_e(e_{ij}) \mid e_{ij} \epsilon E\}$ (weight of edges).

Why Graphs?

Graphs are very flexible **data structures** used for modeling in many different fields, whenever there are **entities** and **relations** between them

➢ Bioinformatics

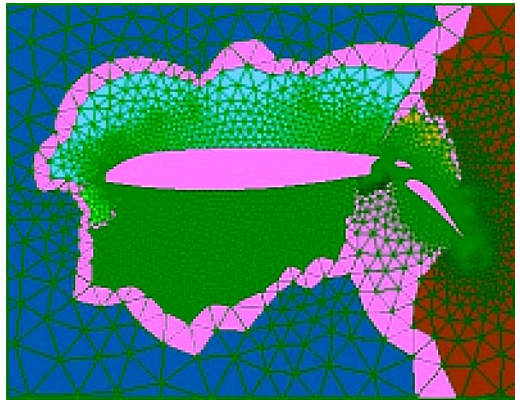


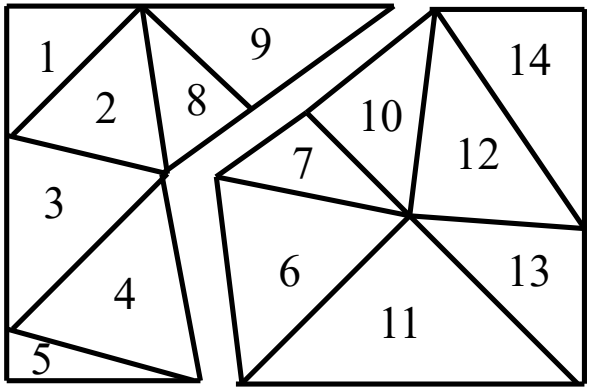
"A folded protein can be thought of as a "spatial graph", where residues are the nodes and edges connect the residues in close proximity" source: https://www.deepmind.com/blog/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology

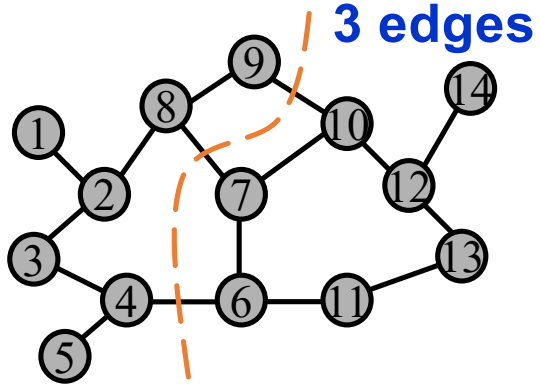
➢ Social Science: Knowledge Graph based recommender systems

Source: https://towardsdatascience.com/introduction-to-knowledge-graph-based-recommender-systems-34254efd1960

➢ HPC in Physics: parallel finite element method for solving Partial Differential Equations







Understanding Graphs: analysis and algorithms

- How to find the most important node in a Graph?

  PageRank!

- How to divide the Graph in two or more parts that are balanced?

  Graph partitioning

- How to find possible clusters in the graph?

  Graph clustering

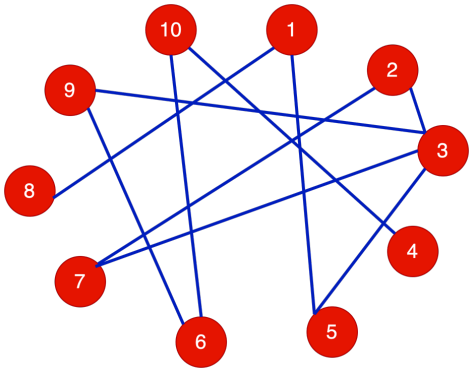
Graph partitioning (bisection): problem setup

You and your friends want to organize a soccer match (whichever team game). There are a total of 10 people, and you need to form **two teams**.

Here is the rule: each individual can specify **one preferred teammate** for the match. However, since you are all friends, **you are open to playing with anyone who selects you** as their preference. If your preference is not taken into account when assembling the teams, you feel disappointed.

How can you create the two teams in a way that maximizes everyone's happiness?

Modeling the problem as a Graph G(V,E)



Each player is a node
Each preference is an undirected edge

E = preferences
V = players

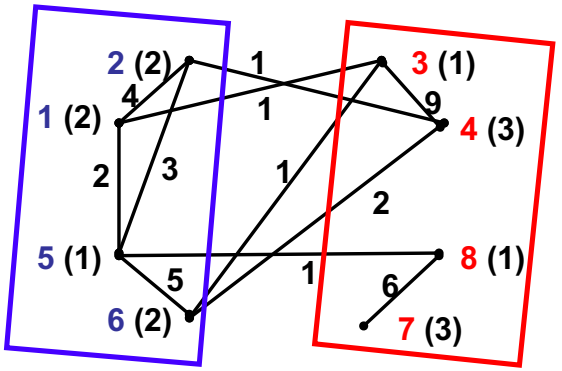
Goal:
Create 2 teams V_1 and V_2 such that:

- $|V_1| = |V_2|$: each team must have the same number of players
- $V = V_1 \cup V_2$: everyone must play
- $V_1 \cap V_2 = \emptyset$: one player cannot play for both teams
- For maximizing the happiness the sum of the edges connecting pairs in V_1 and V_2 is minimized

$$\min |\{e_{IJ} \in E \mid v_i \in V_1 \text{ und } v_j \in V_2\}|$$

That's it, this is the graph bisection problem!

More general mathematical formulation



- Given a graph $G = (V, E, W_V, W_E)$
  - V = nodes (or vertices)
  - E = edges

- Choose a partition $V = V_1 \cup V_2$ such that:

  The sum of the node in each V_j is "about the same"

  $$V = V_1 \cup V_2 \;,\; V_1 \cap V_2 = \emptyset, \quad |V_1| = |V_2|$$

  The sum of edge connecting pairs V_1 and V_2 is minimized

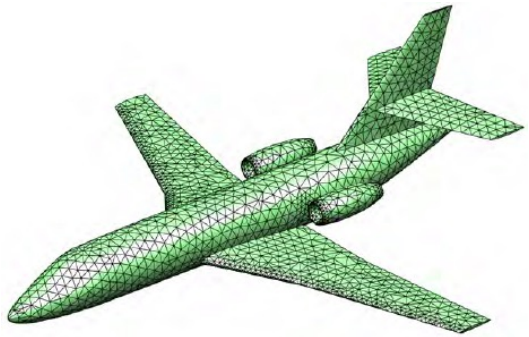
  $$\min |\{e_{IJ} \in E \mid v_i \in V_1 \text{ und } v_j \in V_2\}|$$

How to solve the partitoning problem?

Two families of algorithms

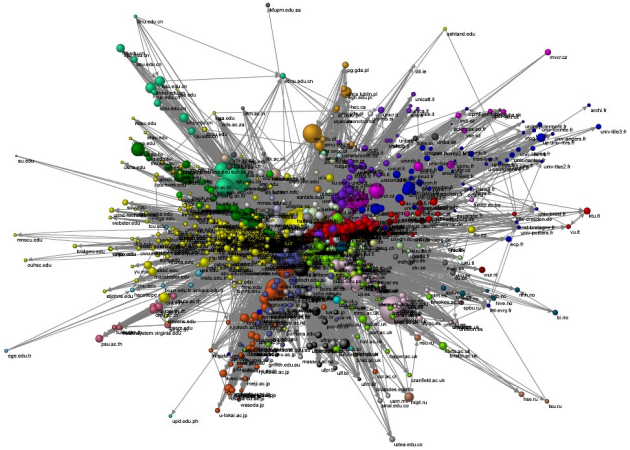
Partitioning **with** nodal coordinates each node has x,y,z coordinates (partition space):

- Coordinate Bisection
- Recursive Coordinate Bisection
- Inertial Partitioning



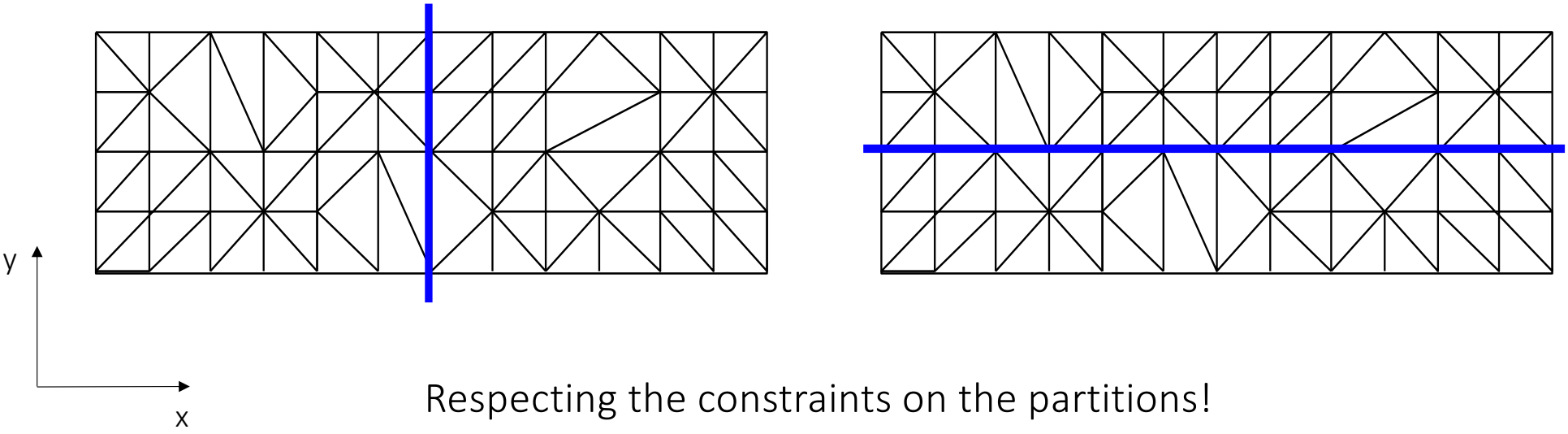
Partitioning **without** Nodal Coordinates

- Spectral Methods



Coordinate Bisection

Partition the graph along hyperplanes orthogonal to the coordinate system



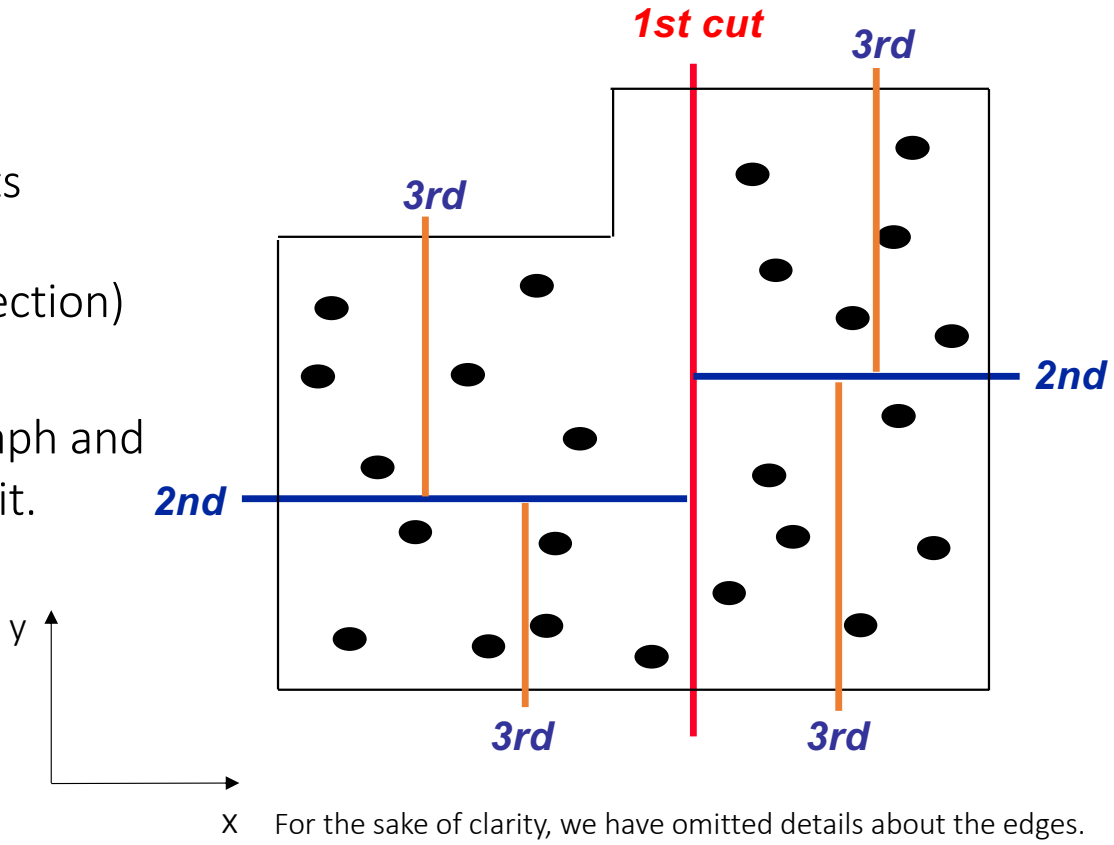
Respecting the constraints on the partitions!

Recursive Coordinate Bisection (RBC)

Developed by Berger & Bokhari (1987)

Idea:

- Divide the graph into two equal parts using a cutting plane orthogonal to a coordinate axis (Coordinate Bisection)
- Consider each partition as a new graph and reapply the Coordinate Bisection to it. (Recursive call)

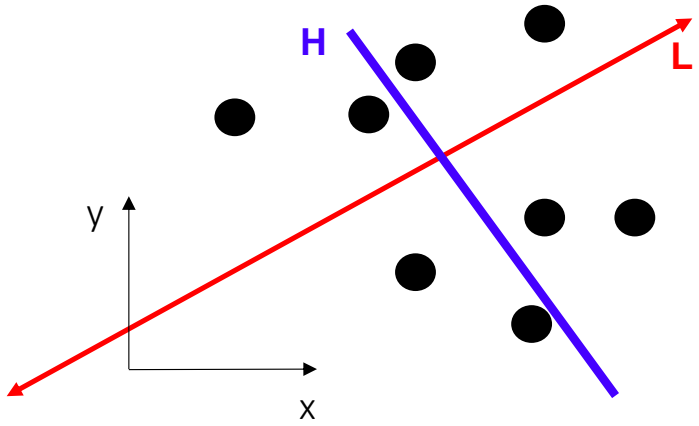


For the sake of clarity, we have omitted details about the edges.

Inertial Partitioning

Generalization of the coordinate partitioning:

1) Select a line **L**, not necessarly orthogonal to the axes
2) Select the line **H** orthogonal to it



How to select the line?

For the sake of clarity, we have omitted details about the edges.

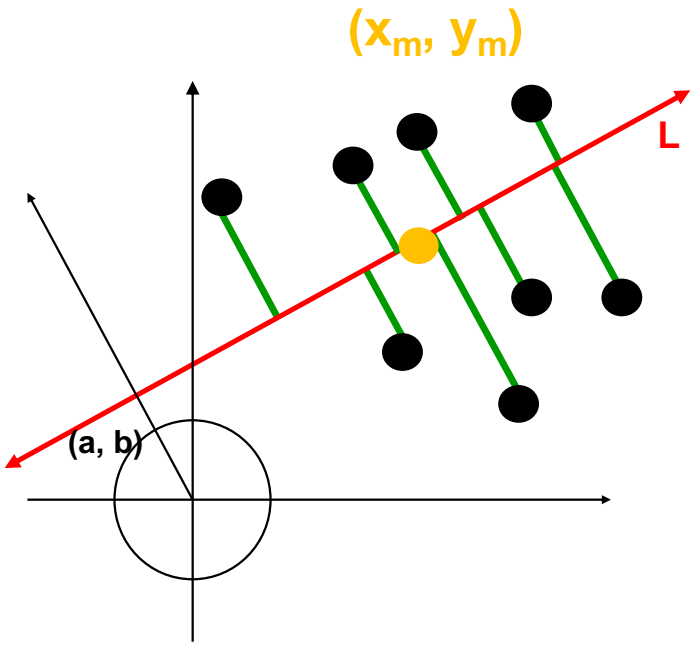
How to select L ?

1) $L : ax + by + c = 0 \qquad a^2 + b^2 = 1$

2) The line L must intersect the center of mass (x_m, y_m):

$$\begin{cases} ax + by + c = 0 \\ ax_m + by_m + c = 0 \end{cases}$$

$$a(x - x_m) + b(y - y_m) = 0$$

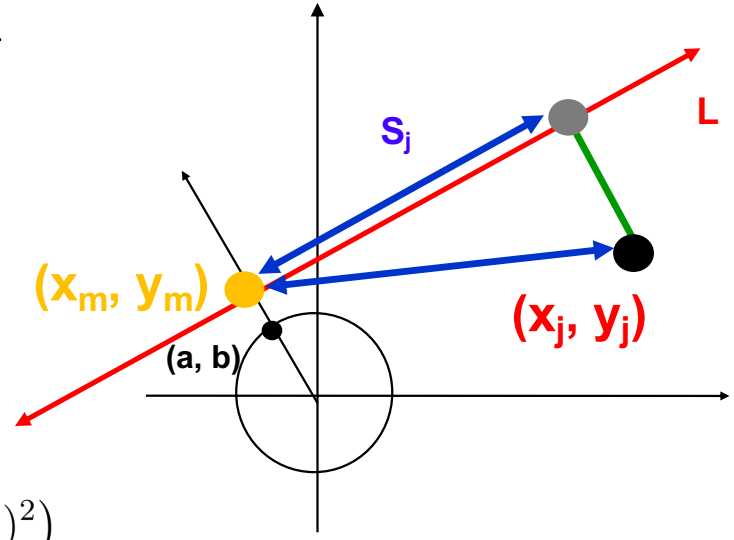


3) *a* and *b* need to be taken to minimize the squared distances of the nodes from the line L

$$\sum_{j=1}^{n} d_j^2 = \sum_{j=1}^{n} (\text{j-th green line})^2$$

4) Compute the projections of the node j on the line L

$$S_j = a(y_j - y_m) - b(x_j - x_m)$$



5) Compute the the green line using **Pythagoras**

$$= \sum_{j=1}^{n} (j - thgreenline)^2$$

$$= \sum_{j=1}^{n} \left((x_j - x_m)^2 + (y_j - y_m)^2 - (-b(x_j - x_m) + a(y_j - y_m))^2 \right)$$

$$= (1 - b^2) \sum_{j}^{n} (x_j - x_m)^2 + 2ab \sum_{j}^{n} (x_j - x_m)(y_j - y_m) + (1 - a^2) \sum_{j}^{n} (y_j - y_m)^2$$

$$= a^2 \sum_{j}^{n} (x_j - x_m)^2 + 2ab \sum_{j}^{n} (x_j - x_m)(y_j - y_m) + b^2 \sum_{j}^{n} (y_j - y_m)^2$$

$$= a^2 \mathbf{X_1} + 2ab \mathbf{X_2} + b^2 \mathbf{X_3}$$

$$= \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} \mathbf{X_1} & \mathbf{X_2} \\ \mathbf{X_2} & \mathbf{X_3} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \mathbf{u}^T \mathbf{M} \mathbf{u}$$

Recall Rayleight quotient $\lambda = \dfrac{u^T A u}{u^T u}$

Therefore, for minimizing the $\mathbf{u}^T\mathbf{M}\mathbf{u}$ we can select u = [a, b] the eigenvector correpondent to the minimum eigenvalue.

6) Once we have the line L, we can compute the median:

$$S_k = median(S_1, \ldots, S_n)$$

And use the median to partition the nodes

Let nodes with $S_j < S_k$ be in V_1, the rest in V_2

Inertial Algorithm:

- Assemble matrix M, find eigenvector u_1 and compute the line L
- Partition the nodes along the line H which goes through the center of mass and is orthogonal to L

Nodal Coodinates– Summary

- Algorithms using nodal coordinates are efficient
- Rely on graphs having nodes connected (mostly) to “nearest neighbors” in space
  - algorithm does **not depend on where actual edges** are!
- Common when graph arises from physical model
- **Ignores edges**, but can be used as good starting guess for subsequent partitioners that do examine edges
- Can do very poorly if graph connection is not spatial

Spectral partitioning

Based on theory of Fiedler (1970s), popularized by Horst Simon (1995)

Spectral because it exploits the spectrum of the Graph.

Spectrum: the eigenvectors v_i of a graph, ordered by the magnitude of their corresponding eigenvalues λ_i. $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ where $\lambda_1 \leq \lambda_2, \ldots \leq \lambda_n$.

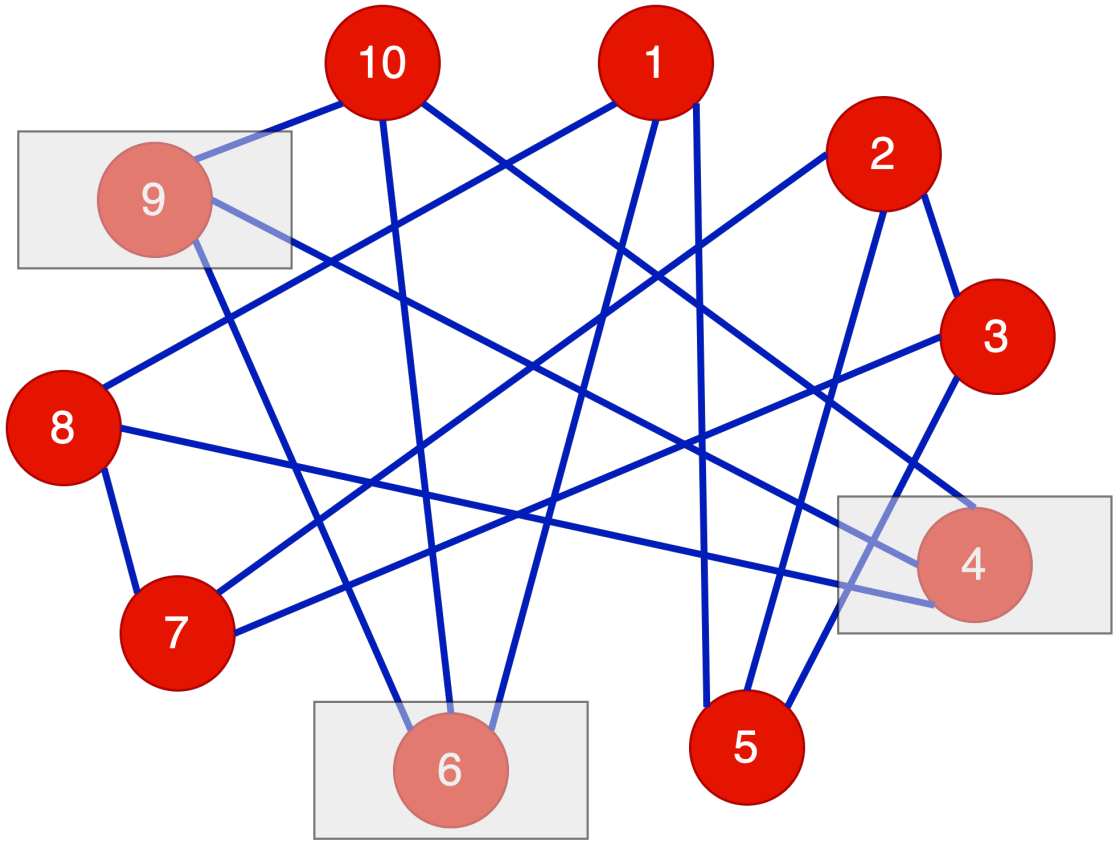
Intuition: Ax where A is the adjancency matrix

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

$$y_i = \sum_j A_{i,j} x_j = \sum_{\text{edges}(i,j)} x_j$$

*The **i-th component** of the dot product of the Adjacency Matrix with a **label vector x** is the **sum of the labels of the nodes that are connected with i***

For simplicity, assume the label vector X is discrete (0 and 1) and consider it as a mask for the nodes



1
1
1
0
1
0
1
1
0
1

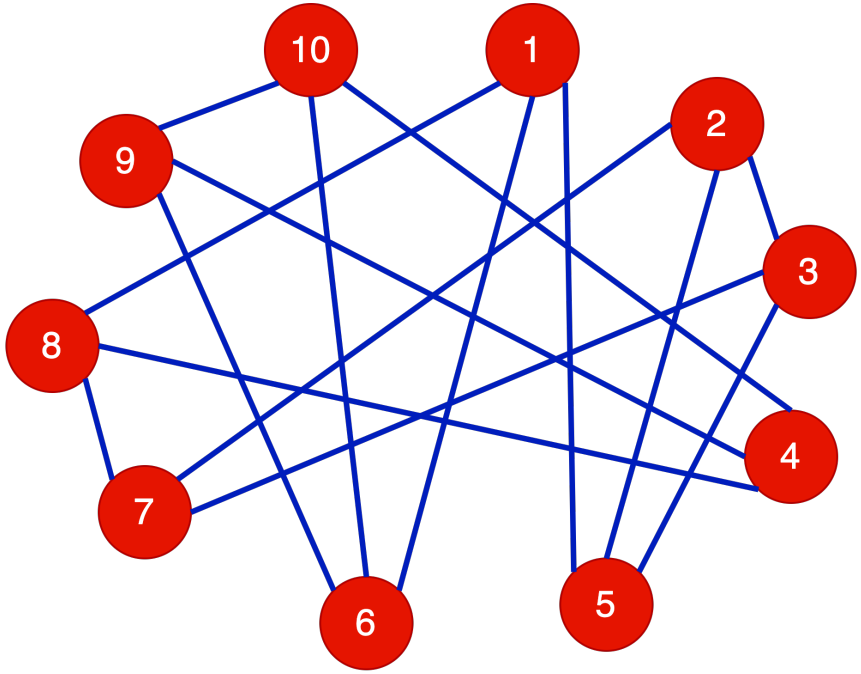
Eigenvalue problem Ax = λx (1)

Take **x** as the vector with all ones and assume every node has a **degree d**

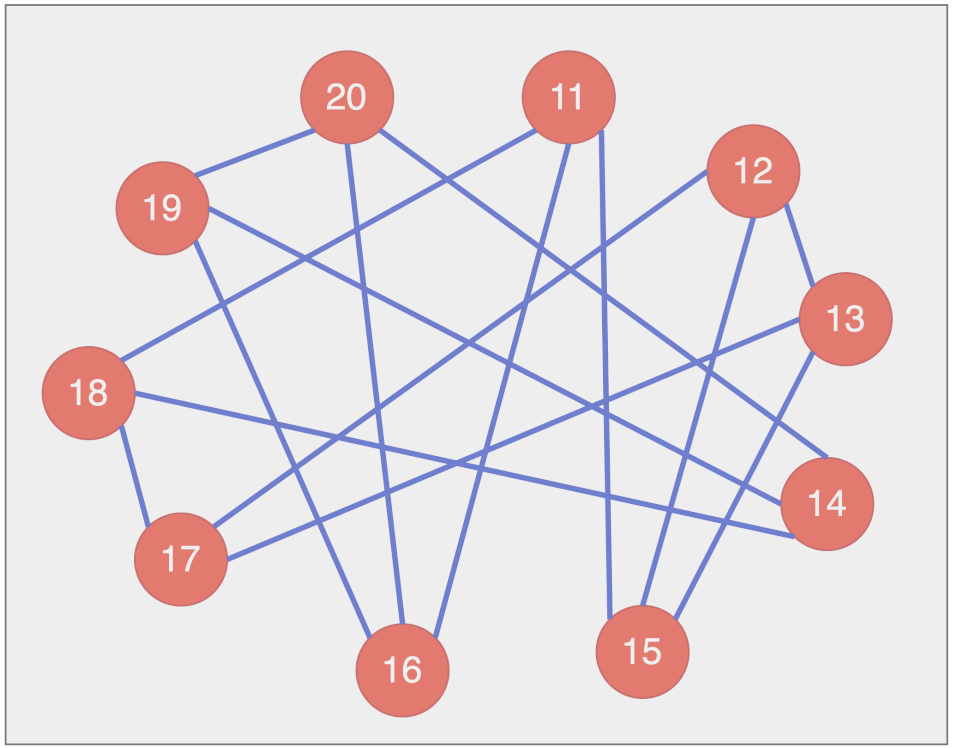
$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} d \\ \vdots \\ d \end{pmatrix} = \lambda \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

We have found the eigenvector x = (1,1,1,1..,1) and the corresponding eigenvalue λ = d

Assume we have a Graph with two disconnected components (E,F)
and each component has nodes with degree d



E



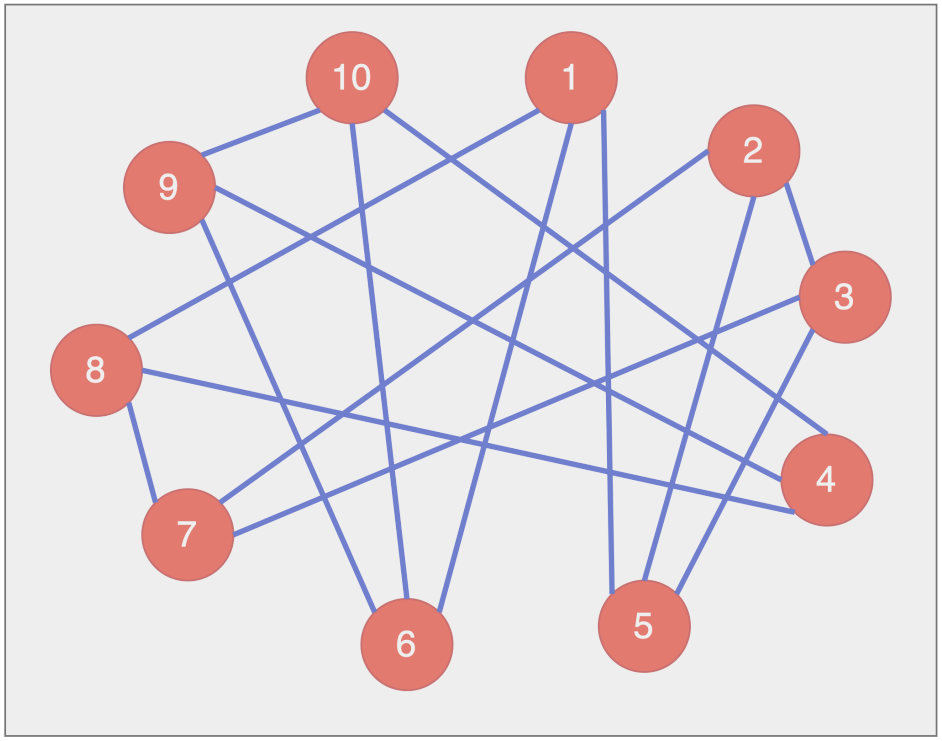
F

E

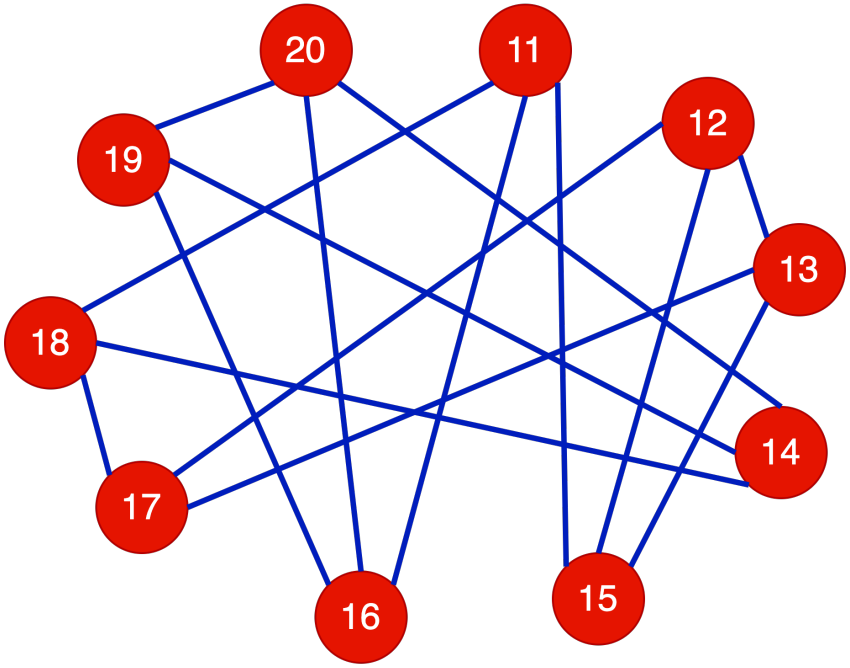
1
1
1
1
1
1
1
1
1
1

F

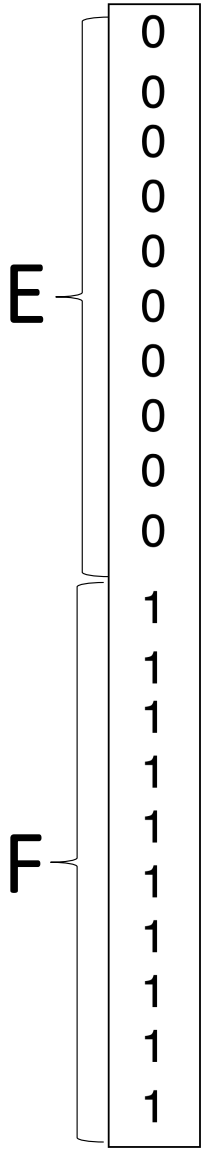
0
0
0
0
0
0
0
0
0
0


10
1
9
2
3
8
4
7
6
5

E


20
11
19
12
13
18
14
17
16
15

F


E
F
0
0
0
0
0
0
0
0
0
0
1
1
1
1
1
1
1
1
1
1
1

Eigenvalue problem Ax = λx (2)

Assume we have a Graph with two disconnected components (E,F) and each component has nodes with degree d

Take ***x*** as the vector with all ones on E and zeros on F or viceversa:

- x' = (1,1,…1, 0, 0, …, 0)
- x" = (0,0,..,0, 1, 1,…, 1)

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix} = \begin{pmatrix} d \\ \vdots \\ 0 \end{pmatrix} = \lambda \begin{pmatrix} 1 \\ \vdots \\ 0 \end{pmatrix}$$

We have found the eigenvectors x' = (1,1,…1,0…,0), x''= (0,0,…0,1…,1), and the corresponding eigenvalue which is λ = d for both x' and x''. λ has multiplicity 2!

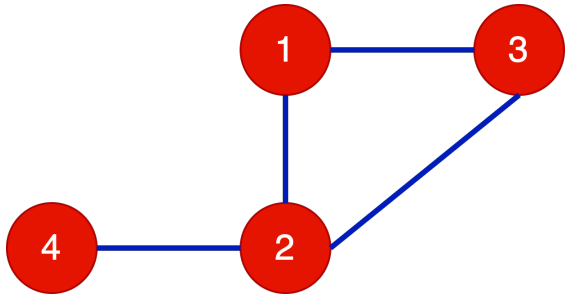
We need a matrix **L(G)** that fully represent the graph

$$L(G) = D_G - A_G$$

- Definition: The Laplacian matrix L(G) of a graph G(V, E) is a |V| by |V| symmetric matrix, with one row and column for each node. It is defined by
  - L(G) (i,i) = degree of node i (number of incident edges)
  - L(G) (i,j) = -1 if i != j and there is an edge (i,j)
  - L(G) (i,j) = 0 otherwise

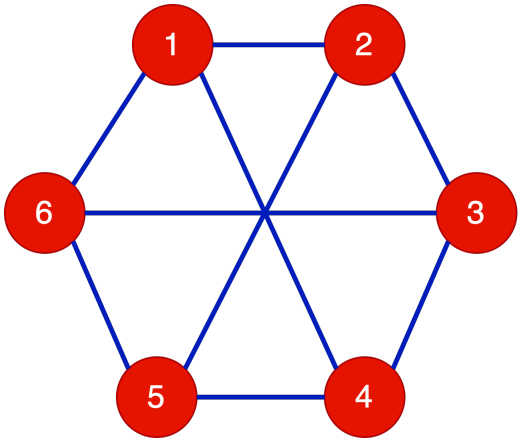
Property: The sum of every row and column is equal to 0!

Which means that if we take the vector x = (1,1,…1), then Lx = 0 → $\lambda = \lambda_1 = 0$



$$A_G = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \qquad D_G = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$L_G = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 2 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}$$



$$A_G = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \quad D_G = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix}$$

$$L_G = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 3 & -1 & 0 & 1 & 0 & -1 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 3 & -1 & 0 & -1 \\ -1 & 0 & -1 & 3 & -1 & 0 \\ 0 & -1 & 0 & -1 & 3 & -1 \\ -1 & 0 & -1 & 0 & -1 & 0 \end{pmatrix}$$

Properties of the Laplacian

- **Theorem:** Given a graph G, L(G) has the following properties
  - L(G) is symmetric — this means the eigenvalues of L(G) are real and its eigenvectors are real and orthogonal.
  - Let e = $[1,\ldots,1]^T$, i.e. the column vector of all ones. Then L(G)*e=0
  - The eigenvalues of L(G) are **nonnegative**:

    $0 = \lambda_1 \leq \lambda_2, \ldots \leq \lambda_n$
  - The number of connected components of G is equal to the number of λ_i equal to 0.

- **Definition**: λ_2 (L(G)) is the algebraic connectivity of G
  - The magnitude of λ_2 measures connectivity
  - In particular, λ_2 != 0 if and only if G is connected

Recall Rayleight quotient, for symmetric matrices $\lambda = \frac{u^T A u}{u^T u}$

For the Laplacian Matrix, symmetric matrices $\lambda_1 = 0$, then $\lambda_2 = \min_x \frac{x^T L x}{x^T x}$

What is the meaning of $\min x^T L x$

$$x^T L x = \sum_{i,j}^{n} L_{i,j} x_i x_j = \sum_{i,j=1}^{n} (D_{i,j} - A_{i,j}) x_i x_j$$

$$= \sum D_{i,i} x_i^2 - \sum_{(i,j) \in E} 2 x_i x_j$$

$$= \sum_{(i,j) \in E} (x_i^2 + x_j^2 - 2 x_i x_j) = \sum_{(i,j) \in E} (x_i - x_j)^2$$

$$\lambda_2 = \min_x \frac{x^T L x}{x^T x}$$

- What else do we know about x_i?

1) $\sum_i x_i^2 = 1$ *NORMALIZED EIGENVECTOR (UNIT VECTOR)*

2) $\sum_i x_i = 0$ *ORTHOGONALITY W.R.T FIRST EIGENVECTOR*

$$\lambda_2 = \min_x x^T L x = \min_{\mathbf{x}} \sum_{(i,j)\in E} (x_i - x_j)^2$$

Minimum value for $x_i = x_j$ but violates (2)

How to find the optimal x ?

- **Theorem (Fiedler, 1975):**
  Let G be connected, L(G) the Laplace matrix, and N_+ and N_- a partitioning with

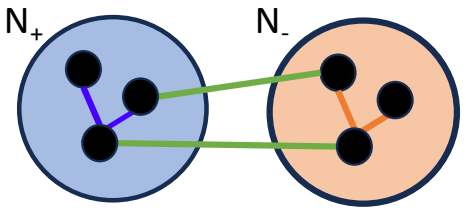
  x(i) = +1 if v_i in N_+
  x(i) = -1 if v_i in N_-.

  Then we have the following property:

$$\frac{x^T L x}{4} = \#\text{edge cutting between } N_+ \text{ and } N_-$$

  Proof: (next slide)

Relation between Laplace Matrix and Graph Partitioning



$$x^T L x = \sum_{(i,j)\in E} (x_i - x_j)^2$$

$$= \sum_{(i,j)\in E\wedge(i,j\in N_+)} (x_i - x_j)^2 + \sum_{(i,j)\in E\wedge(i,j\in N_-)} (x_i - x_j)^2 + \sum_{(i,j)\in E\wedge(i\in N_+,j\in N_-)\vee(j\in N_+,i\in N_-)} (x_i - x_j)^2$$

$$= \sum_{(i,j)\in E\wedge(i,j\in N_+)} (1-1)^2 + \sum_{(i,j)\in E\wedge(i,j\in N_-)} (-1+1)^2 + \sum_{(i,j)\in E\wedge(i\in N_+,j\in N_-)\vee(j\in N_+,i\in N_-)} (2)^2$$

$$\frac{x^T L x}{4} = \sum_{(i,j)\in E\wedge(i\in N_+,j\in N_-)\vee(j\in N_+,i\in N_-)} 1$$

$$\frac{x^T L x}{4} = \#\text{edge cutting between } N_+ \text{ and } N_-$$

Relation between Laplace Matrix and Graph Partitioning

- With the theorem we can formulate the graph bisection as a discrete optimization problem

1. $|V_1| = |V_2| \iff \sum x_i = 0$
2. min #cut edges between V_1 and V_2 $\iff \min x^T L x$

$$\min_x f(x) = \frac{x^T L x}{4} \qquad \text{Constraints} \qquad x_i \in \{+1, -1\}, \quad x^T x = n, \quad \sum x_i = 0$$

- The discrete combinatorial problem is NP-hard → use a continuous problem

$$min_z f(z) = \frac{z^T L z}{4} \qquad \text{Constraints} \qquad z_i \in \mathbb{R}, \quad z^T z = n, \quad \sum z_i = 0$$

$$min_z f(z) = \frac{z^T L z}{4}$$

- L(G) is symmetric → L(G) has n orthonormal eigenvectors

$$u_1, \ldots, u_n \text{ with eigenvalues } 0 = \lambda_1 \leq \ldots \leq \lambda_n$$
$$\text{and } u_1 = \sqrt{n}e \text{ where: } e = (1, 1, \ldots, 1)^T.$$

- A vector z is a linear combination of eigenvectors u_i:

$$z = \sum_i^n \alpha_i u_i$$

First constraint	Second constraint
$\sum_i z_i = z^T e = z^T u_1 = 0$	$z^T z = n$
$z^T u_1 = (\sum \alpha_i u_i)^T u_1 = \alpha_1 u_1^T u_1 = \alpha_1$ $\rightarrow \alpha_1 = 0$	$z^T z = (\sum \alpha_i u_i)^T (\sum \alpha_j u_j)$ $= \sum_i \sum_j \alpha_i \alpha_j u_i^T u_j = \sum \alpha_i^2 = n$

$$min_z f(z) = \frac{z^T L z}{4}$$

$$z^T L(G) z = (\sum \alpha_i u_i)^T L(\sum \alpha_j u_j) = (\sum \alpha_i u_i)^T (\sum \alpha_j \lambda_j u_j)$$
$$= \sum_i \sum_j \alpha_j \alpha_i \lambda_j u_i^T u_j = \sum_i \alpha_i^2 \lambda_i$$
$$= \sum_i \alpha_i^2 \lambda_i \geq \lambda_2 \sum \alpha_i^2 = \lambda_2 n$$

Minimum is at z = sqrt(n)*u_2.

Spectral Bisection Algorithm:

- Compute eigenvector u_2 corresponding to $\lambda_2(L(G))$
- For each vertex v of G
  - if $u_2(v) < 0$ put node v in partition V_1
  - else put vertex v in partition V_2

The second eigenvector u_2 is called Fiedler Eigenvector of the Graph Partitioning problem.

Summary:

- Quick recap & Motivation
- Partitioning **with Node Coordinates**:
  - Coordinate Bisection
  - Recursive Coordinate Bisection
  - Inertial Partitioning
- Partitioning **without Node Coordinates** :
  - Spectral Method

Resources:

- Deepmind, 2020 “AlphaFold: a solution to a 50-year-old grand challenge in biology”

  https://www.deepmind.com/blog/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology
- Notes: “Graph Partitioning: A survey” by Ulrich Elsner

  https://www.tu-chemnitz.de/sfb393/Files/PDF/sfb97-27.pdf
- Mining Massive Datasets, Jure Leskovec , Stanford University, lectures 31-34 https://www.youtube.com/playlist?list=PLLssT5z_DsK9JDLcT8T62VtzwyW9LNepV