Università
della
Svizzera
italiana

Institute of Computing CI

BSC INF Course - Graph-Partitioning

October 18, 2022

Olaf Schenk
Institute of Computing
Faculty of Informatics
USI Lugano

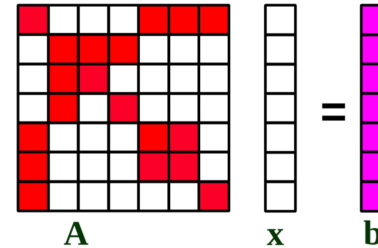
Content

- Motivation for graph partitioning
- Overview of heuristics

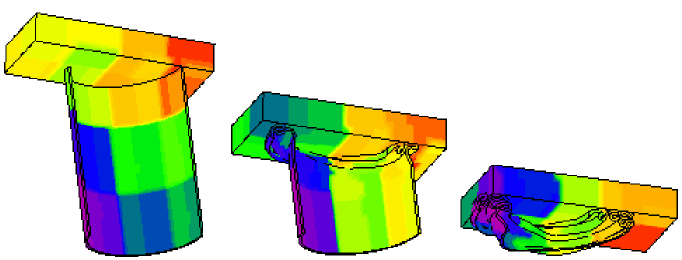
- Partitioning with nodal coordinates
  - Ex: In finite element models, node at point in (x, y, z) space
    Recursive Coordinate Bisection
    Inertial Partitioning
- Partitioning without nodal coordinates
  - Ex: In model of WWW, nodes are web pages
    Spectral Methods

Partitioning and Load Balancing

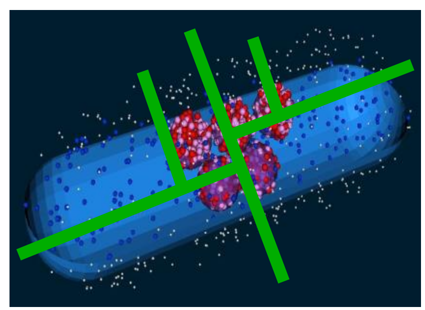
- **Goal:** assign data to processors to
  - minimize parallel application runtime
  - maximize utilization of computing resources
- **Metrics:**
  - minimize processor idle time (balance workloads)
  - keep inter-processor communication costs low
- Impacts performance of a wide range of simulations



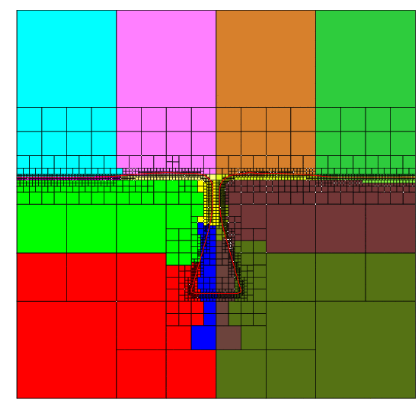
Linear solvers & preconditioners



Contact detection



Particle simulations



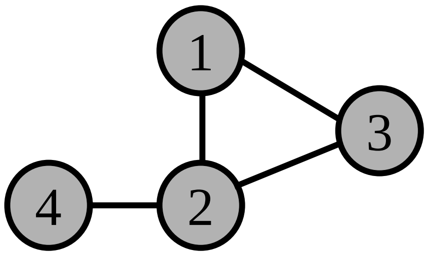
Adaptive mesh refinement

Graph Partitioning

- Work-horse of load-balancing community.
- **Highly successful model for PDE problems.**
- Model problem as a graph:
  - vertices = work associated with data (computation)
  - edges = relationships between data/computation (communication)
- Goal: Evenly distribute vertex weight while minimizing weight of cut edges.

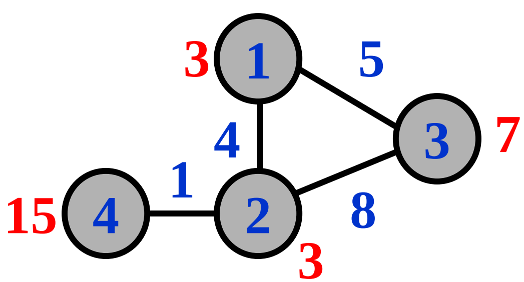
Definition of Graph

- Given a graph G = (V, E) with
  - Vertices $V = \{ v_i \mid i=1,\ldots,n\}$
  - Edges $E = \{ e_{ij} \mid v_i \text{ and } v_j \text{ are connected}\}$



$V = \{1, 2, 3, 4\}$
$E = \{ (1,2), (1,3), (2,3), (2,4)\}$

- A weighted graph $G = (V, E, W_v, W_e)$ has **node weights** and **edge weights**
  - $W_v = \{ w_v(v_i) \mid v_i \epsilon V\}$ („weight of vertices“).
  - $W_e = \{ w_e(e_{ij}) \mid e_{ij} \epsilon E\}$ („weight of edges“).

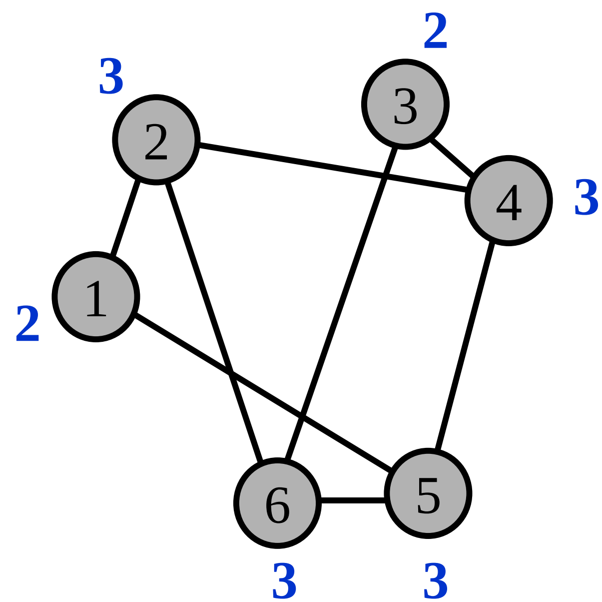


$W_v = \{ 3, 3, 7, 15\}$, $W_e = \{ 4, 5, 8, 1\}$

Examples for Graphs

- Symmetric sparse matrix and Graph G_A

$$A = \begin{array}{c|cccccc|} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 1 & 1 & & & 7 & \\ 2 & 8 & 6 & & 1 & & 1 \\ 3 & & & 3 & 1 & & 1 \\ 4 & & 5 & 1 & 2 & 1 & \\ 5 & 2 & & & 1 & 5 & 1 \\ 6 & & 1 & 5 & & 1 & 1 \\ \hline \end{array}$$



- $G_A = (V, E, W_V, W_E)$; $V = \{1, 2, 3, 4, 5, 6\}$,

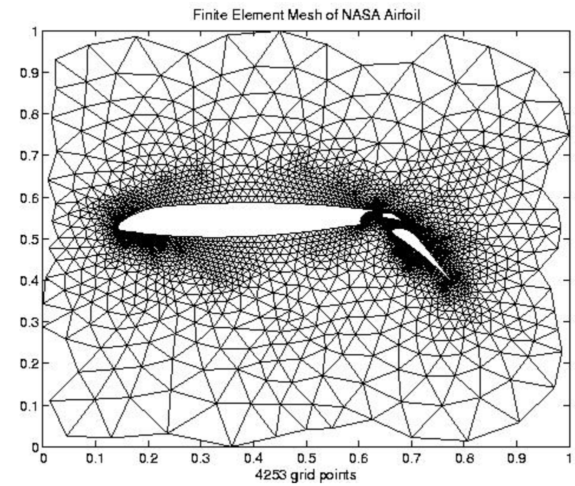
  $E = \{ (1,2), (1,5), (2,4), (2,6), (3,4), (3,6), (4,5), (5,6) \}$

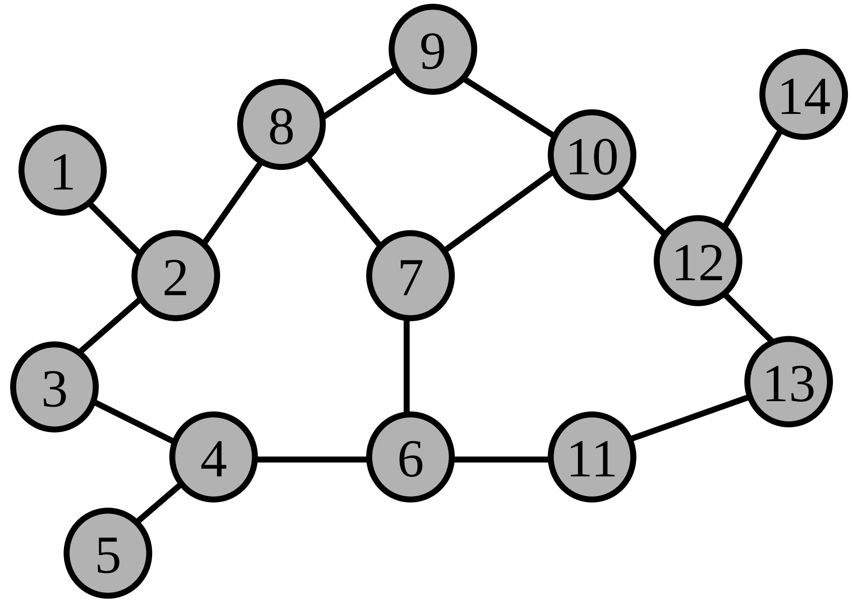
  $W_v = \{2, 3, 2, 3, 3, 3\}$ e.g. numbers of nonzeros in each row

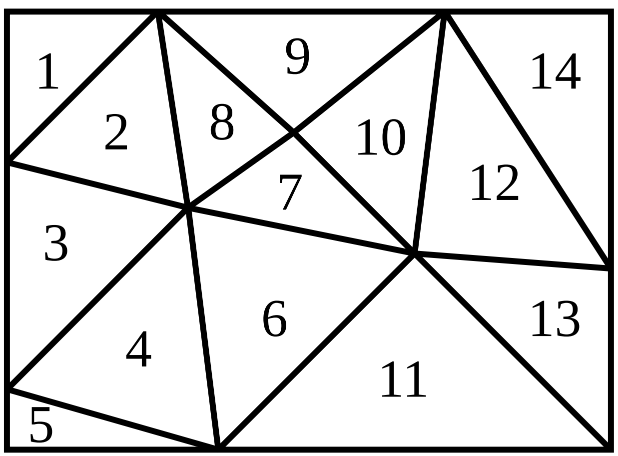
  $W_e = \{1, 1, 1, 1, 1, 1, 1, 1\}$

Examples for Graphs

- Finite-Element Simulations







Finite-Element Mesh

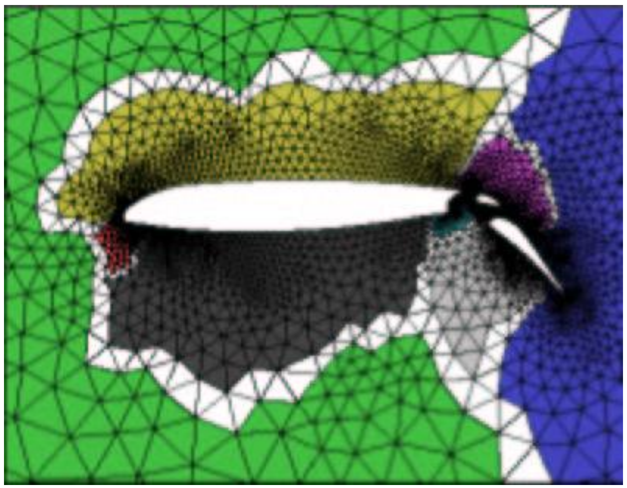
$$G_{FE} = (V, E),\ V = \{1, \ldots, 14\}$$

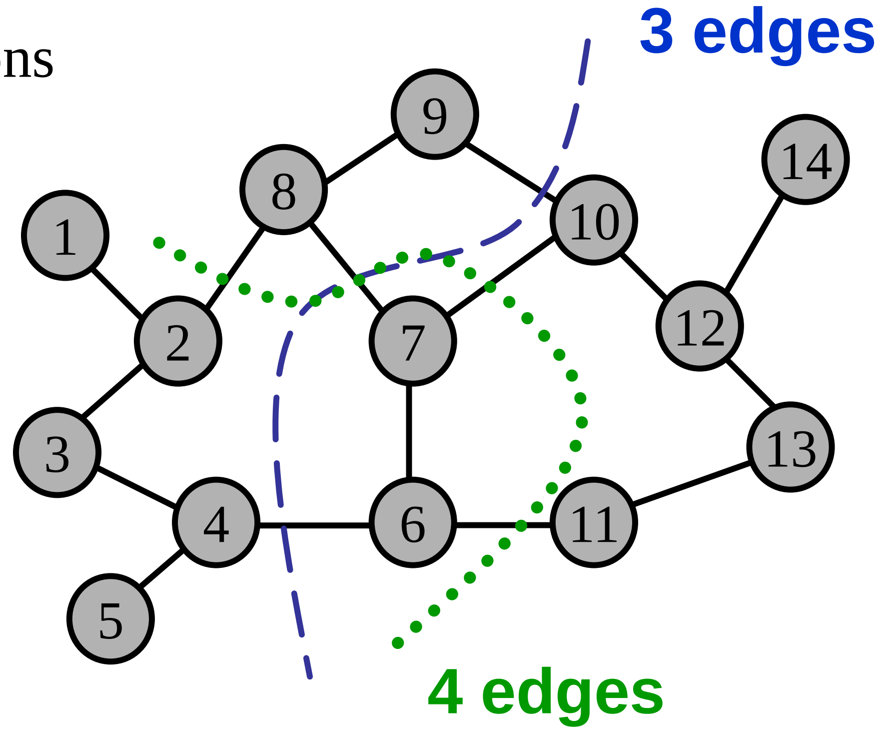
$$E = \{(1,2), \ldots, (12,14)\}$$

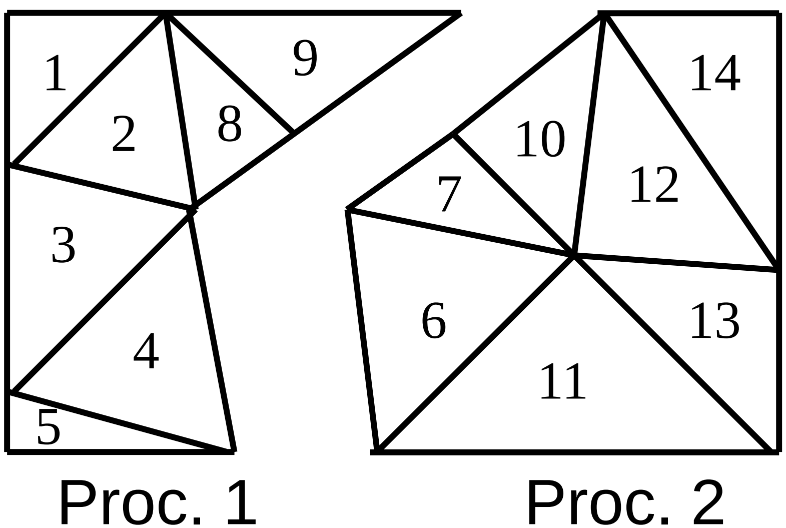
$$W_e \equiv 1,\ \ W_v \equiv 1$$

Examples for Graph Partitioning

- Parallel Finite-Element Simulations



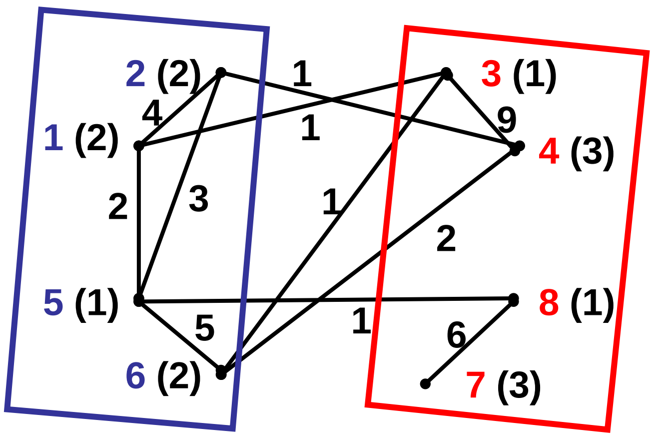




A good partitioning G_{FE} results in
- equal #elements/processor („load“ and „storage balancing“).
- Minimal #edges between P1 and P2 (minimal communication volume).

Definition of Graph Partitioning: Bisection

- 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

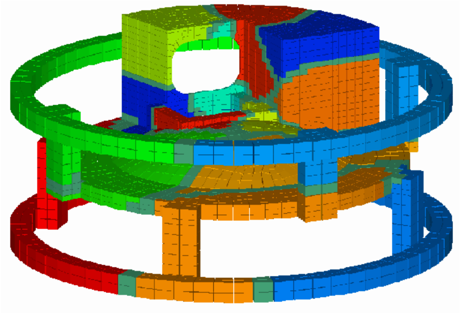
$$\min |\{e_{IJ} \epsilon E \mid v_i \,\epsilon V_1 \text{ und } v_j \,\epsilon\, V_2\}|$$

Heuristics — Overview of Bisection Algorithms

- Partitioning with nodal coordinates — e.g. each node has x,y,z coordinates → partition space

  Algorithms: **Recursive Coordinate Bisection**

  Inertial Partitioning

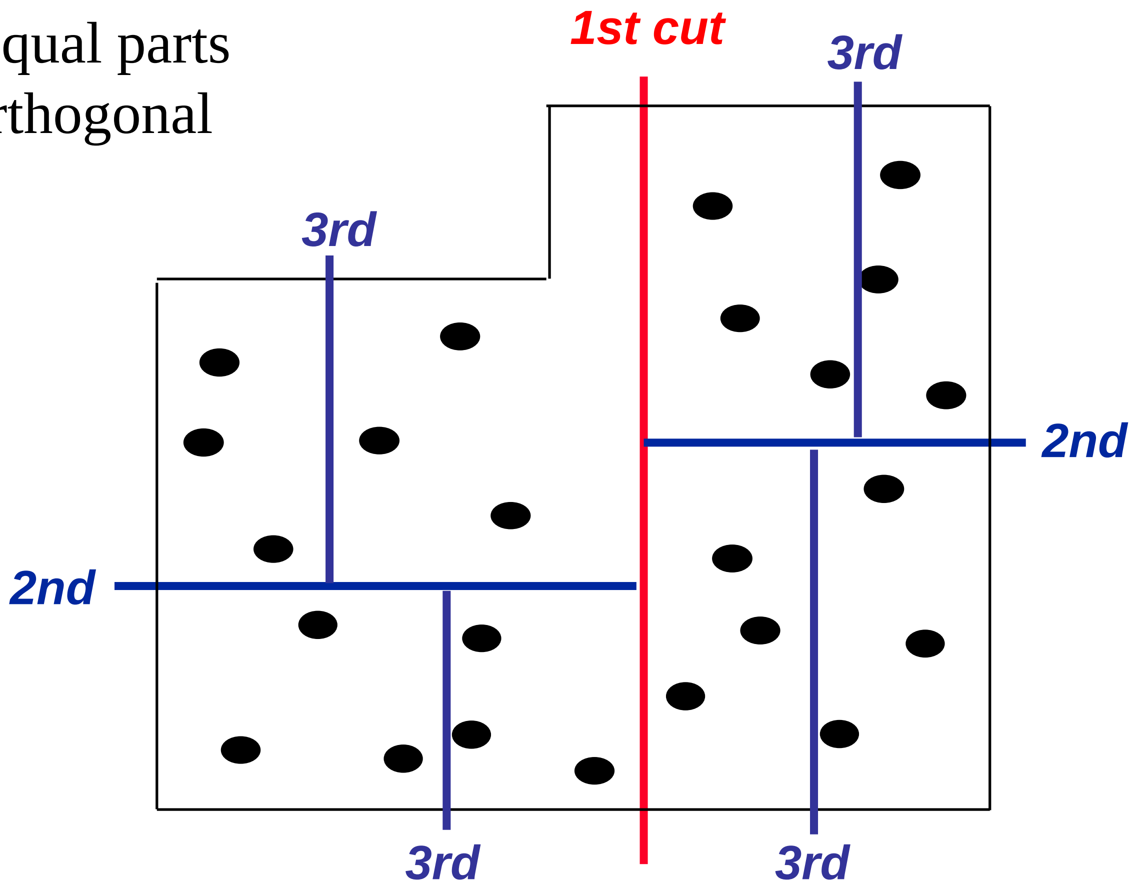


- Partitioning without Nodal Coordinates — e.g. indexing of web documents A(j,k) = # times keyword j appears in URL k

  Algorithms: **Spectral Methods**

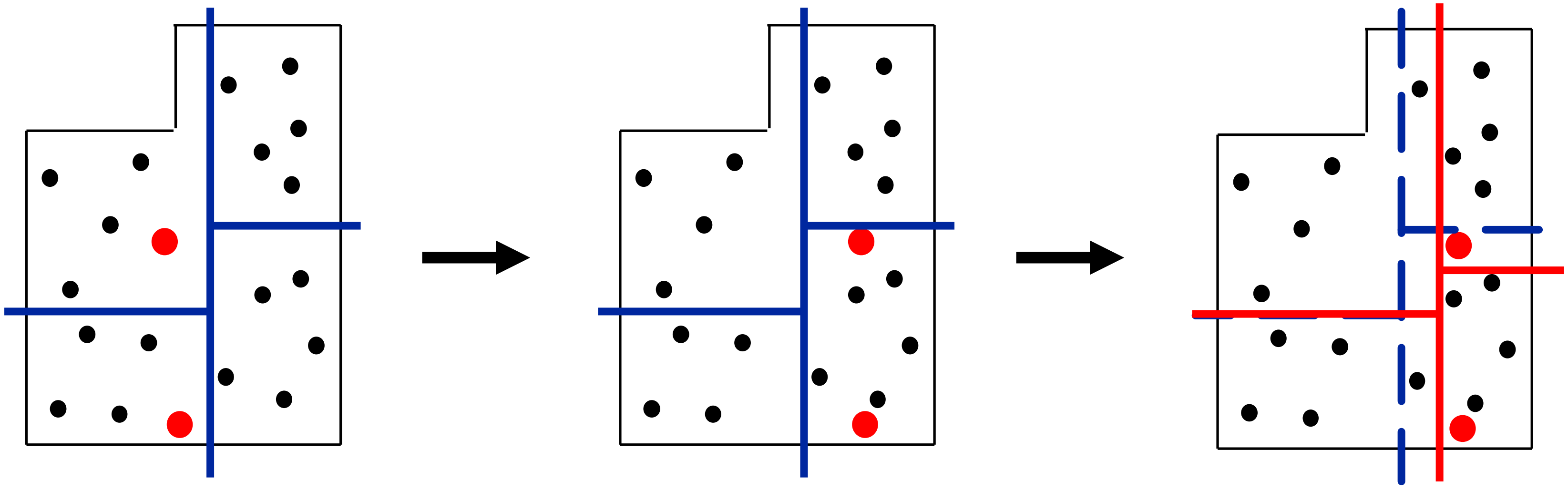
Nodal Coordinates — Recursive Coordinate Bisection (RCB)

- Developed by Berger & Bokhari (1987)
  - Independently discovered by others.
- Idea:
  - Divide work into two equal parts using a cutting plane orthogonal to a coordinate axis.
  - Recursively cut the resulting subdomains.



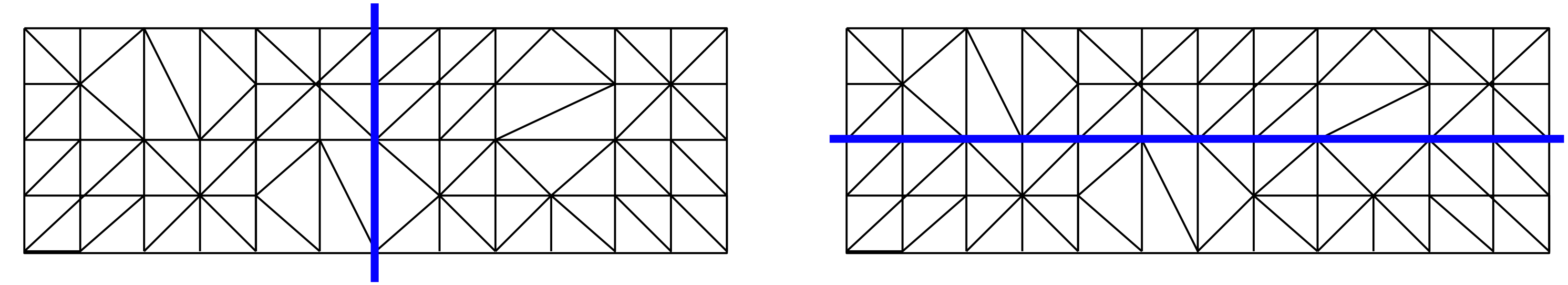
Nodal Coordinates — RCB Advantages

- Conceptually simple; fast and inexpensive.
- Regular subdomains.
  - Can be used for structured or unstructured applications.
- Effective when connectivity info is not available.
- **Incremental, but no control** of communication costs.

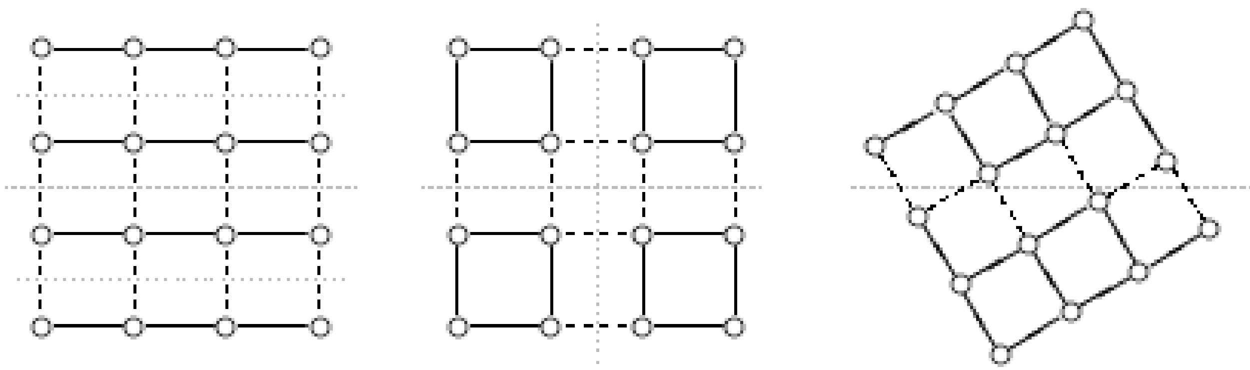


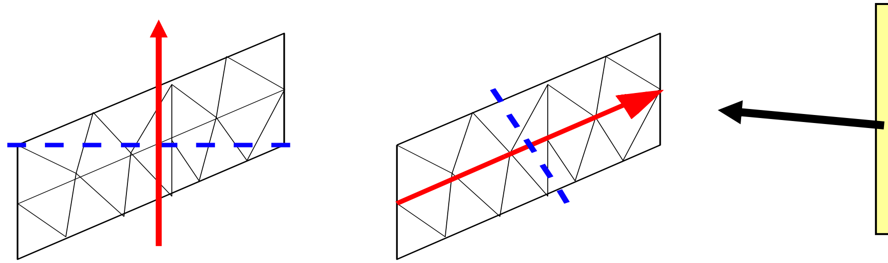
Nodal Coordinates — Coordinate Bisection

- Partition the domain along hyperplanes with node coordinates



-

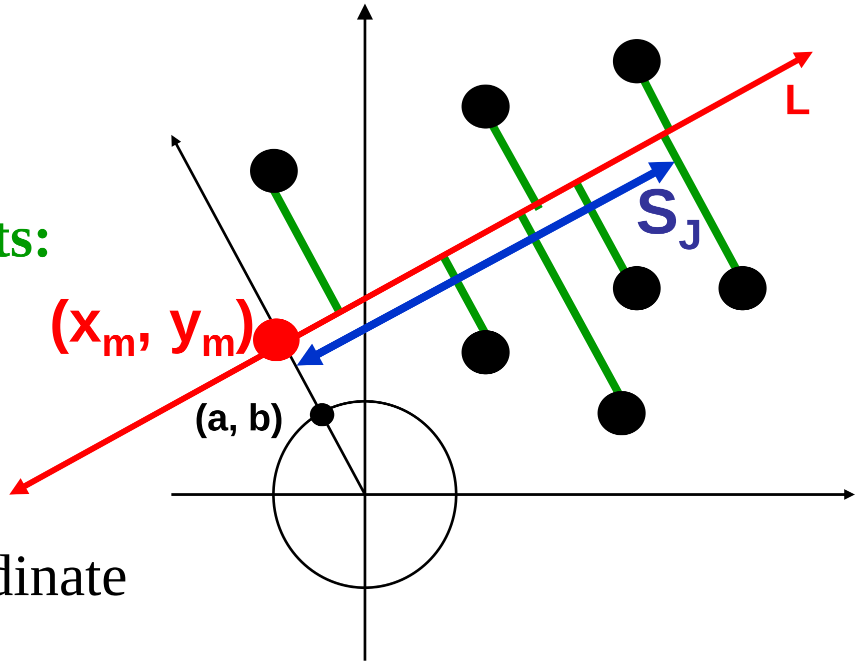




Good choice of coordinate system leads to inertial bisection

Nodal Coordinates — Inertial Partitioning

- Choose a line L, and then choose a line H orthogonal to it, with half the nodes on either side



1) Center of mass: x_m, y_m

(2) Choose a line L through the points:

L given by $a*(x-x_m)+b*(y-y_m)=0$ with $a^2+b^2=1$; (a, b) is a unit vector orthogonal to L

(3) Project each point to the line

For each $n_j = (x_j, y_j)$ compute coordinate $S_j = -b*(x_j-x_m) + a*(y_j-y_m)$ along L

(4) Compute the median

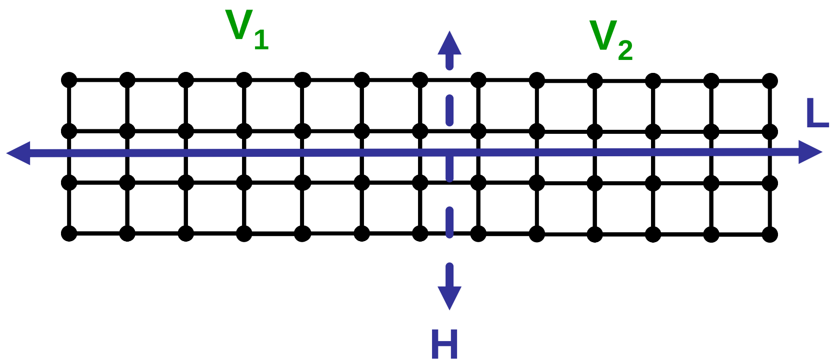
– Let $S_k = \text{median}(S_1, \ldots, S_n)$

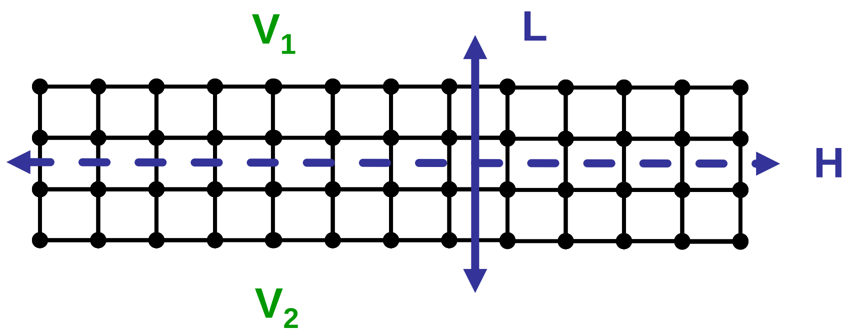
(5) Use median to partition the nodes

– Let nodes with $S_j < S_m$ be in V_1, rest in V_2

Nodal Coordinates — Inertial Partitioning, Choosing L

- Clearly prefer L on left below

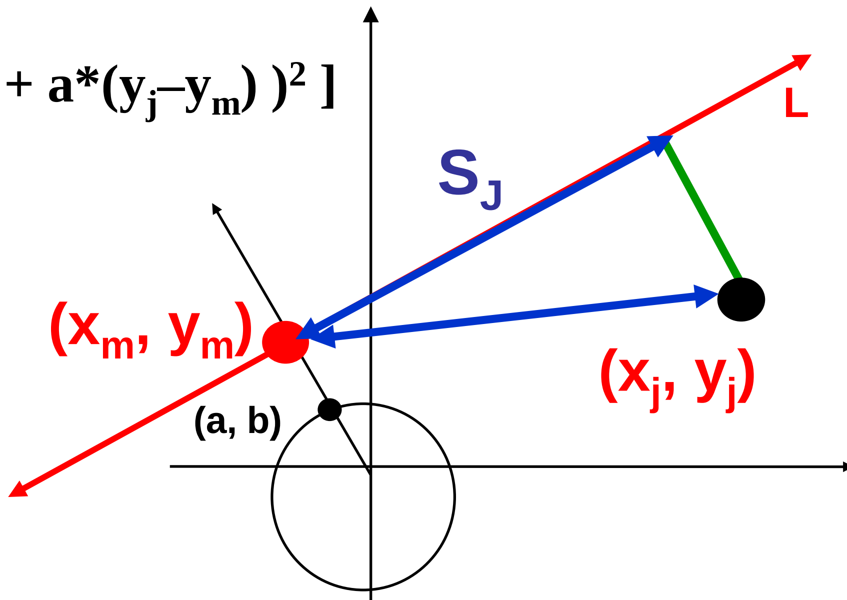




- Mathematically, choose L to be a **total least squares fit of the nodes**
  - Minimize sum of squares of distances to L (green lines on last slide)
  - Equivalent to choosing L as axis of rotation that minimizes the moment of inertia of nodes (unit weights) - source of name

Nodal Coordinates — Inertial Partitioning, Choosing L

- **$\sum_j$ (length of j-th green line)^2**
  $= \sum_J \; [\, (x_j - x_m)^2 + (y_j - y_m)^2 - (-b^*(x_j - x_m) + a^*(y_j - y_m))^2 \,]$
  … Pythagorean Theorem



$$= (1 - b^2) * \sum_j (x_j - x_m)^2 + 2*a*b* \sum_j (x_j - x_m)*(y_j - y_m) + (1-a^2) \sum_j (y_j - y_m)^2$$

$$= a^2 * \sum_j (x_j - x_m)^2 + 2*a*b* \sum_j (x_j - x_m)*(y_j - y_m) + b^2 \sum_j (y_j - y_m)^2$$

$$= a^2 * \quad X_1 \quad + 2*a*b* \quad X_2 \quad + b^2 * \quad X_3$$

$$= \begin{vmatrix} a & b \end{vmatrix} * \begin{vmatrix} X1 & X2 \\ X2 & X3 \end{vmatrix} * \begin{vmatrix} a \\ b \end{vmatrix} = \underline{\text{minimum}} = \lambda$$

$\boxed{\text{Minimizing } \lambda \text{ ?}}$

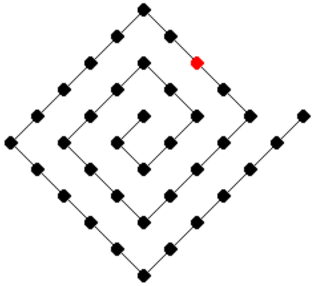
<u>Minimized</u> by choosing
$(x_m , y_m) = (\sum_j x_j , \sum_j y_j) / n$ = center of mass
(a,b

$$\boxed{\begin{array}{l} M\,u = \lambda\,u \leftrightarrow \\ u^T M\,u = u^T \lambda\,u = \lambda\,u^T u = \lambda \end{array}}$$

Nodal Coordinates — 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

Example (graph that is not spatial connected)



In the printed version, the solutions can be found in the appendix

Content

- Motivation for graph partitioning
- Overview of heuristics
- Partitioning with nodal coordinates
  - Ex: In finite element models, node at point in (x,y,z) space
- **Partitioning without Nodal Coordinates**
  - **Ex: In model of WWW, nodes are web pages**

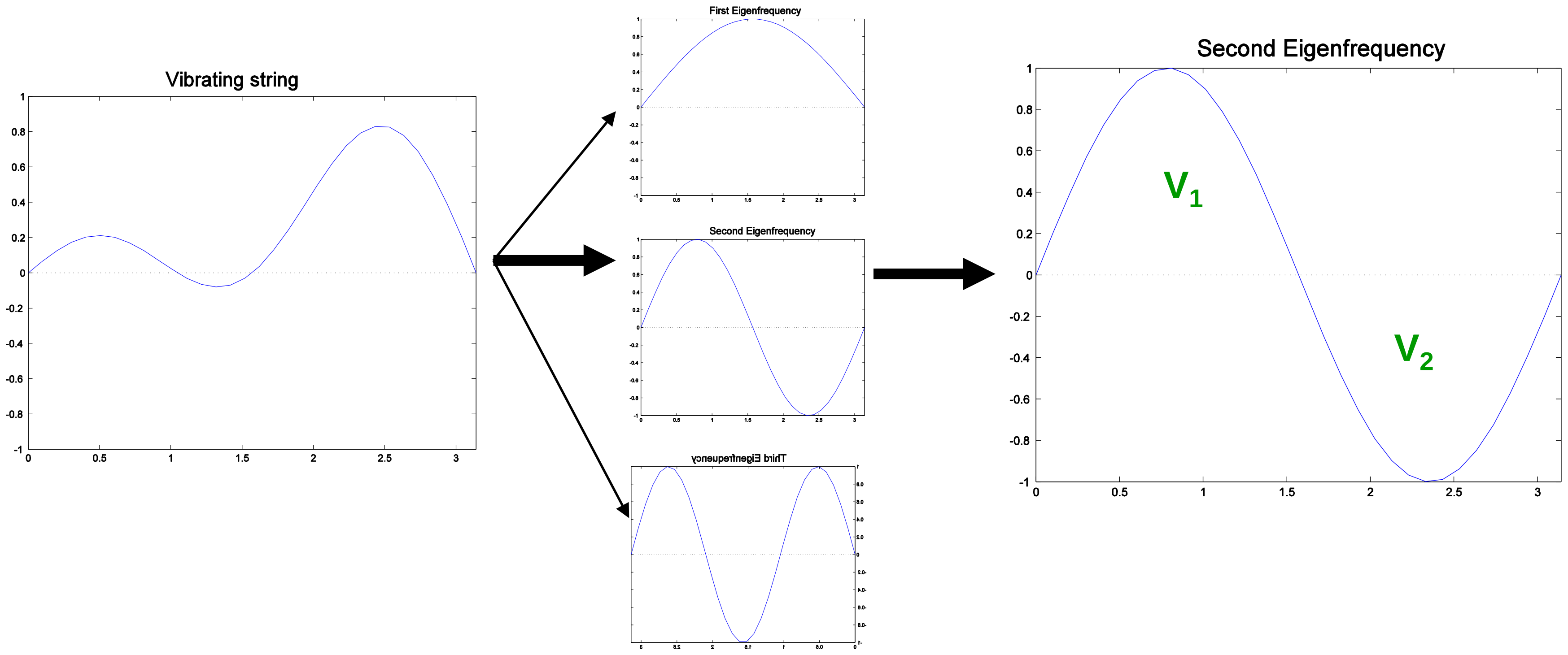
    Spectral Methods

Coordinate-Free — Spectral Methods

- **Spectral methods** as an example for **global partitioning** algorithms
- Heavily use of Eigenvalue/Eigenvector analysis

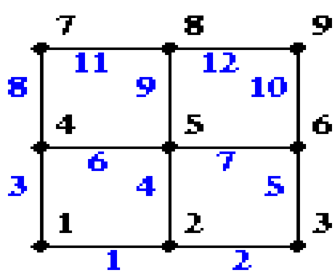
Coordinate-Free — Spectral Methods

- Based on theory of Fiedler (1970s), popularized by Horst Simon (1995)
- First motivation with vibrating string
- Label nodes by whether mode - or + to partition into V_1 and V_2



Coordinate-Free — Spectral Methods, Basic Definitions

- **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



$$
\begin{array}{c|ccccccccc|}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\hline
1 & 2 & -1 & & -1 & & & & & \\
2 & -1 & 3 & -1 & & -1 & & & & \\
3 & & -1 & 2 & & & -1 & & & \\
4 & -1 & & & 3 & -1 & & -1 & & \\
5 & & -1 & & -1 & 4 & -1 & & -1 & \\
6 & & & -1 & & -1 & 3 & & & -1 \\
7 & & & & -1 & & & 2 & -1 & \\
8 & & & & & -1 & & -1 & 3 & -1 \\
9 & & & & & & -1 & & -1 & 2
\end{array}
$$

Properties of Laplacian matrices

- **<u>Theorem:</u>** 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*e = 0
  - The eigenvalues of L(G) are **<u>nonnegative</u>**:

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

- **<u>Definition</u>**: 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

Relation between Laplace Matrix and Graph Partitioning

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

$$x(i) = +1 \quad \text{if } v_i \text{ in } N_+$$

$$x(i) = -1 \quad \text{if } v_i \text{ in } N_-.$$

Then we have the following property:

$$\boxed{\textbf{\#edge-cut between } \mathbf{N_+} \textbf{ and } \mathbf{N_-} = \mathbf{\tfrac{1}{4} * x^T * L(G) * x}}$$

Proof: (next slide)

Relation between Laplace Matrix and Graph Partitioning

$$
\begin{aligned}
x^T \cdot L(G) \cdot x &= \sum_j \sum_i L(G)_{(i,j)} \cdot x_i \cdot x_j \\
&= \sum_{i=j} L(G)_{(i,i)} \cdot x_i^2 + \sum_{i \neq j} L(G)_{(i,j)} \cdot x_i \cdot x_j \\
&= \sum_{i=j} L(G)_{(i,i)} \cdot x_i^2 \\
&\quad + \sum_{i \neq j;\ i,\, j \in N^+} L(G)_{(i,j)} \cdot x_i \cdot x_j + \sum_{i \neq j;\ i,\, j \in N^-} L(G)_{(i,j)} \cdot x_i \cdot x_j \\
&\quad + \sum_{i \neq j;\ i \in N^+,\, j \in N^-} L(G)_{(i,j)} L(G)_{(i,j)} \cdot x_i \cdot x_j \\
&= \sum_i degree(i) \\
&\quad + \sum_{i \neq j;\ i,\, j \in N^+} (-1) \cdot (+1) \cdot (+1) + \sum_{i \neq j;\ i,\, j \in N^-} (-1) \cdot (-1) \cdot (-1) \\
&\quad + \sum_{i \neq j;\ i \in N^+,\, j \in N^-} (-1) \cdot (+1) \cdot (-1)
\end{aligned}
$$

Relation between Laplace Matrix and Graph Partitioning

$$
\begin{aligned}
x^T \cdot L(G) \cdot x &= \sum_{i,\, j} L(G)_{(i,j)} \cdot x_i \cdot x_j \\
&= \sum_{i} degree(i) \\
&\quad + \sum_{i \neq j;\ i,\, j \in N^+} (-1) \cdot (+1) \cdot (+1) + \sum_{i \neq j;\ i,\, j \in N^-} (-1) \cdot (-1) \cdot (-1) \\
&\quad + \sum_{i \neq j;\ i \in N^+,\, j \in N^-} (-1) \cdot (+1) \cdot (-1) \\
&= \quad 2 \cdot \textbf{\#edges in G} \\
&\quad -2 \cdot (\textbf{\#edges connecting node in } N^+ \textbf{ to nodes in } N^+) \\
&\quad -2 \cdot (\textbf{\#edges connecting node in } N^- \textbf{ to nodes in } N^-) \\
&\quad +2 \cdot (\textbf{\#edges connecting node in } N^+ \textbf{ to nodes in } N^-) \\
&= 4 \cdot (\textbf{\#edges connecting node in } N^+ \textbf{ to nodes in } N^-)
\end{aligned}
$$

Relation between Laplace Matrix and Graph Partitioning

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

1.	$\lvert V_1 \rvert = \lvert V_2 \rvert$	$\Leftrightarrow \sum_i x(i) = 0$
2.	min #cut edges between V_1 and V_2	$\Leftrightarrow$ min $x^T * L(G) * x$

or

min	$f(x) = \frac{1}{4}\, x^T * L(G) * x$
constraints	$x_I = \{+/- 1\}$, $x^T * x = n$
	$x^T * e = 0$ with $e = [1, 1, \ldots, 1]^T$

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

min	$f(z) = \frac{1}{4}\, z^T * L(G) * z$
constraints	$z^T * z = n$, z real vector
	$z^T * e = 0$ with $e = [1, 1, \ldots, 1]^T$

Relation between Laplace Matrix and Graph Partitioning

- Let’ try to solve the continuous graph bisection problem

Relation between Laplace Matrix and Graph Partitioning

- Minimal solution of $z^T * L(G) * z$ is easy to find.
- $L(G)$ is symmetric $\rightarrow$ $L(G)$ has n orthonormal eigenvectors $u_1, \ldots, u_n$ with eigenvalues $0 = \lambda_1 \leq \ldots \leq \lambda_n$ and $u_1 = \mathrm{sqrt}(n) * e,\ \ e = [1, 1, \ldots, 1]^T$.
- A vector z is a linear combination of eigenvectors u_i:

$$z = \sum \alpha_i u_i = \alpha_1 u_1 + \alpha_2 u_2 + \ldots + \alpha_n u_n.$$

- **First constrained**: $z^T * e = 0$ or $z^T * u_1 = 0$ it is necessary that

$$z^T * u_1 = (\sum \alpha_i u_i)^T * u_1 = \alpha_1 u_1^T * u_1 = \alpha_1 \qquad \Rightarrow \boldsymbol{\alpha_1 = 0}$$

- **Second constrained:** $z^T * z = n$ it is necessary that

$$z^T * z = (\sum \alpha_i u_i)^T * (\sum \alpha_j u_j) = \sum\sum \alpha_i \alpha_j u_i^T * u_j = \sum \alpha_i^2 = n$$

- <u>**Minimize $4 * f(z) = z^T * L(G) * z$**</u>

$$z^T * L(G) * z = (\sum \alpha_i u_i) * L * (\sum \alpha_j u_j) = (\sum \alpha_i u_i)^T * (\sum \alpha_j \lambda_j u_j) =$$

$$\sum\sum \alpha_j \alpha_i \lambda_j u_i^T * u_j = \sum \alpha_i^2 \lambda_j \geq \lambda_2 \sum \alpha_i^2 = \boldsymbol{\lambda_2 * n}$$

Relation between Laplace Matrix and Graph Partitioning

- **Minimize $4*f(z) = z^T * L(G) * z$**

$$z^T*L(G)*z \quad = \quad (\sum \alpha_i u_i) *L*(\sum \alpha_j u_j) = (\sum \alpha_i u_i)^T*(\sum \alpha_j \lambda_j u_j) =$$
$$\sum\sum \alpha_j \alpha_i \lambda_j u_i^T*u_j = \sum \alpha_i^2 \lambda_j \quad \geq \quad \lambda_2 \sum \alpha_i^2 = \boldsymbol{\lambda_2 * n}$$

- Minimum is at $z = \mathrm{sqrt}(n)*u_2$.

- <u>**Spectral Bisection Algorithm**</u>**:**
  - Compute eigenvector u_2 corresponding to $\boldsymbol{\lambda}_2 (L (G))$
  - 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.

Demo – Partitioning in Matlab and Julia