Graph partitioning

Prof. Richard Vuduc

Georgia Institute of Technology

CSE/CS 8803 PNA: Parallel Numerical Algorithms

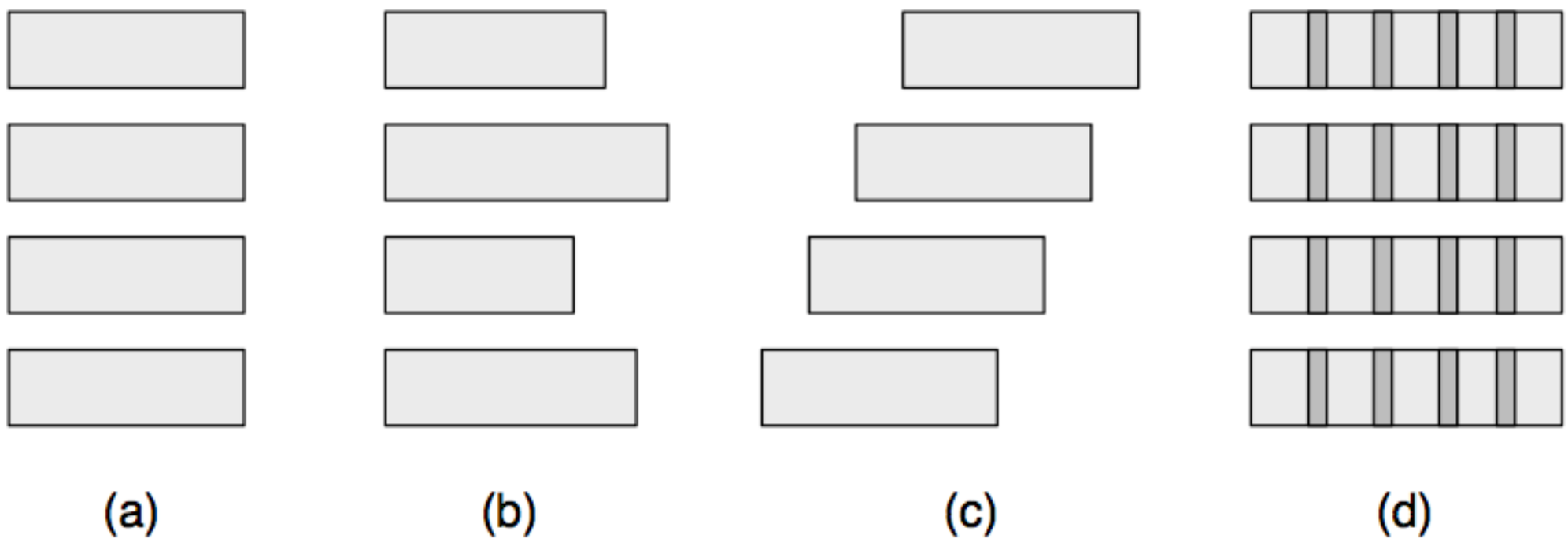
[L.27] Tuesday, April 22, 2008

Today's sources

- CS 194/267 at UCB (Yelick/Demmel)
- "Intro to parallel computing" by Grama, Gupta, Karypis, & Kumar

Review: Dynamic load balancing

Parallel efficiency: 4 scenarios



Consider **load balance**, **concurrency**, and **overhead**

Summary

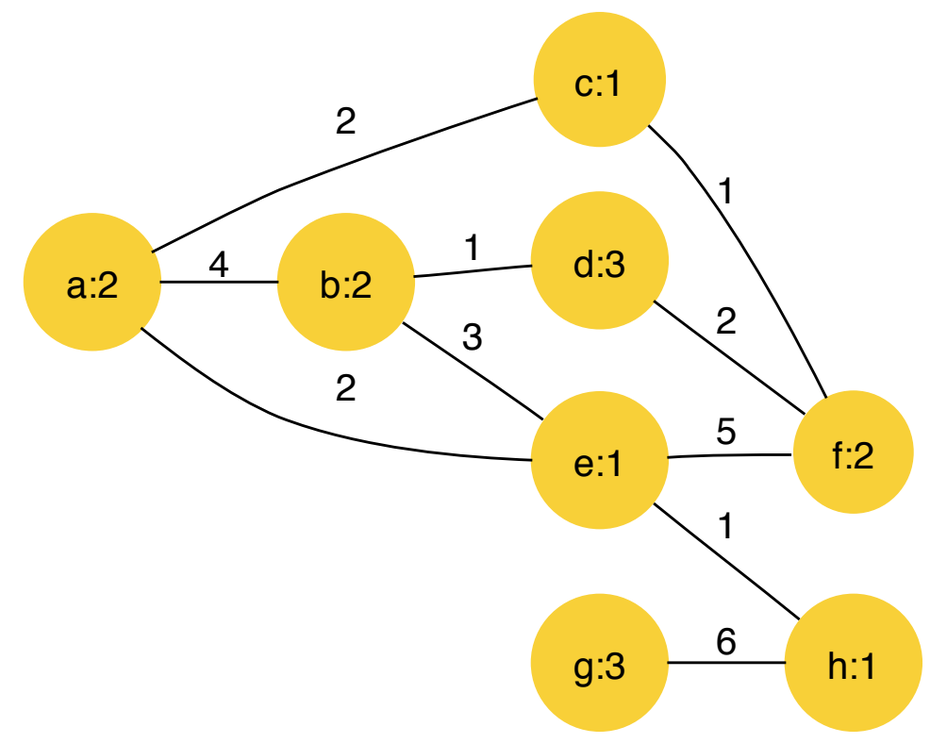
- Unpredictable loads → online algorithms
- Fixed set of tasks with unknown costs → self-scheduling
- Dynamically unfolding set of tasks → work stealing
- Theory ⇒ randomized should work well
- Other scenarios: What if…
  - … locality is of paramount importance? ⇒ Diffusion-based models?
  - … processors are heterogeneous? ⇒ Weighted factoring?
  - … task graph is known in advance? ⇒ Static case; graph partitioning (today)

Graph partitioning

Problem definition

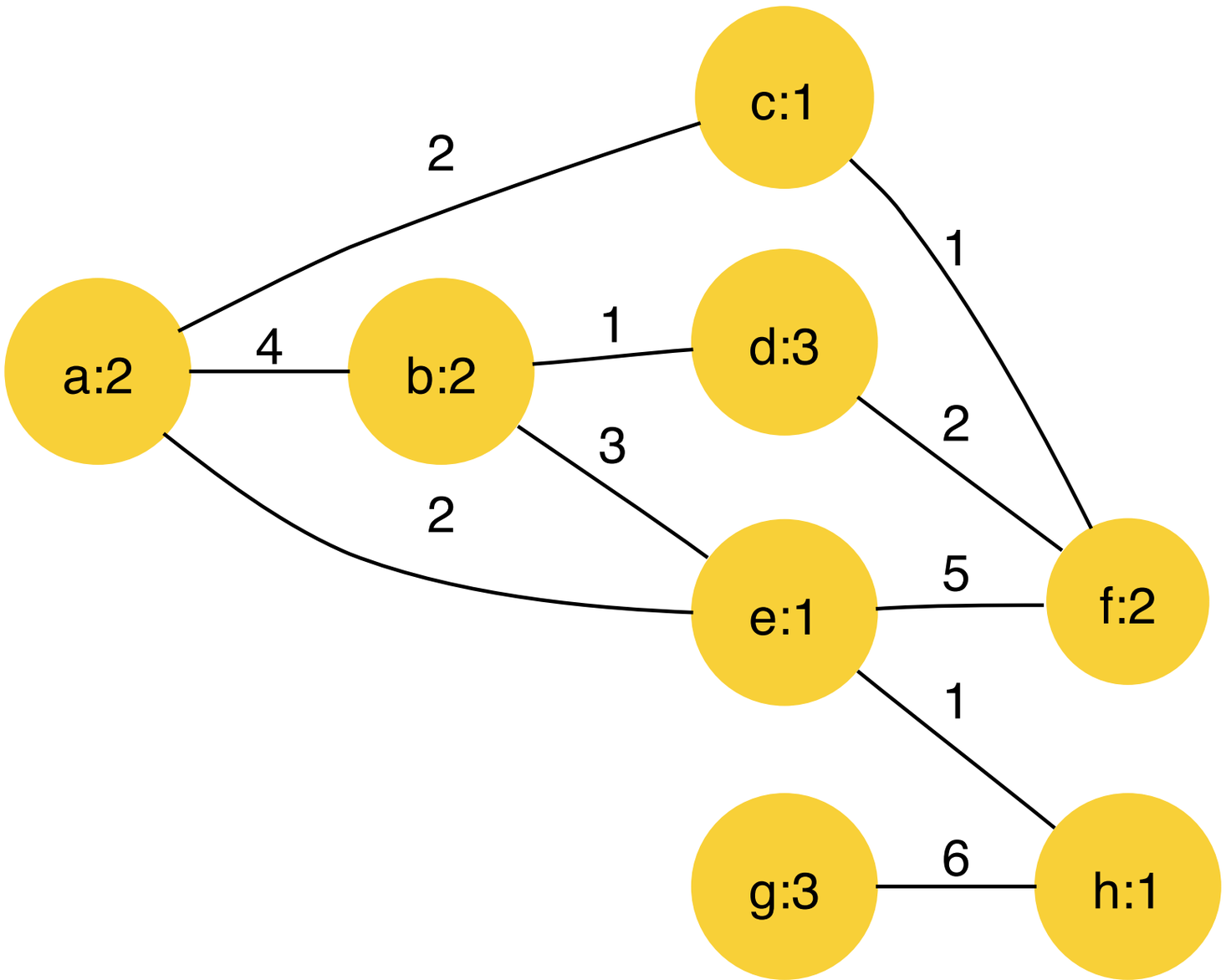
- Weighted graph

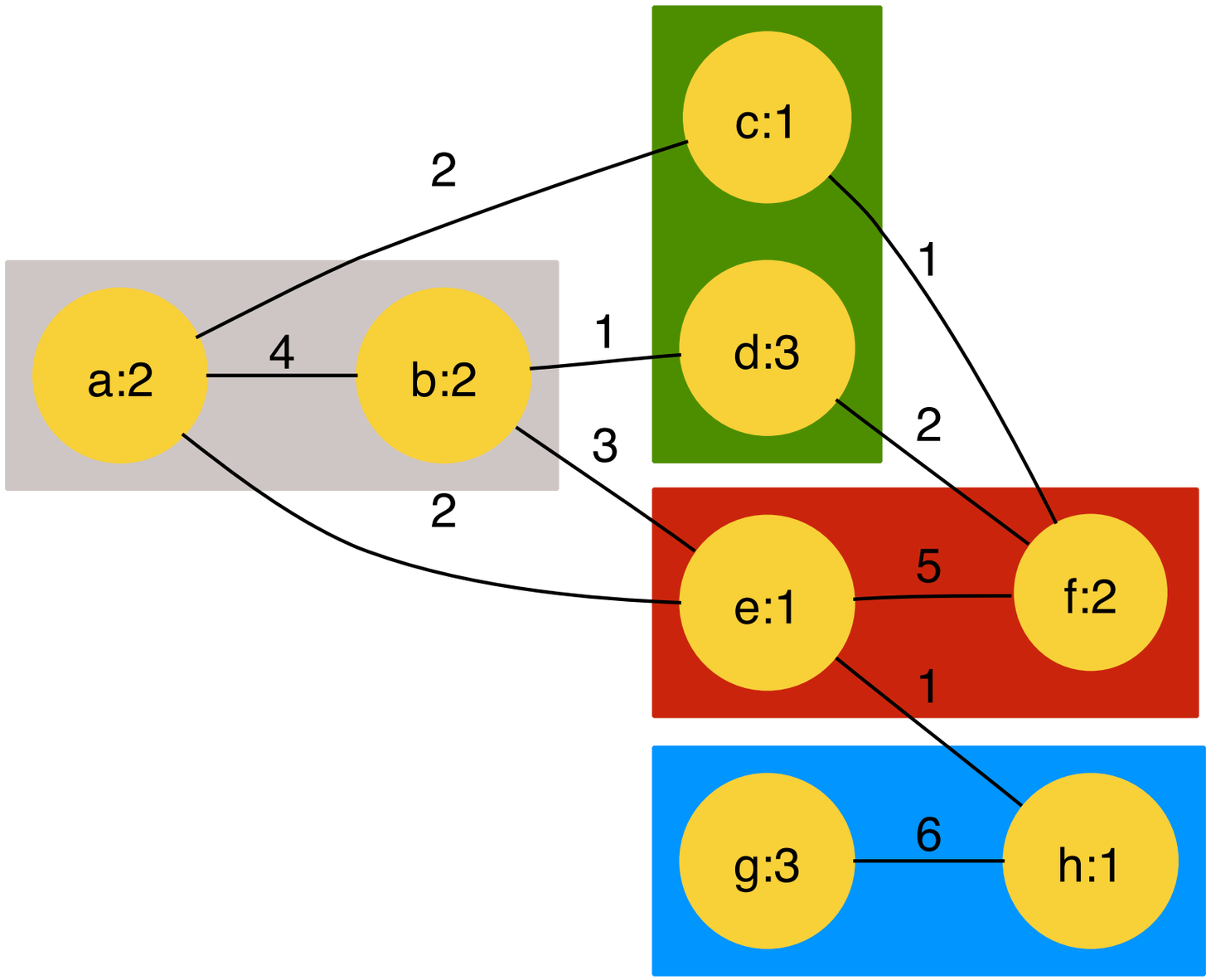
$$G = (V, E, W_V, W_E)$$



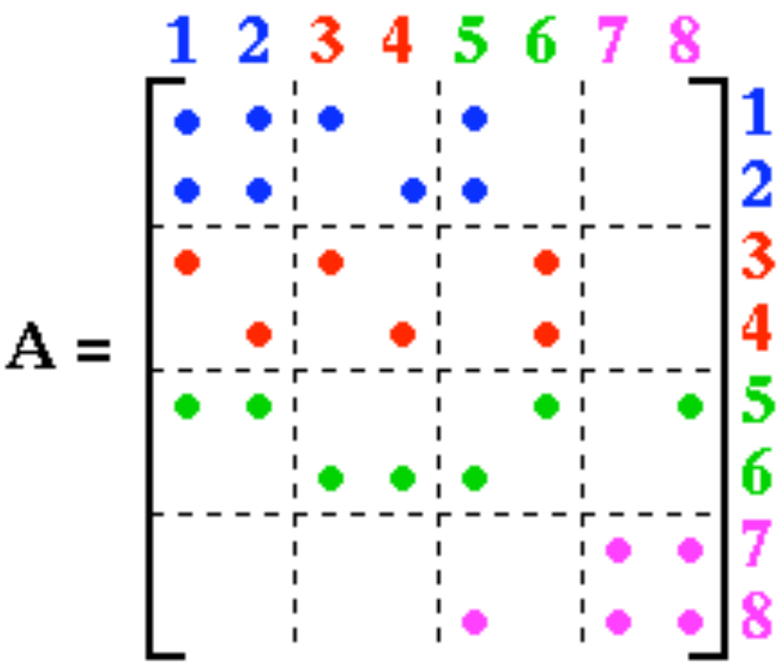
- Find partitioning of nodes s.t.:
  - Sum of node-weights ~ even
  - Sum of inter-partition edge-weights minimized

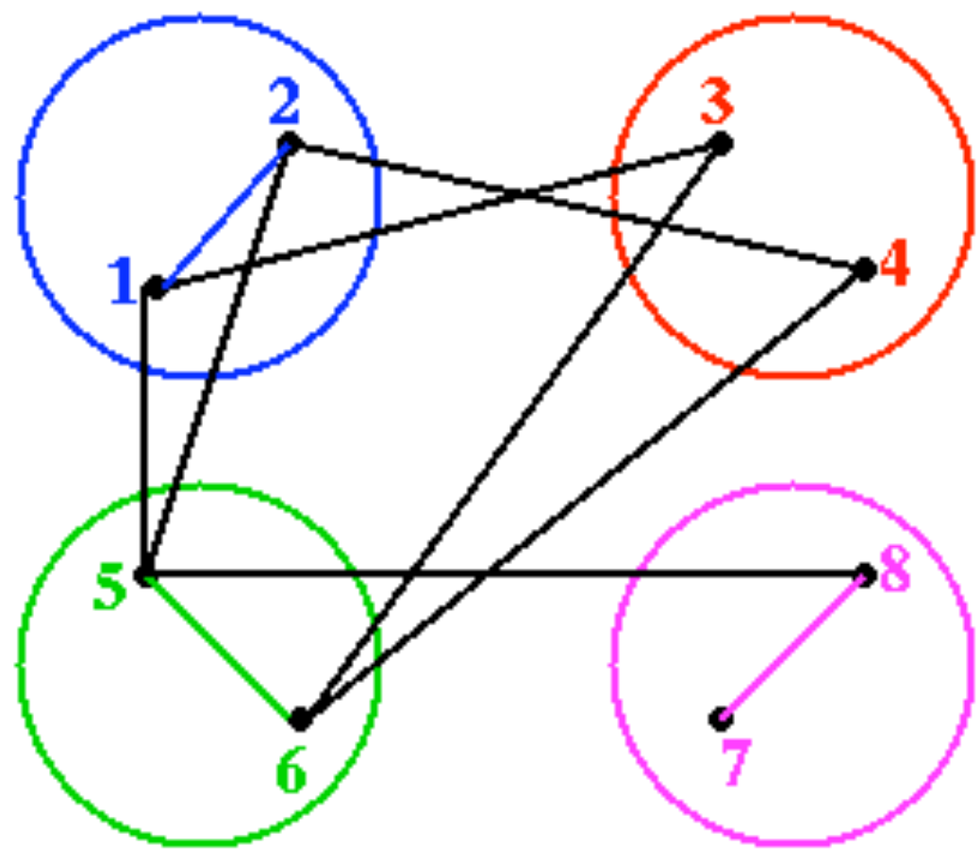
$$V = V_1 \cup V_2 \cup \cdots \cup V_p$$


c:1
2
1
a:2
4
b:2
1
d:3
3
2
2
5
e:1
f:2
1
6
g:3
h:1


c:1
2
1
a:2
4
b:2
1
d:3
3
2
2
e:1
5
f:2
1
g:3
6
h:1

Partitioning a Sparse Symmetric Matrix



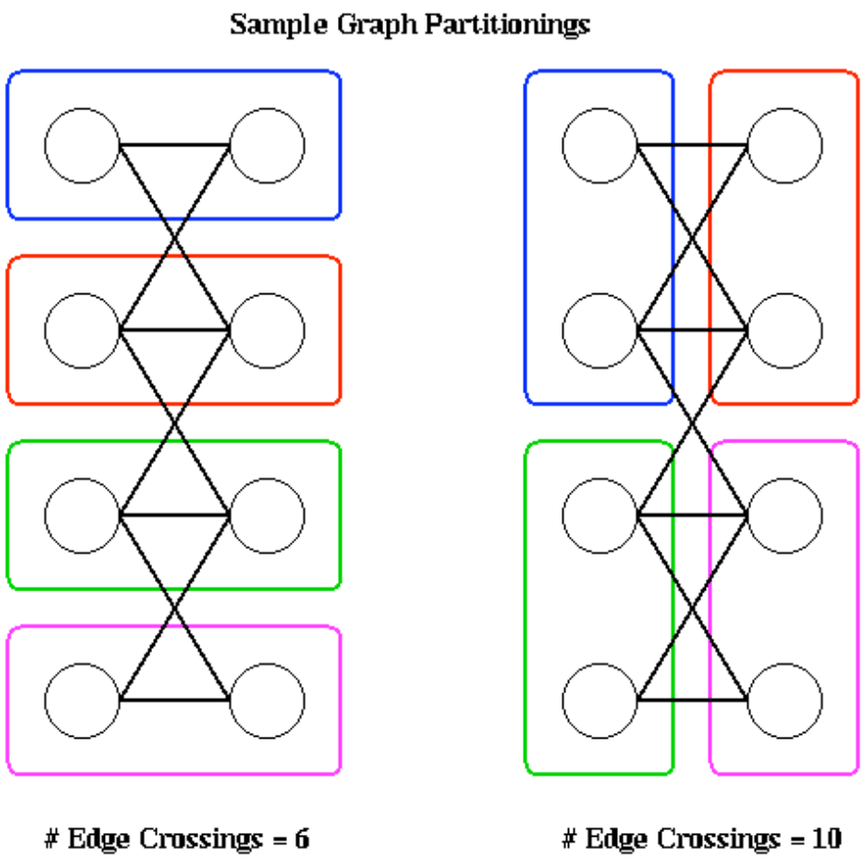


Cost of graph partitioning

- Many possible partitions
- Consider $V = V_1 \cup V_2$

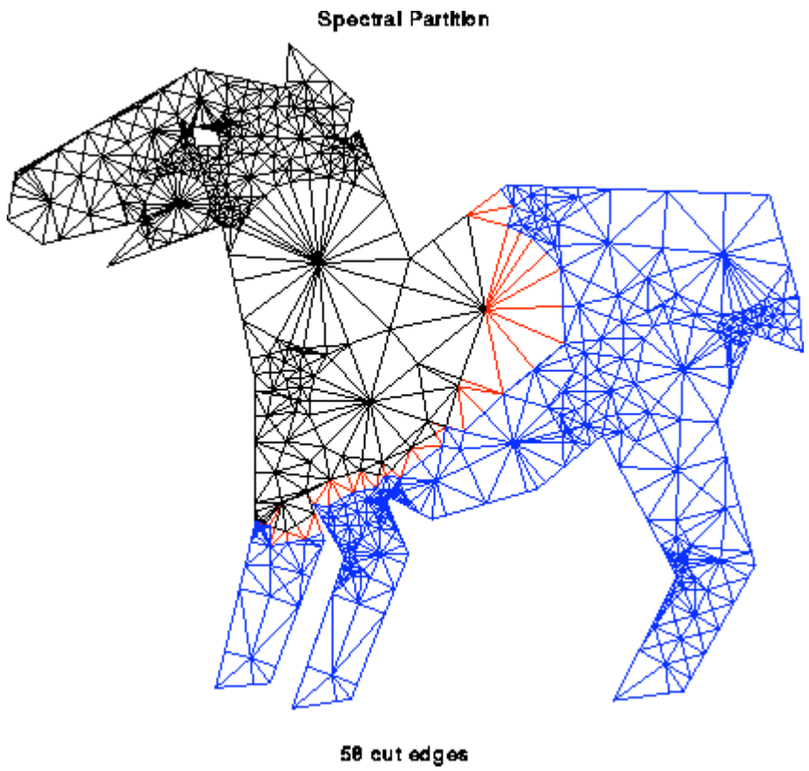
$$\binom{n}{\frac{n}{2}} \approx \sqrt{\frac{2}{\pi n}} \cdot 2^n$$

- Problem is NP-Complete, so need heuristics



First heuristic: Repeated graph bisection

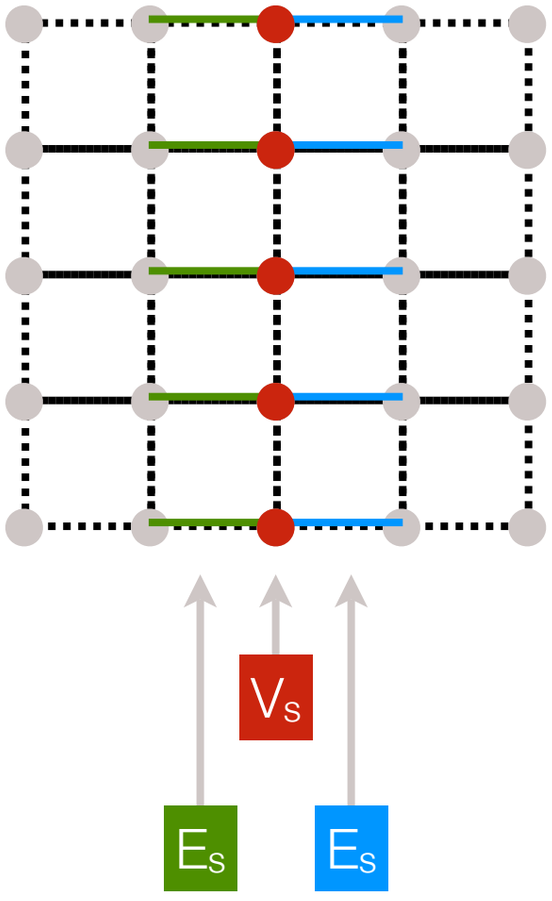
- To get 2^k partitions, bisect k times



Edge vs. vertex separators

- **Edge separator**: $E_s \subset E$, s.t. removal creates two disconnected components
- **Vertex separator**: $V_s \subset V$, s.t. removing Vs and its incident edges creates two disconnected components
- $E_s \rightarrow V_s$: $|V_s| \leq |E_s|$
- $V_s \rightarrow E_s$: $|E_s| \leq d \cdot |V_s|,$

$$d = \text{max degree}$$



Overview of bisection heuristics

- **With** nodal coordinates: Spatial partitioning
- **Without** nodal coordinates
- **Multilevel** acceleration: Use coarse graphs

Partitioning with nodal coordinates

Intuition: Planar graph theory

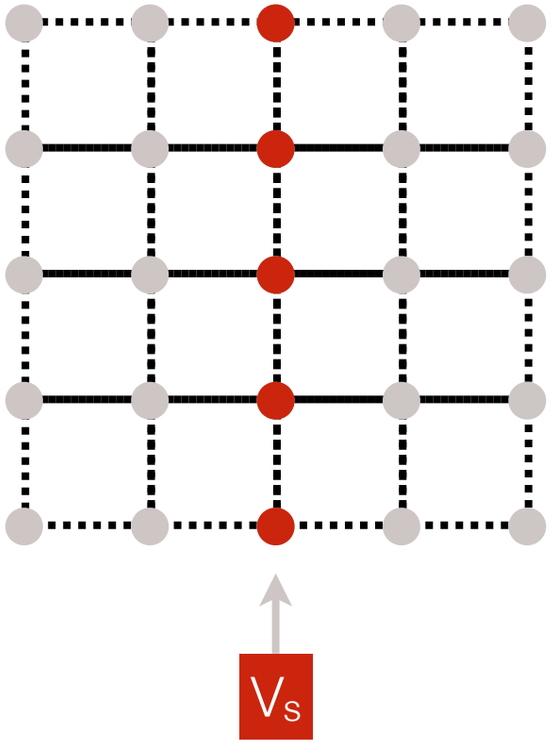
- **Planar graph**: Can draw G in the plane w/o edge crossings

- **Theorem** (Lipton & Tarjan '79): Planar $G \Rightarrow \exists\, V_s$ s.t.

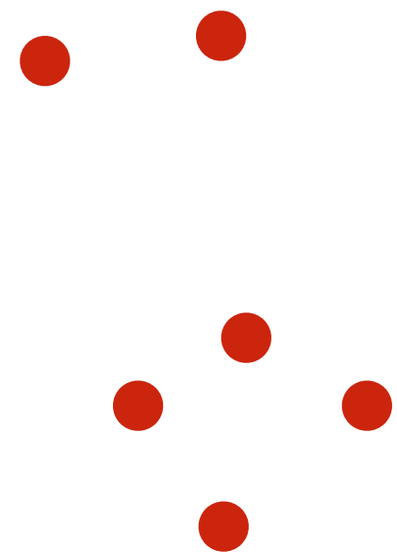
$$(1) \qquad V = V_1 \cup V_s \cup V_2$$

$$(2) \qquad |V_1|, |V_2| \leq \frac{2}{3}|V|$$

$$(3) \qquad |V_s| \leq \sqrt{8|V|}$$

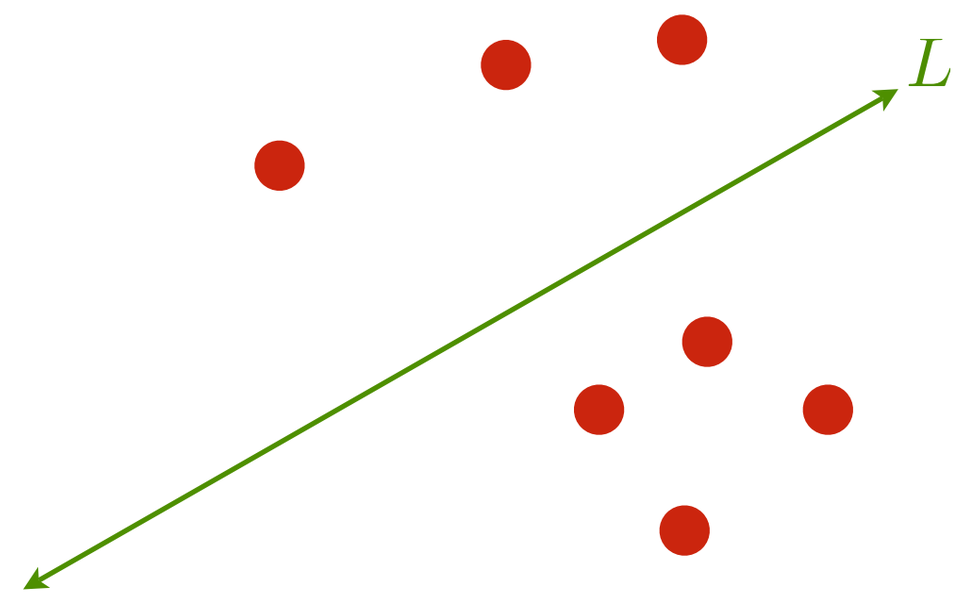


Inertial partitioning



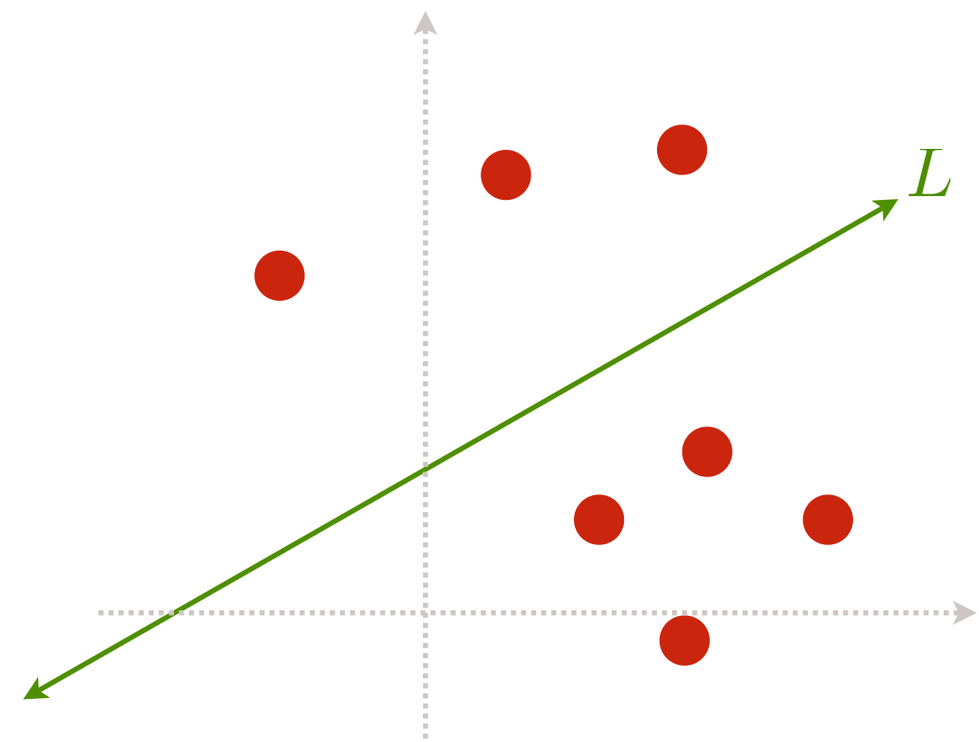
Inertial partitioning

- Choose line L



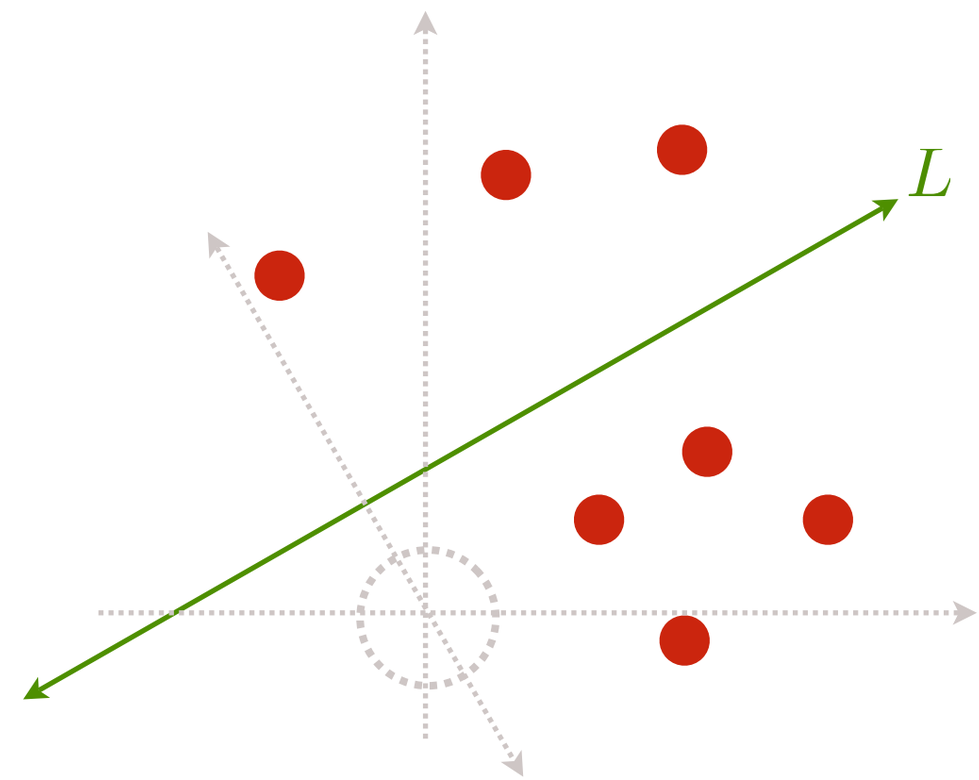
Inertial partitioning

- Choose line L



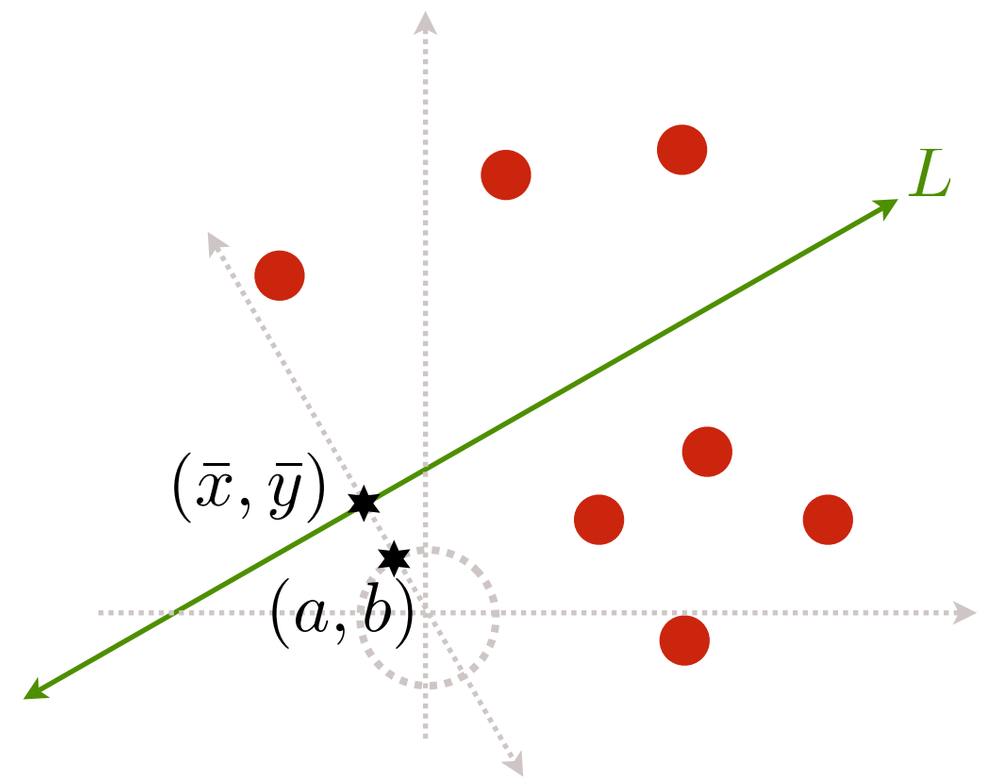
Inertial partitioning

- Choose line L



Inertial partitioning

- Choose line L

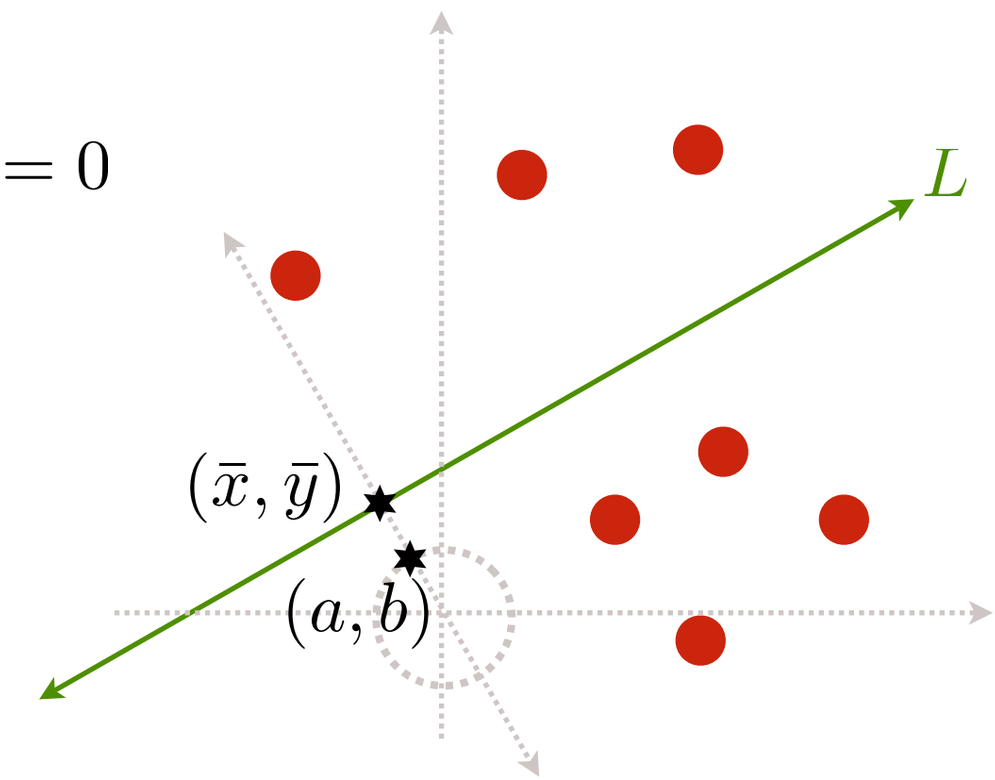


Inertial partitioning

- Choose line L

$$L: \quad a \cdot (x - \bar{x}) + b \cdot (y - \bar{y}) = 0$$

$$a^2 + b^2 = 1$$

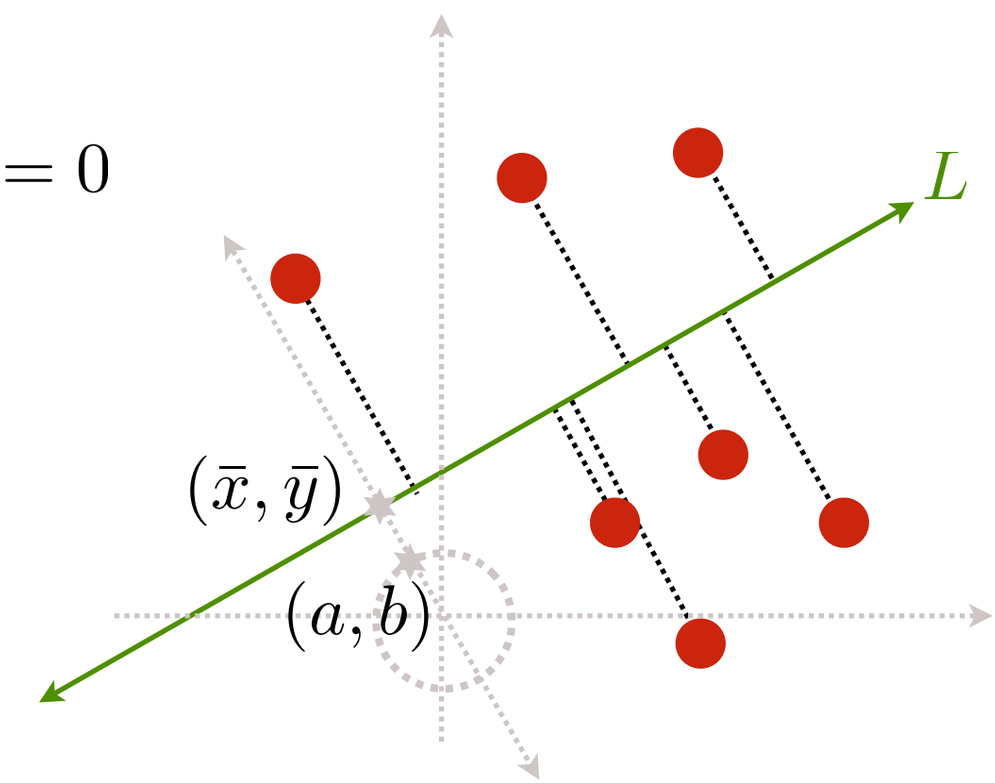


Inertial partitioning

- Choose line L

$$L: \quad a \cdot (x - \bar{x}) + b \cdot (y - \bar{y}) = 0$$
$$a^2 + b^2 = 1$$

- Project points onto L

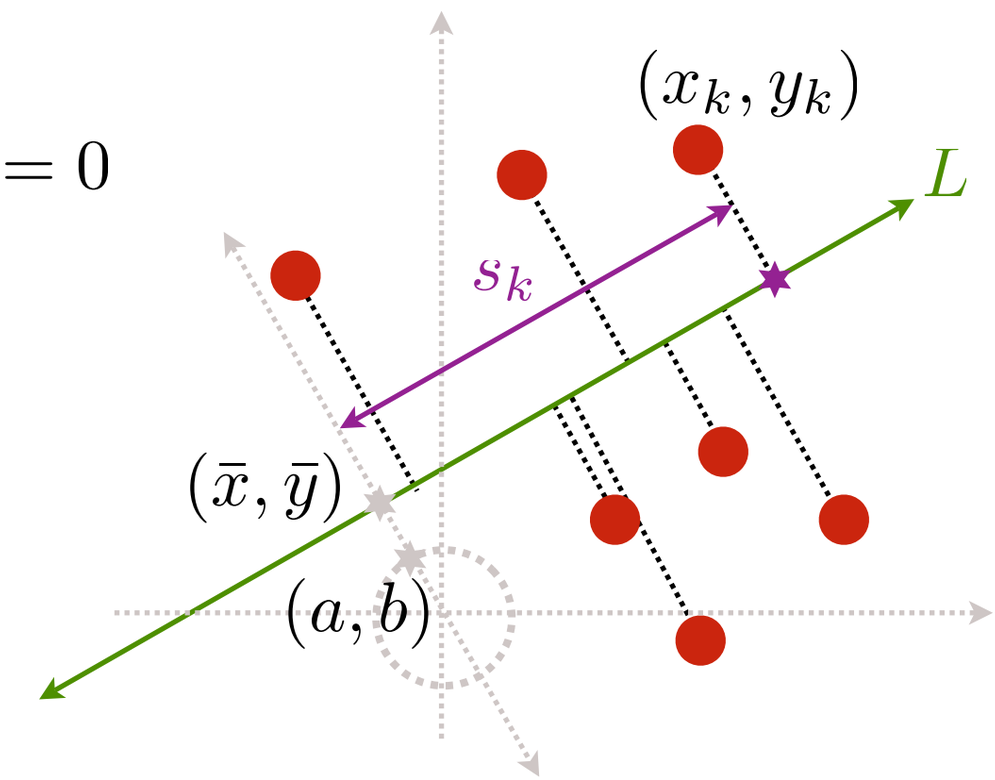


Inertial partitioning

- Choose line L

$$L: \quad a \cdot (x - \bar{x}) + b \cdot (y - \bar{y}) = 0$$
$$a^2 + b^2 = 1$$

- Project points onto L



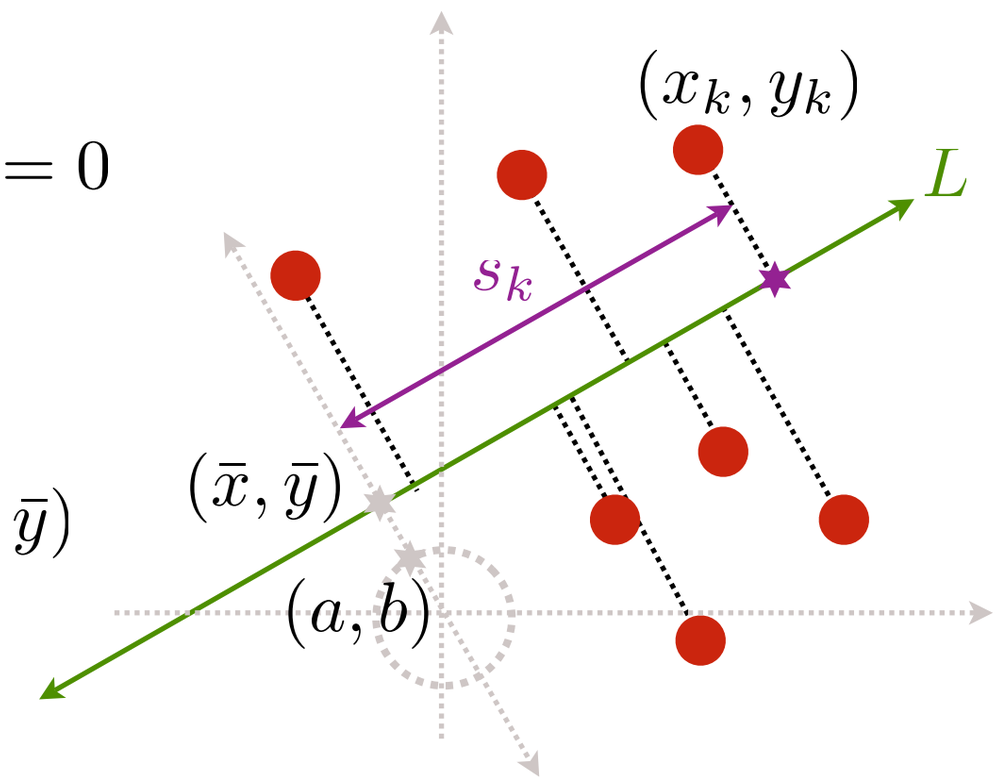
Inertial partitioning

- Choose line L

$$L: \quad a \cdot (x - \bar{x}) + b \cdot (y - \bar{y}) = 0$$
$$a^2 + b^2 = 1$$

- Project points onto L

$$s_k = -b \cdot (x_k - \bar{x}) + a \cdot (y_k - \bar{y})$$



Inertial partitioning

- Choose line L

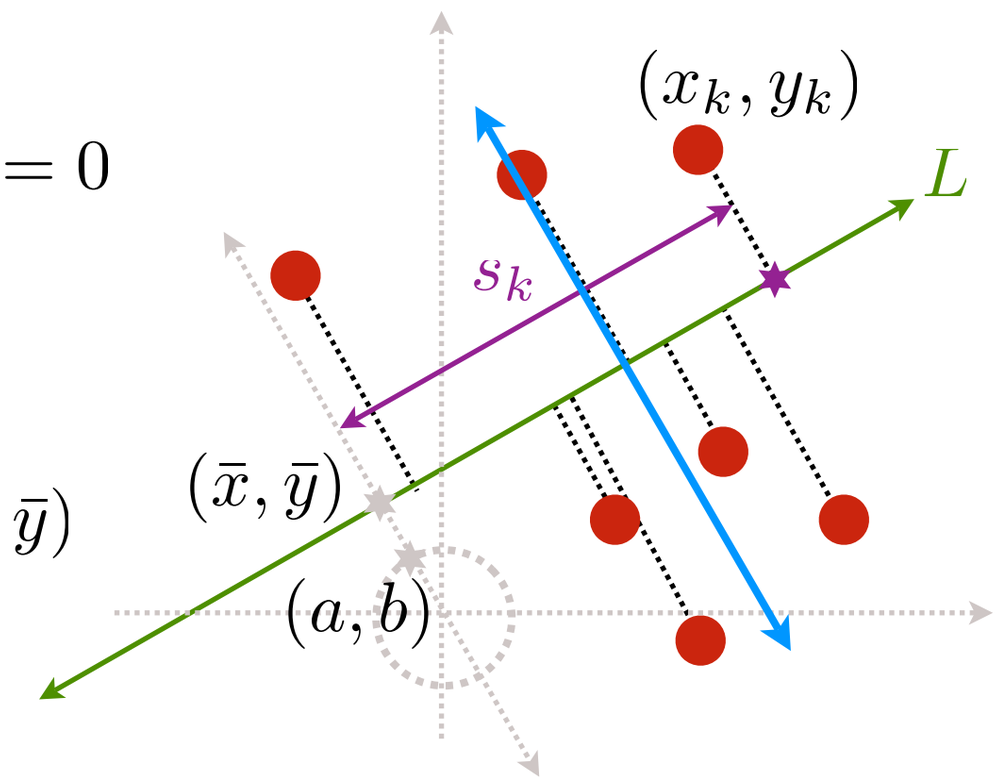
$$L: \quad a \cdot (x - \bar{x}) + b \cdot (y - \bar{y}) = 0$$
$$a^2 + b^2 = 1$$

- Project points onto L

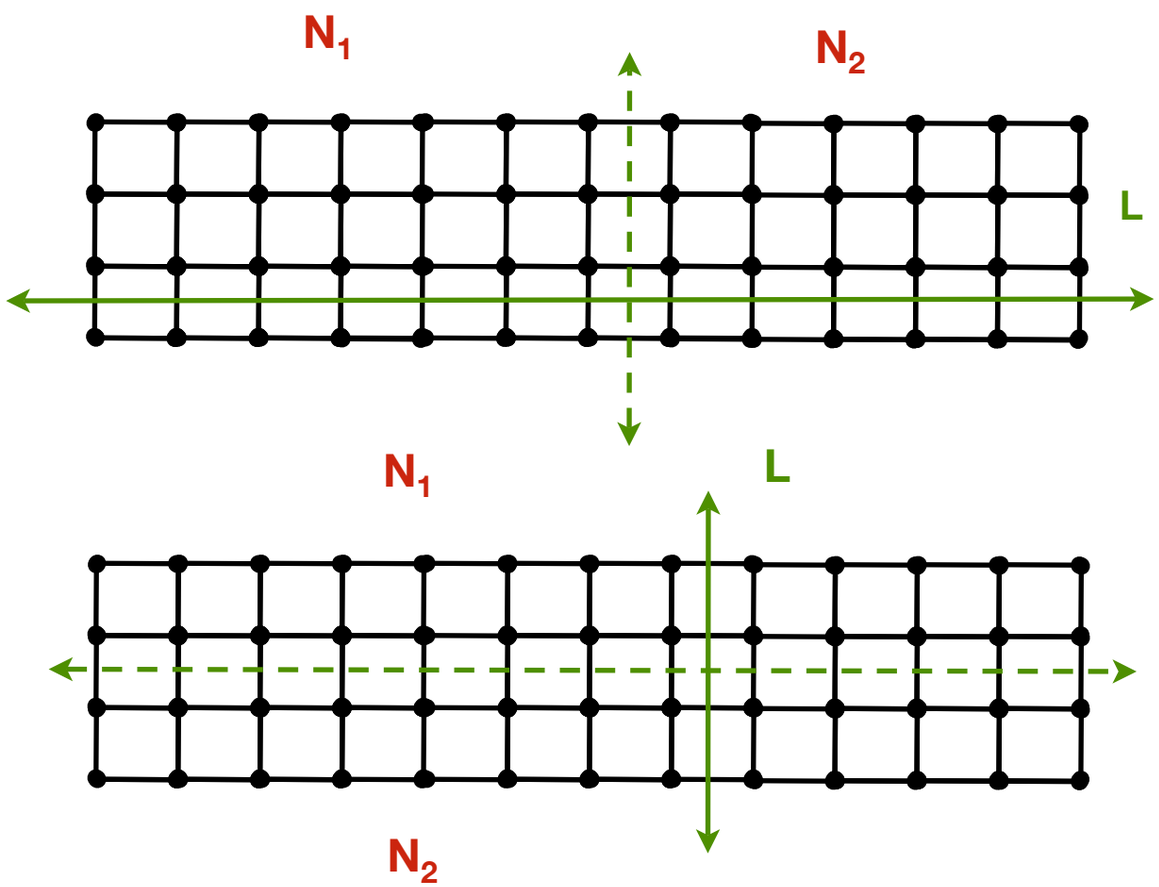
$$s_k = -b \cdot (x_k - \bar{x}) + a \cdot (y_k - \bar{y})$$

- Compute **median** and separate

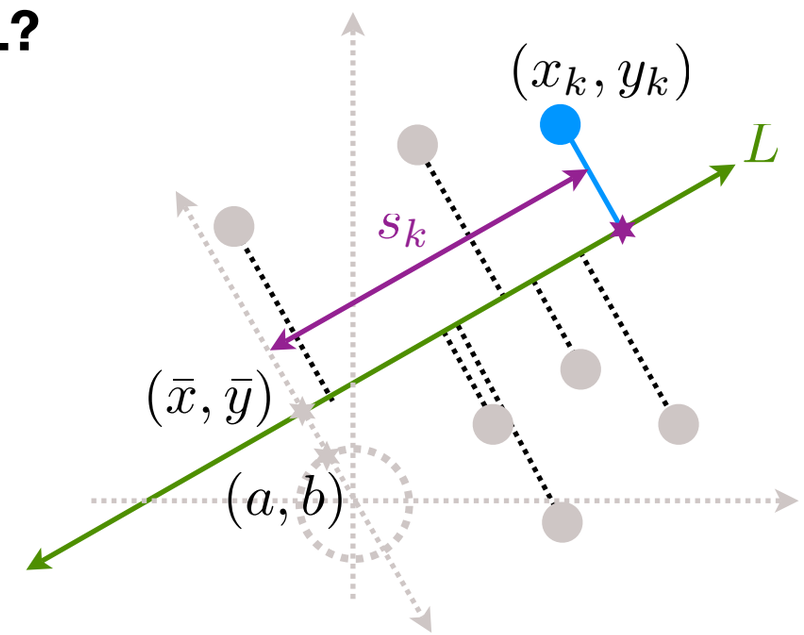
$$\bar{s} = \text{median}(s_1, \ldots, s_n)$$



How to choose L?



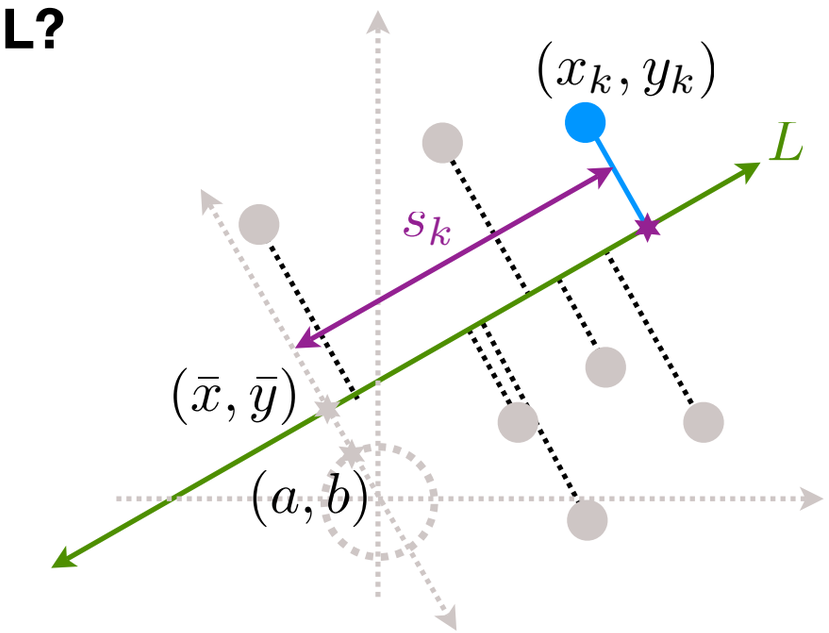
How to choose L?



Least-squares fit:
Minimize sum-of-square distances

$$
\begin{aligned}
& \sum_k (d_k)^2 \\
= \quad & \sum_k \left[(x_k - \bar{x})^2 + (y_k - \bar{y})^2 - (s_k)^2\right] \\
= \quad & \sum_k \left[(x_k - \bar{x})^2 + (y_k - \bar{y})^2 - (-b(x_k - \bar{x}) + a(y_k - \bar{y}))^2\right] \\
= \quad & a^2 \sum_k (y_k - \bar{y})^2 + 2ab \sum_k (x_k - \bar{x})(y_k - \bar{y}) + b^2 \sum_k (x_k - \bar{x})^2 \\
= \quad & a^2 \cdot \alpha_1 + 2ab \cdot \alpha_2 + b^2 \cdot \alpha_3 \\
= \quad & \begin{pmatrix} a & b \end{pmatrix} \cdot \begin{pmatrix} \alpha_1 & \alpha_2 \\ \alpha_2 & \alpha_3 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \end{pmatrix}
\end{aligned}
$$

How to choose L?



Least-squares fit:
Minimize sum-of-square distances

Interpretation:
Equivalent to choosing L as axis of rotation that minimizes moment of inertia.

$$\text{Minimize: } \sum_k (d_k)^2 = \begin{pmatrix} a & b \end{pmatrix} \cdot A(\bar{x}, \bar{y}) \cdot \begin{pmatrix} a \\ b \end{pmatrix}$$

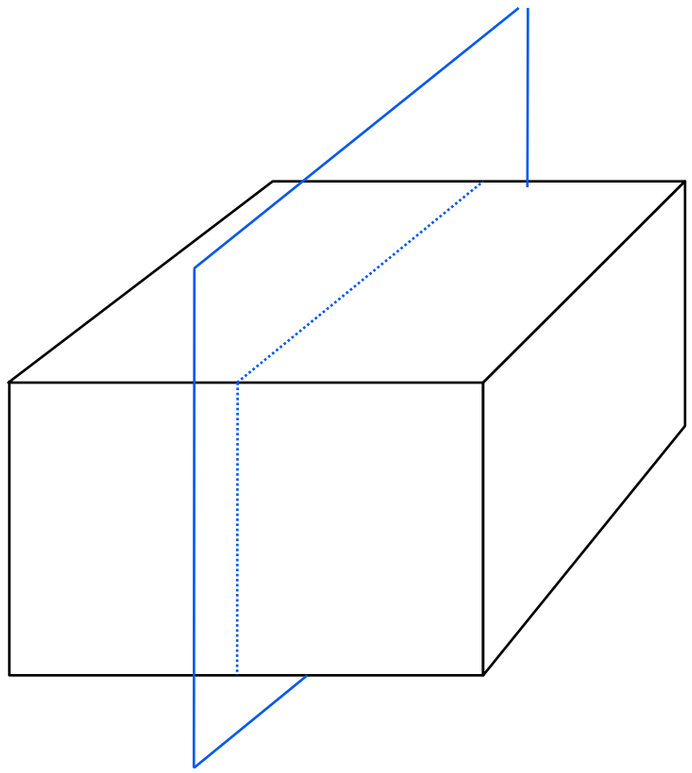
$$\Longrightarrow \quad \bar{x} = \frac{1}{n} \sum_k x_k$$

$$\bar{y} = \frac{1}{n} \sum_k y_k$$

$$\begin{pmatrix} a \\ b \end{pmatrix} = \text{Eigenvector of smallest eigenvalue of } A$$

What about 3D (or higher dimensions)?

- **Intuition**: Regular n x n x n mesh
  - Edges to 6 nearest neighbors
  - Partition using planes
- General graphs: Need notion of "**well-shaped**" like a mesh

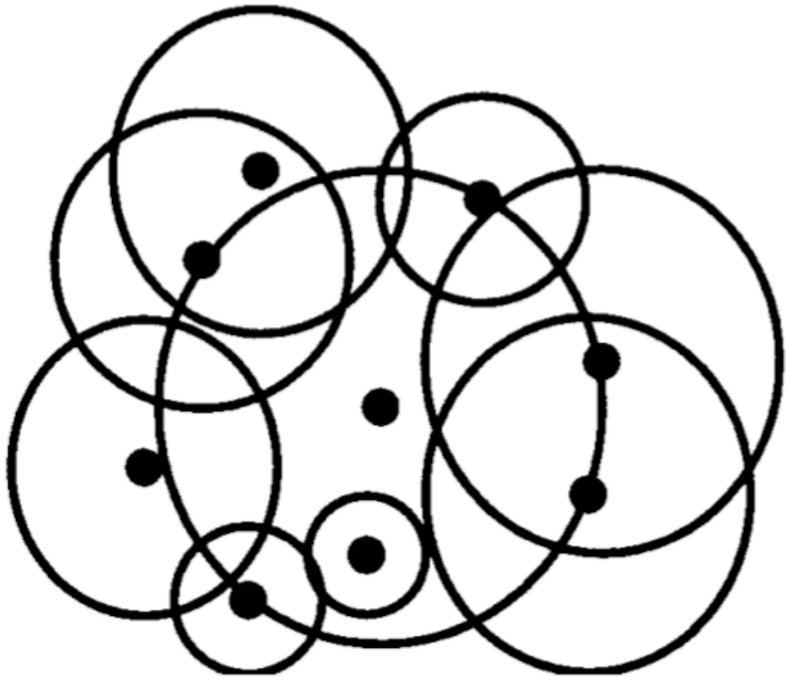


$$
\begin{aligned}
|V| &= n^3 \\
|V_s| &= n^2 \\
&= O(|V|^{\frac{2}{3}}) = O(|E|^{\frac{2}{3}})
\end{aligned}
$$

Random spheres

"Separators for sphere packings and nearest neighbor graphs."
Miller, Teng, Thurston, Vavasis (1997), *J. ACM*

- Definition: A **k-ply neighborhood system in d dimensions** = set $\{D_1, \ldots, D_n\}$ of closed disks in R^d such that no point in R^d is strictly interior to more than k disks

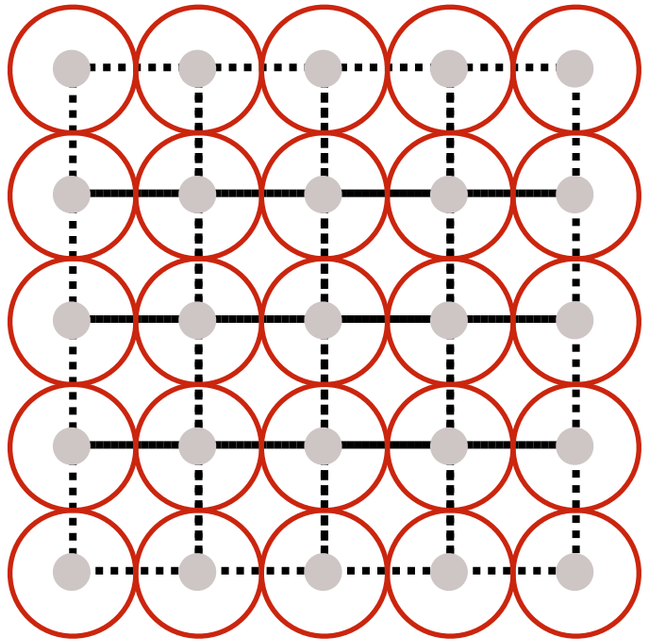


Example: 3-ply system

Random spheres

"Separators for sphere packings and nearest neighbor graphs."
Miller, Teng, Thurston, Vavasis (1997), *J. ACM*

- Definition: A **k-ply neighborhood system in d dimensions** = set $\{D_1, \ldots, D_n\}$ of closed disks in R^d such that no point in R^d is strictly interior to more than k disks
- Definition: An **(α,k) overlap graph**, for α >= 1 and a k-ply neighborhood:
  - Node = D_j
  - Edge j → i if expanding radius of smaller disk by >α causes two disks to overlap



Example: (1,1) overlap graph for a 2D mesh.

Random spheres (cont'd)

- *Theorem* (Miller, et al.): Let G = (V, E) be an (**α**, k) overlap graph in d dimensions, with n = |V|. Then there is a separator $\mathbf{V_s}$ s.t.:

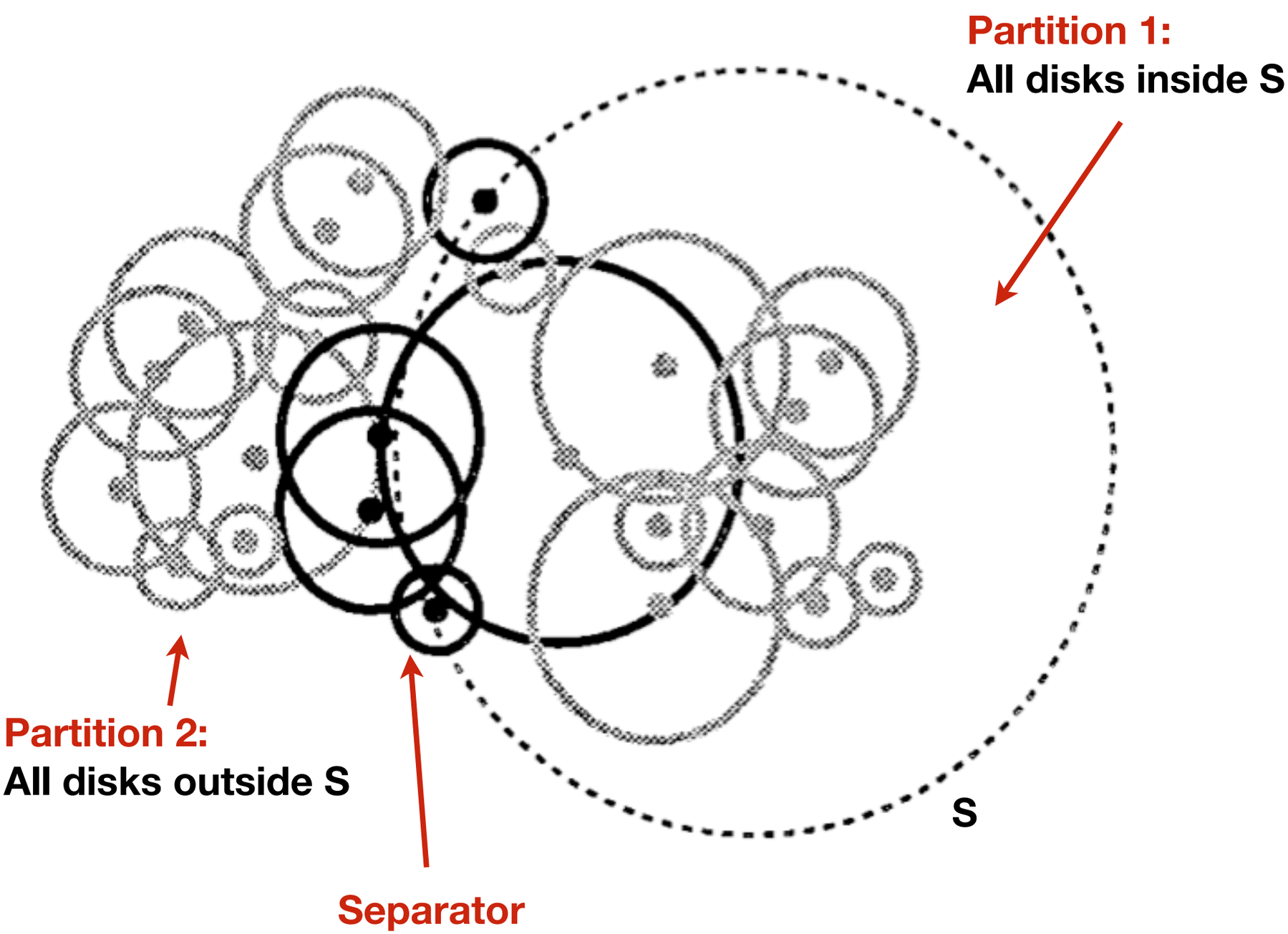
$$
\begin{aligned}
V &= V_1 \cup V_s \cup V_2 \\
|V_1|, |V_2| &< \frac{d+1}{d+2} \cdot n \\
|V_s| &= O\left(\alpha \cdot k^{\frac{1}{d}} \cdot n^{\frac{d-1}{d}}\right)
\end{aligned}
$$

- In 2D, same as Lipton & Tarjan

Random spheres: An algorithm

- Choose a sphere S in R^d
- Edges that S "cuts" form edge separator E_s
- Build V_s from E_s
- Choose S "randomly," s.t. satisfies theorem with high probability

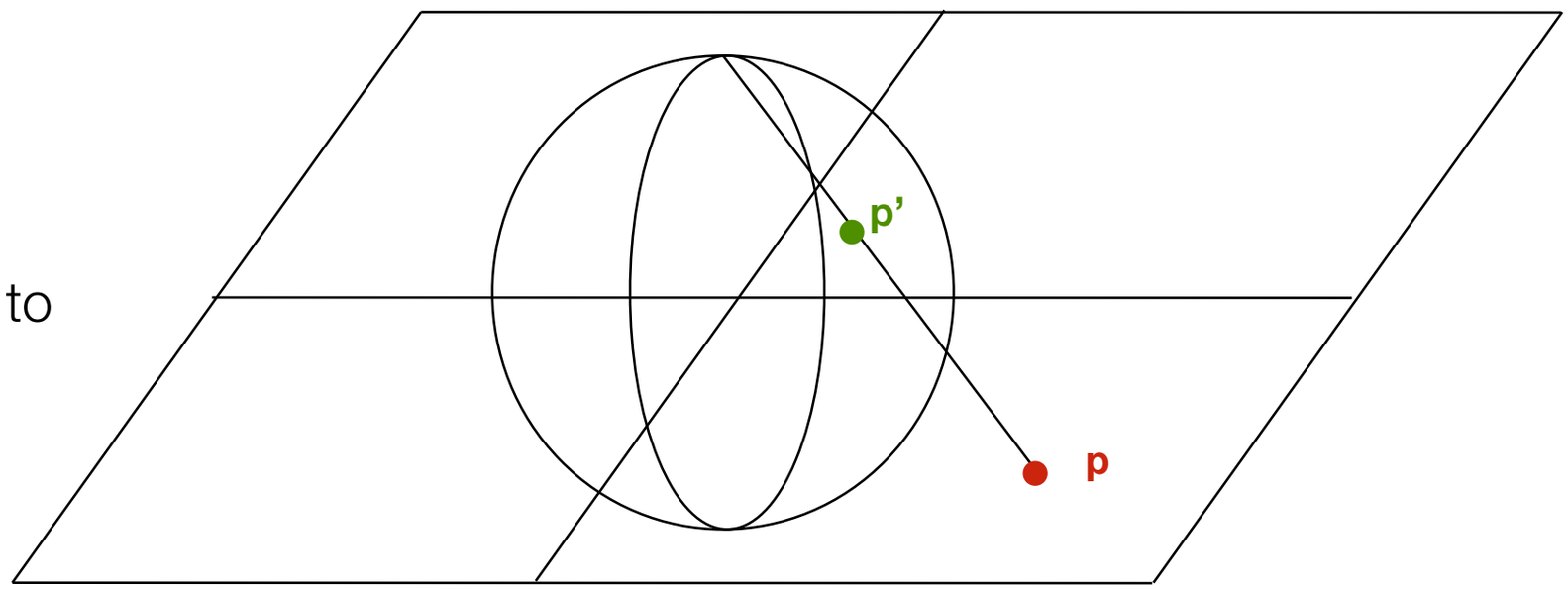
Random spheres algorithm



Choosing a random sphere: Stereographic projections

Given **p** in plane, project to **p'** on sphere.

1. Draw line from **p** to north pole.
2. **p'** = intersection.



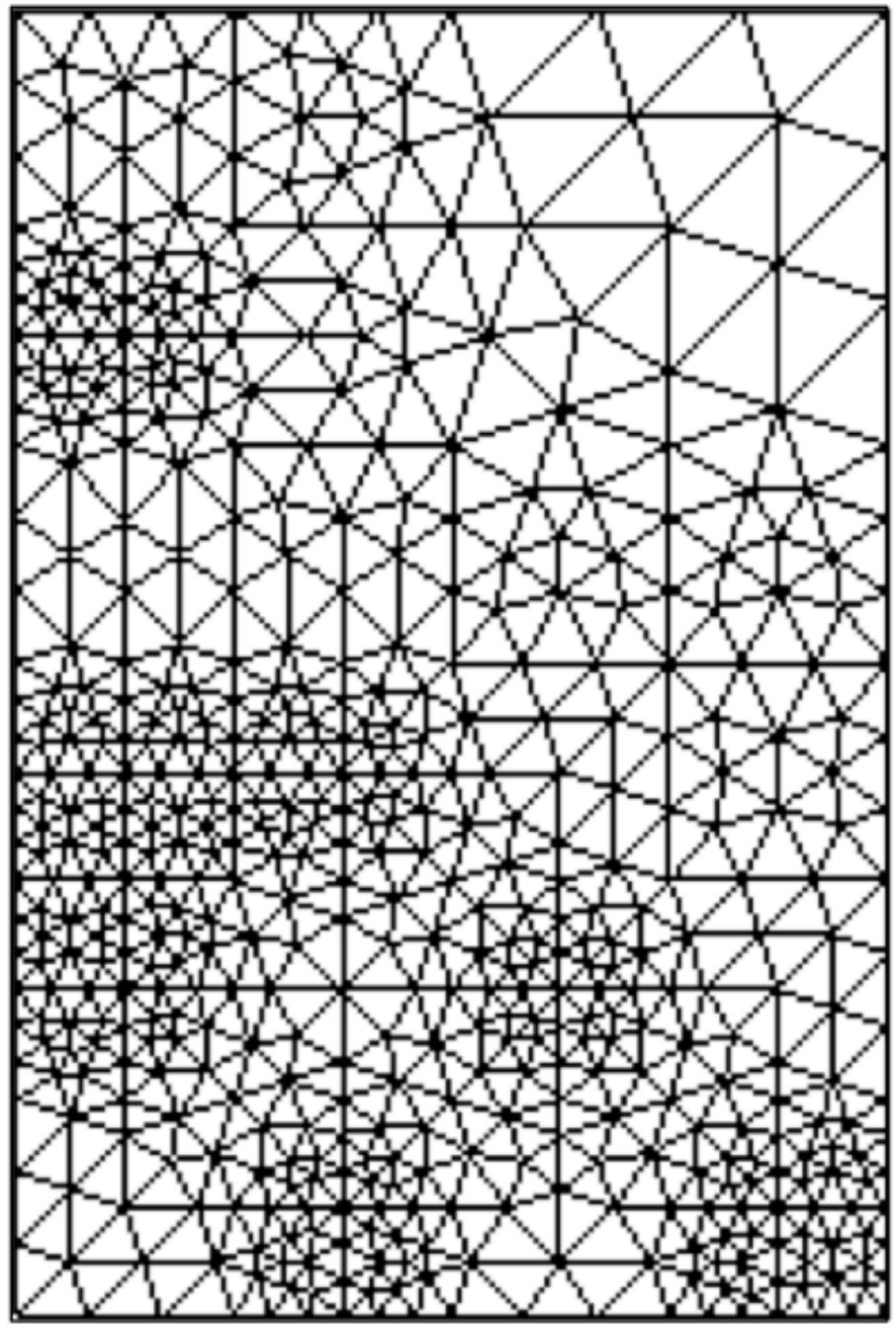
$$p = (x, y)$$

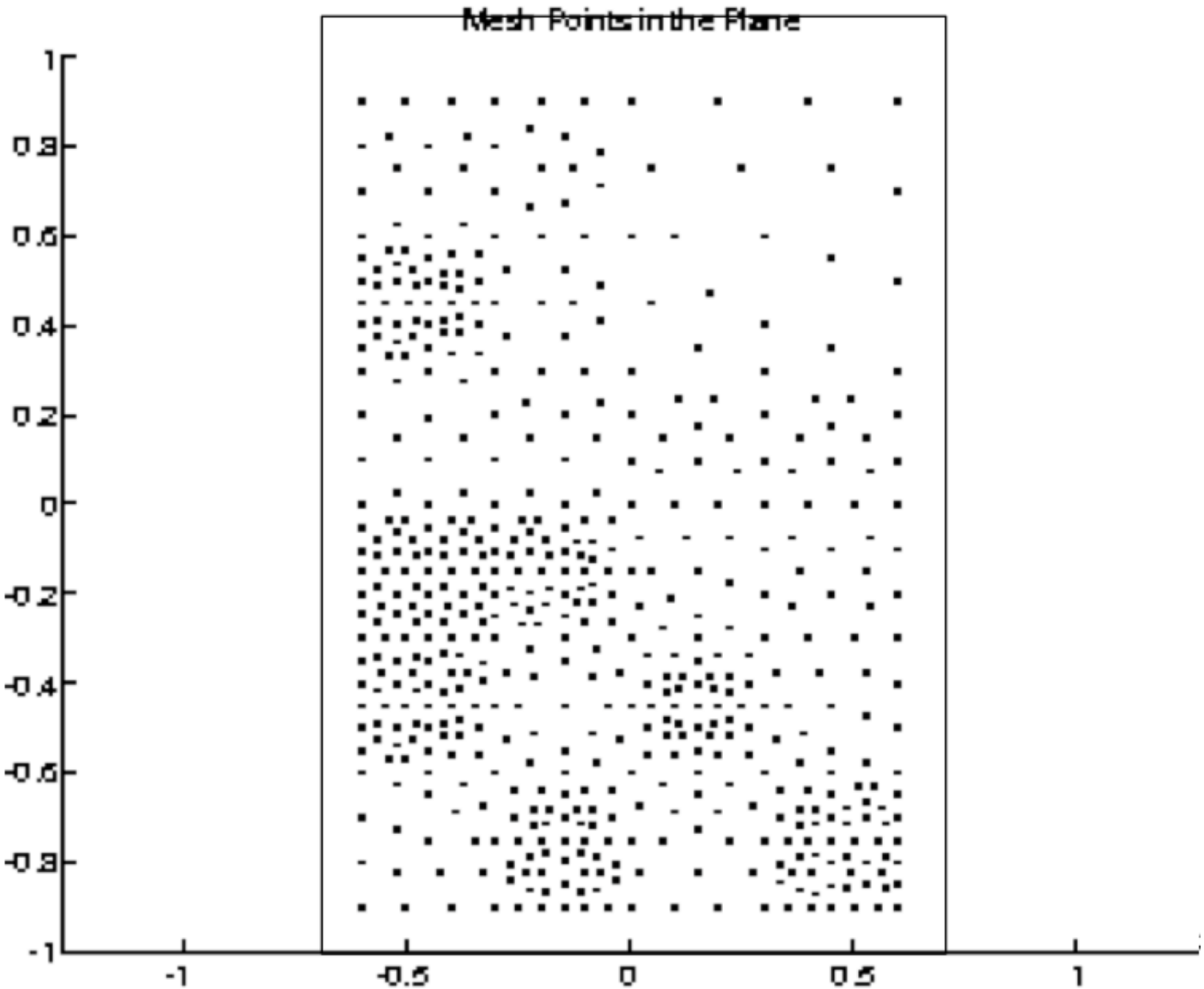
$$p' = \frac{(2x, 2y, x^2 + y^2 - 1)}{x^2 + y^2 + 1}$$

Random spheres separator algorithm (Miller, *et al.*)

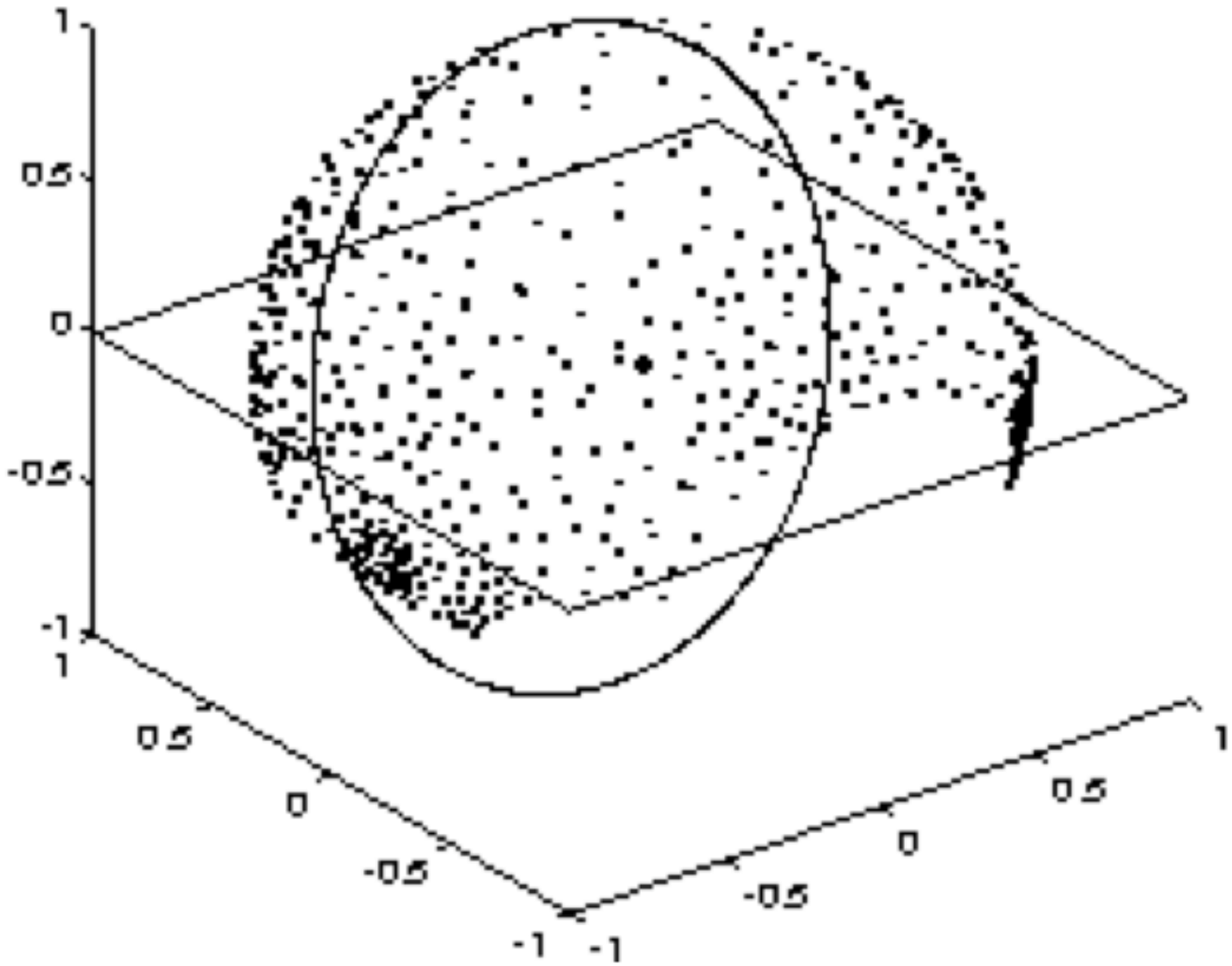
- Do stereographic projection from R^d to sphere S in R^{d+1}
- Find center-point of projected points
  - Center-point c: Any hyperplane through c divides points ~ evenly
  - There is a linear programming algorithm & cheaper heuristics
- Conformally map points on sphere
  - Rotate points around origin so center-point at (0, 0, …, 0, r) for some r
  - Dilate points: Unproject; multiply by sqrt((1-r)/(1+r)); project
  - Net effect: Maps center-point to origin & spreads points around S
- Pick a random plane through the origin; intersection of plane and sphere S = "circle"
- Unproject circle, yielding desired circle C in R^d
- Create $\mathbf{V_s}$: Node j in $\mathbf{V_s}$ if if $\alpha^* D_j$ intersections C

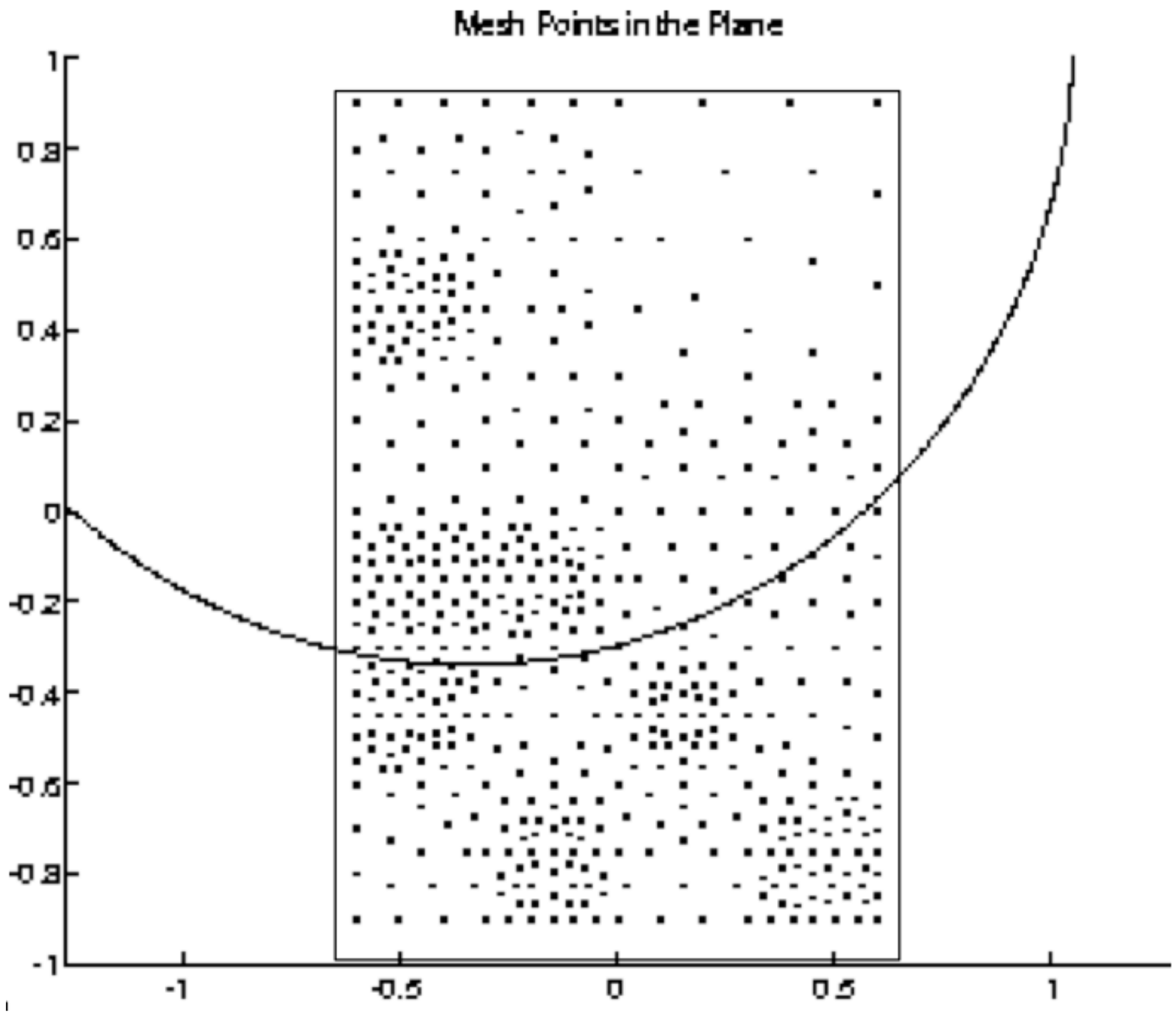
Finite Element Mesh



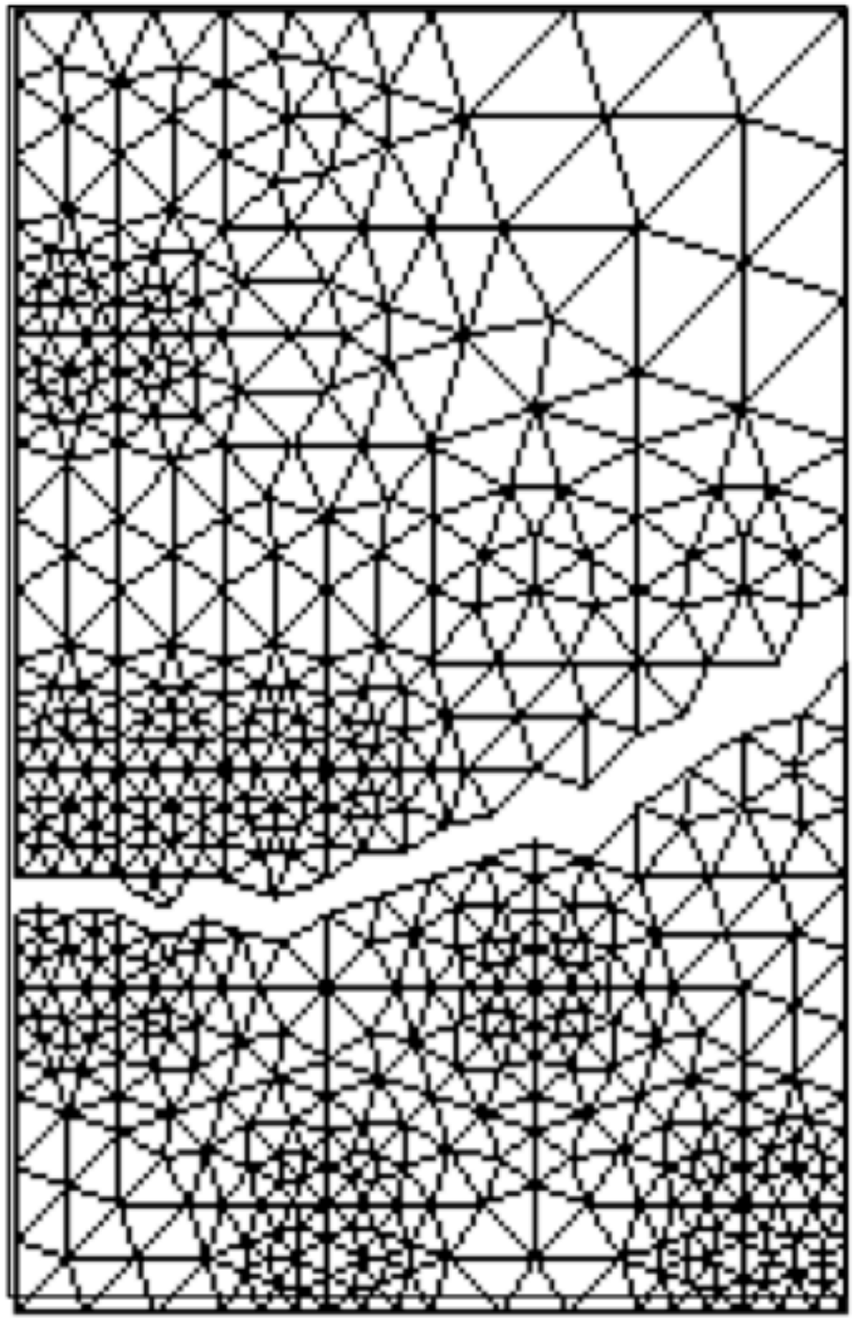

Mesh Points in the Plane
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1


Points Projected onto the Sphere
1
0.5
0
-0.5
-1
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1


1
0.5
0
-0.5
-1
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

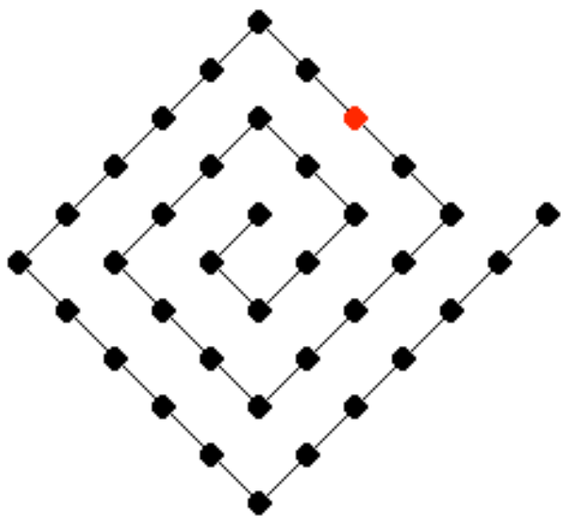

Mesh Points in the Plane
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

Partition of the Original Mesh



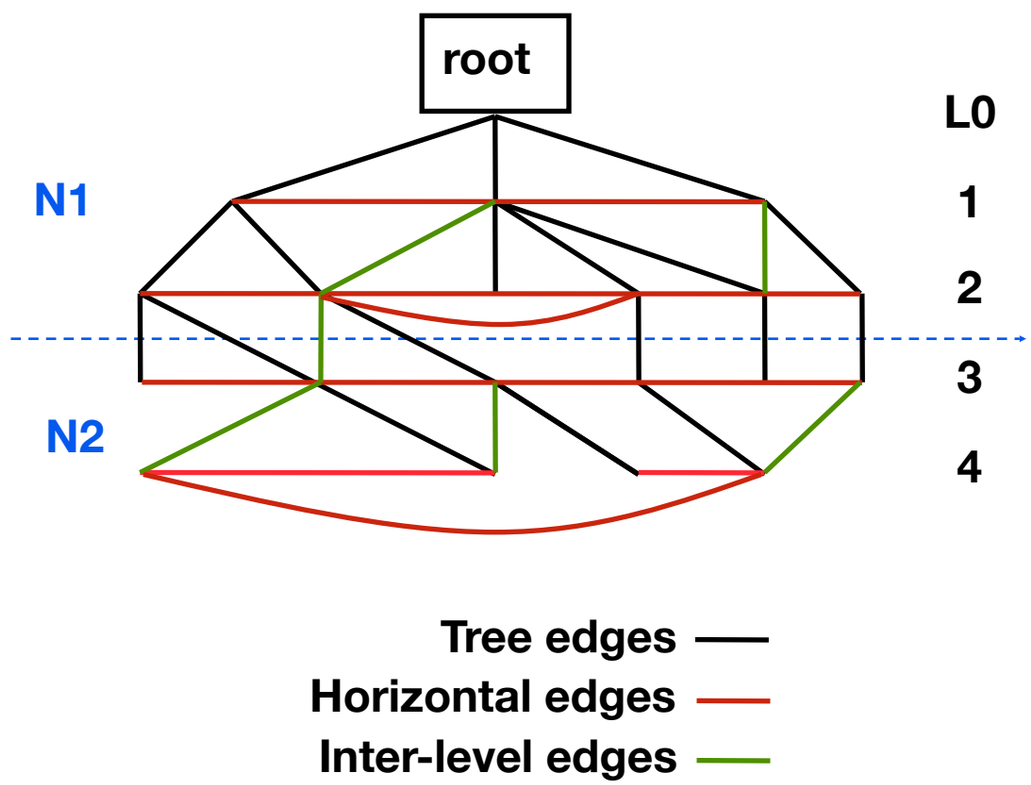
Summary:
Nodal coordinate-based algorithms

- Other variations exist
- Algorithms are efficient: O(points)
- Implicitly assume nearest neighbor connectivity: Ignores edges!
  - Common for graphs from physical models
  - Good “initial guess” for other algorithms
  - Poor performance on non-spatial graphs

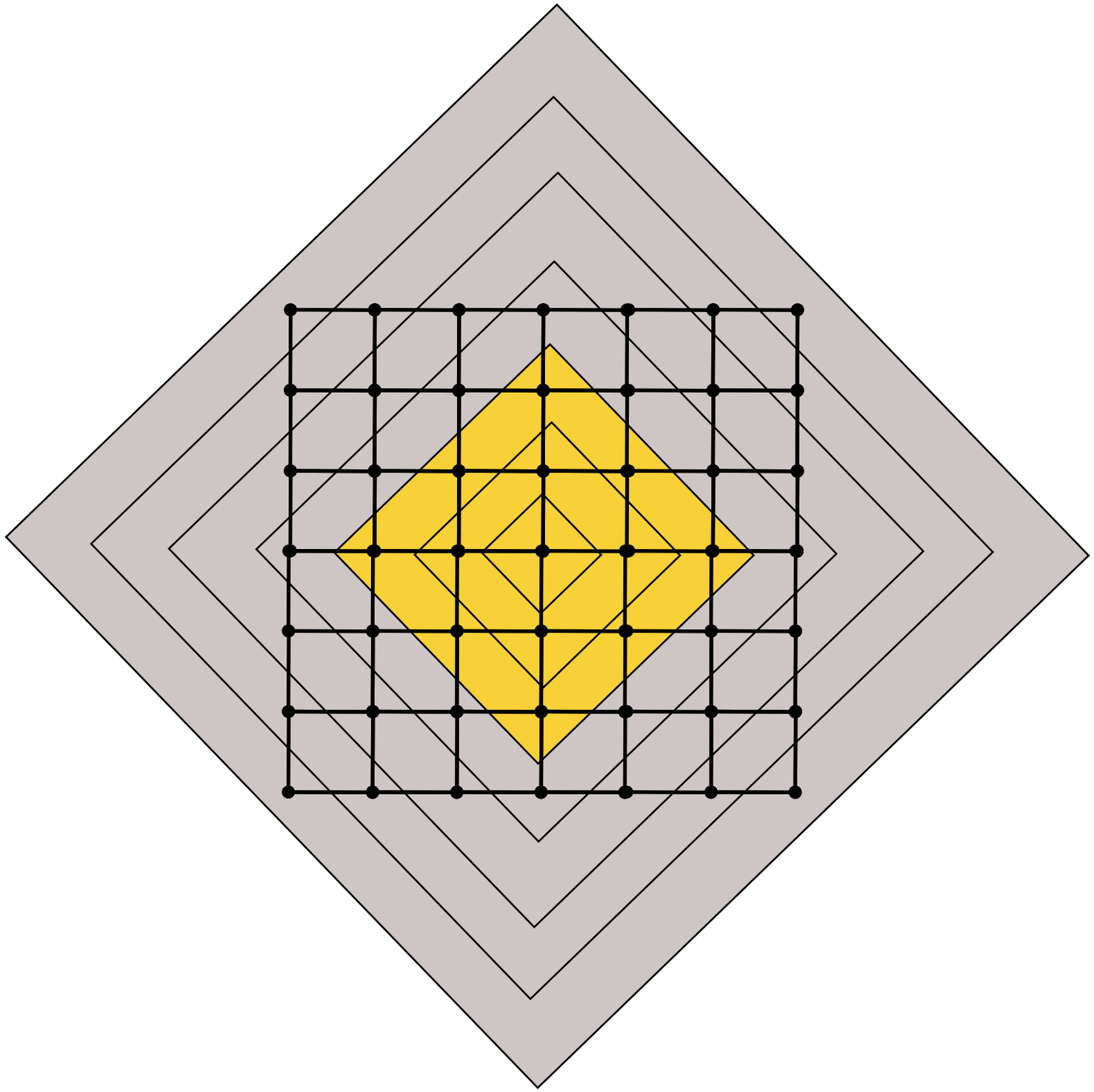


Partitioning without nodal coordinates

A coordinate-free algorithm: Breadth-first search



- Choose root r and run BFS, which produces:
  - Subgraph T of G (same nodes, subset of edges)
  - T rooted at r
  - Level of each node = distance from r



Kernighan/Lin (1970): Iteratively refine

- Given edge-weighted graph and partitioning:

$$
\begin{aligned}
G &= (V, E, W_E) \\
V &= A \cup B, \qquad |A| = |B| \\
E_s &= \{(u,v) \in E : u \in A, v \in B\} \\
T \equiv \mathrm{cost}(A,B) &\equiv \sum_{e \in E_s} w(e)
\end{aligned}
$$

- Find equal-sized subsets X, Y of A, B s.t. swapping reduces cost
- **Need ability to quickly compute cost for many possible X, Y**

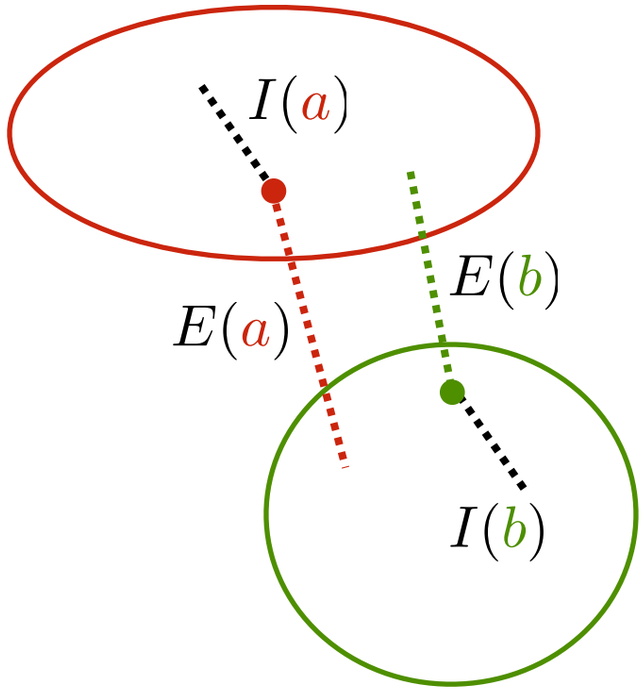
K-L refinement: Definitions

- **Definition**: “External” and “internal” costs of a ∈ A, and their difference; similarly for B:

$$E(a) \equiv \sum_{(a,b)\in E_s} w(a,b)$$

$$I(a) \equiv \sum_{(a,a')\in A} w(a,a')$$

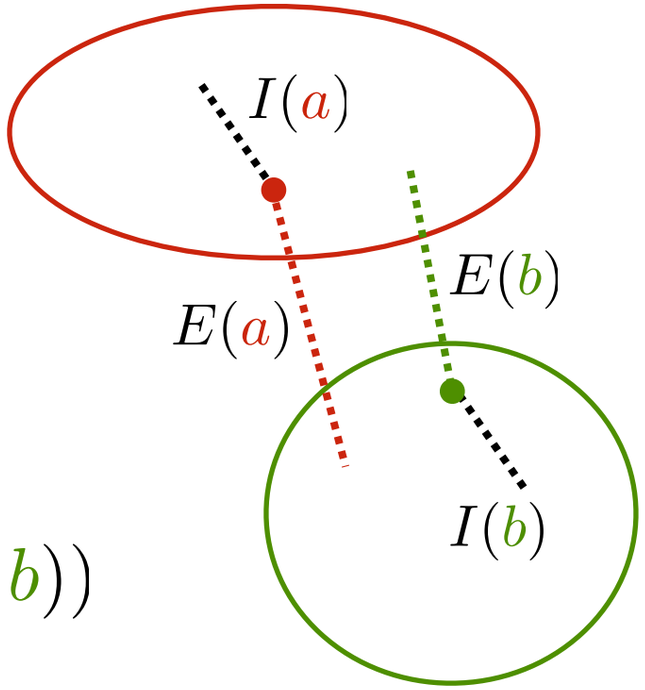
$$D(a) \equiv E(a) - I(a)$$



Consider swapping two nodes

- Swap X = {**a**} and Y = {**b**}:

$$
\begin{aligned}
A' &= (A - a) \cup b \\
B' &= (B - b) \cup a
\end{aligned}
$$



- Cost changes:

$$
\begin{aligned}
T' &= T - (D(a) + D(b) - 2w(a, b)) \\
&\equiv T - \text{gain}(a, b)
\end{aligned}
$$

```
KL-refinement-algorithm (A, B):
   Compute  T = cost(A,B) for initial A, B                    … cost = O(|V|²)
   Repeat
        … One pass greedily computes |V|/2 possible X,Y to swap, picks best
        Compute costs D(v) for all v in V                     … cost = O(|V|²)
        Unmark all nodes in V                                 … cost = O(|V|)
        While there are unmarked nodes                        … |V|/2 iterations
            Find an unmarked pair (a,b) maximizing gain(a,b)      … cost = O(|V|²)
            Mark a and b (but do not swap them)                   … cost = O(1)
            Update D(v) for all unmarked v,
                  as though a and b had been swapped              … cost = O(|V|)
         Endwhile
            … At this point we have computed a sequence of pairs
            …  (a1,b1), … , (ak,bk)   and gains gain(1),…., gain(k)
            … where k = |V|/2, numbered in the order in which we marked them
        Pick m maximizing Gain = Σ_{k=1 to m}  gain(k)        … cost = O(|V|)
            … Gain is reduction in cost from swapping (a1,b1) through (am,bm)
        If Gain > 0 then   … it is worth swapping
            Update newA = A - { a1,…,am } U { b1,…,bm }       … cost = O(|V|)
            Update newB = B - { b1,…,bm } U { a1,…,am }       … cost = O(|V|)
            Update T = T - Gain                               … cost = O(1)
        endif
   Until Gain <= 0
```

Comments

- Expensive: O(n3)
  - Complicated, but cheaper, alternative exists: Fiduccia and Mattheyses (1982), “A linear-time heuristic for improving network partitions.” GE Tech Report.
- Some gain(k) may be negative, but can still get positive final gain
  - Escape local minima
- Outer loop iterations?
  - On very small graphs, |V| <= 360, KL show convergence after 2-4 sweeps
  - For random graphs, probability of convergence in 1 sweep goes down like $2^{-|V|/30}$

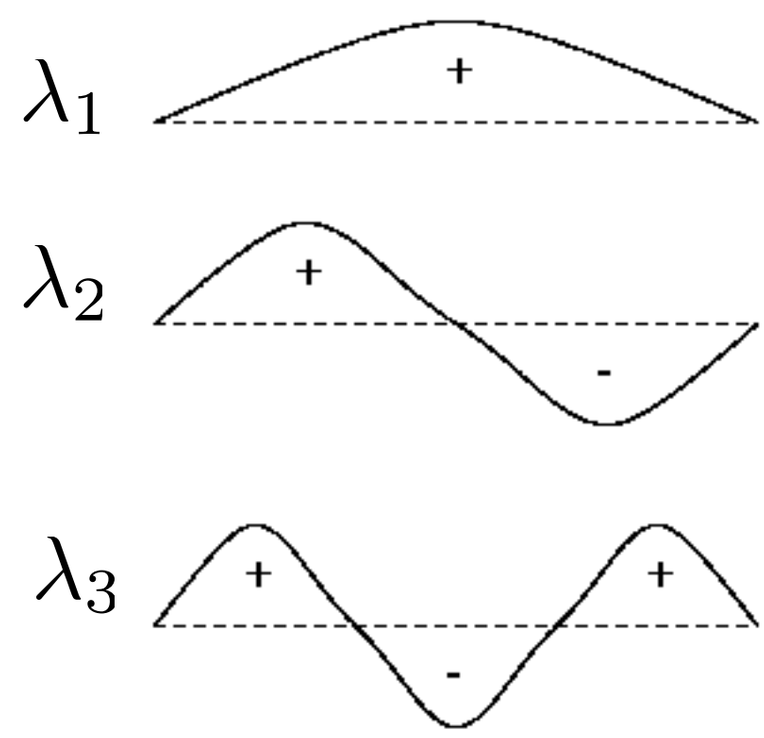
Spectral bisection

- Theory by Fiedler (1973): “Algebraic connectivity of graphs.” *Czech. Math. J.*
- Popularized by Pothen, Simon, Liu (1990): “Partitioning Sparse Matrices with Eigenvectors of Graphs.” *SIAM J. Mat. Anal. Appl.*
- Motivation: Vibrating string
- Computation: Compute eigenvector
  - To optimize sparse matrix-vector multiply, partition graph
  - To graph partition, find eigenvector of matrix associated with graph
  - To find eigenvector, do sparse matrix-vector multiply

A physical intuition: Vibrating strings

- G = 1D mesh of nodes connected by vibrating strings
- String has vibrational modes, or harmonics
- Label nodes by “+” or “-” to partition into N_- and N_+
- Same idea for other graphs, *e.g.*, planar graph → trampoline

Vibrational modes

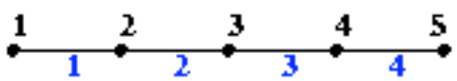


Definitions

- Definition: **Incidence matrix In(G)** = |V| x |E| matrix, s.t. edge e = (i, j) →
  - In(G)[i, e] = +1
  - In(G)[j, e] = -1
  - Ambiguous (multiply by -1), but doesn't matter
- Definition: **Laplacian matrix L(G)** = |V| x |V| matrix, s.t. L(G)[i, j] = …
  - degree of node i, if i == j;
  - -1, if there is an edge (i, j) and i ≠ j
  - 0, otherwise

Examples of incidence and Laplacian matrices

Graph G

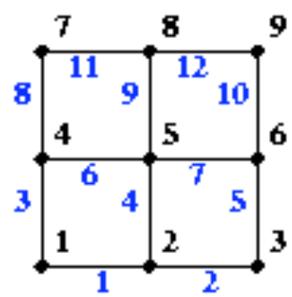


Incidence Matrix In(G)

$$\begin{array}{c|c} & \begin{matrix} 1 & 2 & 3 & 4 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} -1 & & & \\ 1 & -1 & & \\ & 1 & -1 & \\ & & 1 & -1 \\ & & & 1 \end{bmatrix} \end{array}$$

Laplacian Matrix L(G)

$$\begin{array}{c|c} & \begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} & \begin{bmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & 2 & -1 & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{bmatrix} \end{array}$$



$$\begin{array}{c|c} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{matrix} & \begin{bmatrix} -1 & & 1 & & & & & & & & & \\ 1 & -1 & & 1 & & & & & & & & \\ & 1 & & & 1 & & & & & & & \\ & & -1 & & & -1 & & 1 & & & & \\ & & & -1 & & 1 & -1 & & 1 & & & \\ & & & & -1 & & 1 & & & 1 & & \\ & & & & & & & -1 & & & -1 & \\ & & & & & & & & -1 & & 1 & -1 \\ & & & & & & & & & -1 & & 1 \end{bmatrix} \end{array}$$

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

Nodes numbered in black

Edges numbered in blue

Theorem: Properties of L(G)

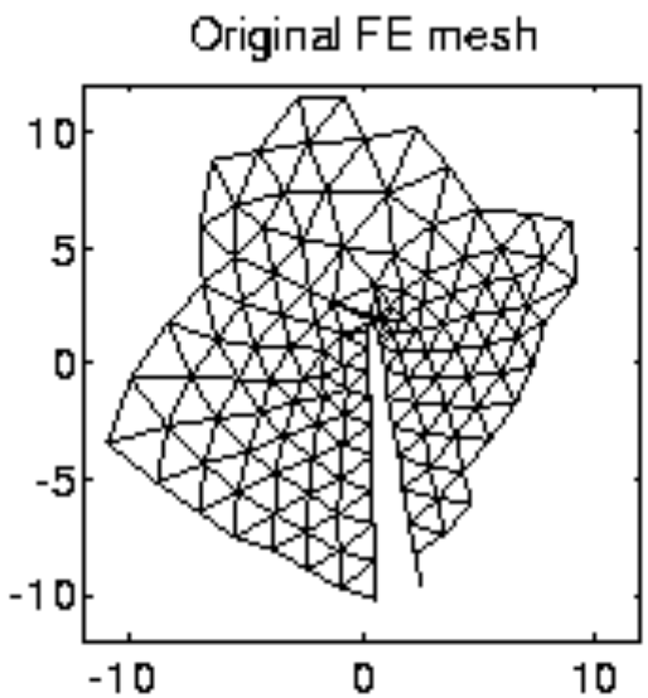
- L(G) is symmetric ⇒ eigenvalues are real, eigenvectors real & orthogonal
- $In(G) * In(G)^T = L(G)$
- Eigenvalues are non-negative, *i.e.*, $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$
- Number of connected components of G = number of 0 eigenvalues
- Definition: $\lambda_2(G)$ = **algebraic connectivity** of G
  - Magnitude measures connectivity
  - Non-zero if and only if G is connected

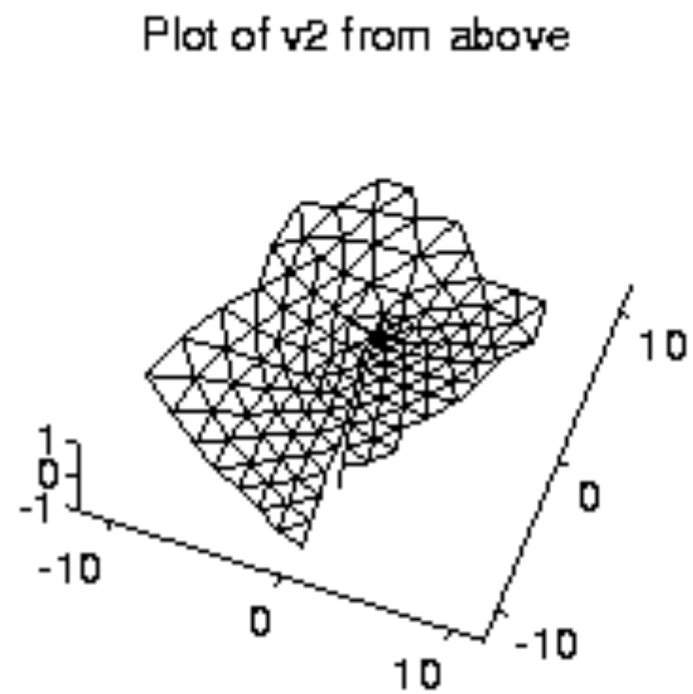
Spectral bisection algorithm

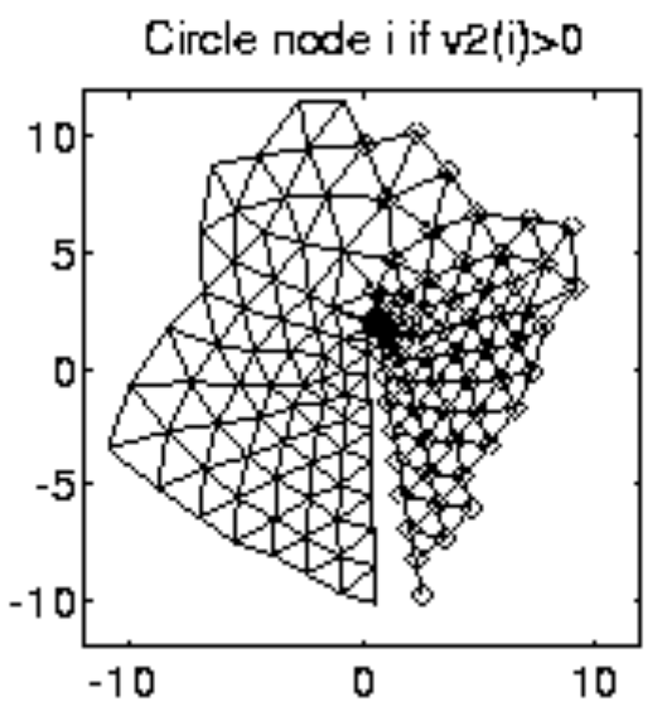
- Algorithm:
  - Compute eigenpair (λ_2, q_2)
  - For each node v in G:
    - if q2(v) < 0, place in partition N_-
    - else, place in partition N_+
- Why?

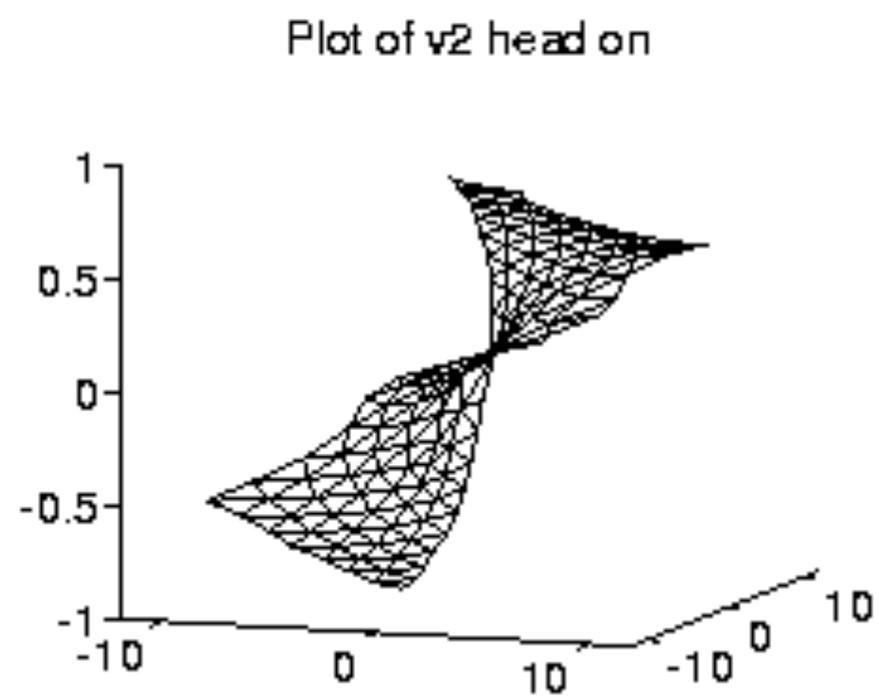
Why the spectral bisection is "reasonable": Fiedler's theorems

- Theorem 1:
  - G connected ⇒ N_- connected
  - All $q_2(v) \neq 0 \Rightarrow N_+$ connected
- Theorem 2: Let G_1 be "less-connected" than G, *i.e.*, has same nodes & subset of edges. Then $\lambda_2(G_1) \leq \lambda_2(G)$
- Theorem 3: G is connected and $V = V1 \cup V2$, with $|V1| \sim |V2| \sim |V|/2$
  - $\Rightarrow |E_s| >= 1/4 * |V| * \lambda_2(G)$


Original FE mesh
10
5
0
-5
-10
-10
0
10


Plot of v2 from above
1
0
-1
-10
0
10
10
0
-10


Circle node i if v2(i)>0
10
5
0
-5
-10
-10
0
10


Plot of v2 head on
1
0.5
0
-0.5
-1
-10
0
10
-10
0
10

Spectral bisection: Key ideas

- Laplacian matrix represents graph connectivity
- Second eigenvector gives bisection
- Implement via Lanczos algorithm
  - Requires matrix-vector multiply, which is why we partitioned…
  - Do first few slowly, accelerate the rest

Administrivia

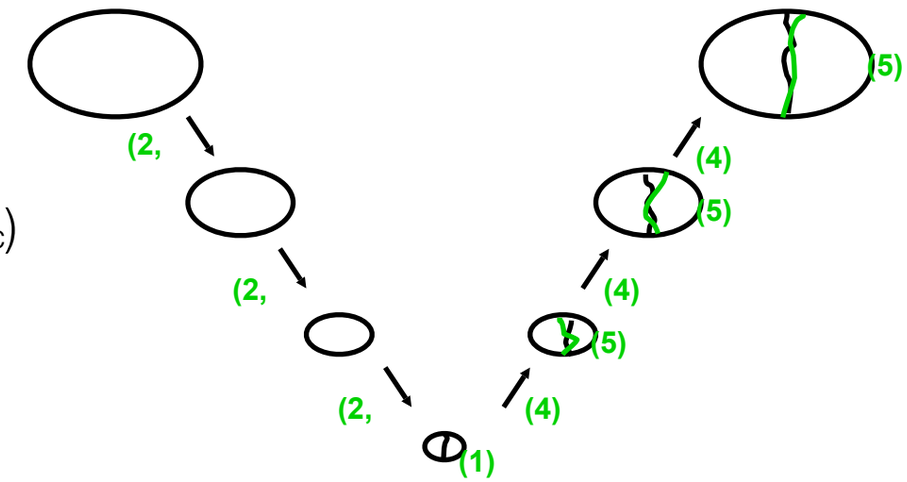
Final stretch…

- Today is last class (woo hoo!)
- BUT: 4/24
  - Attend HPC Day (Klaus atrium / 1116E)
  - Go to SIAM Data Mining Meeting
- Final project presentations: Mon 4/28
  - Room and time TBD
  - Let me know about conflicts
  - Everyone must attend, even if you are giving a poster at HPC Day

Multilevel partitioning

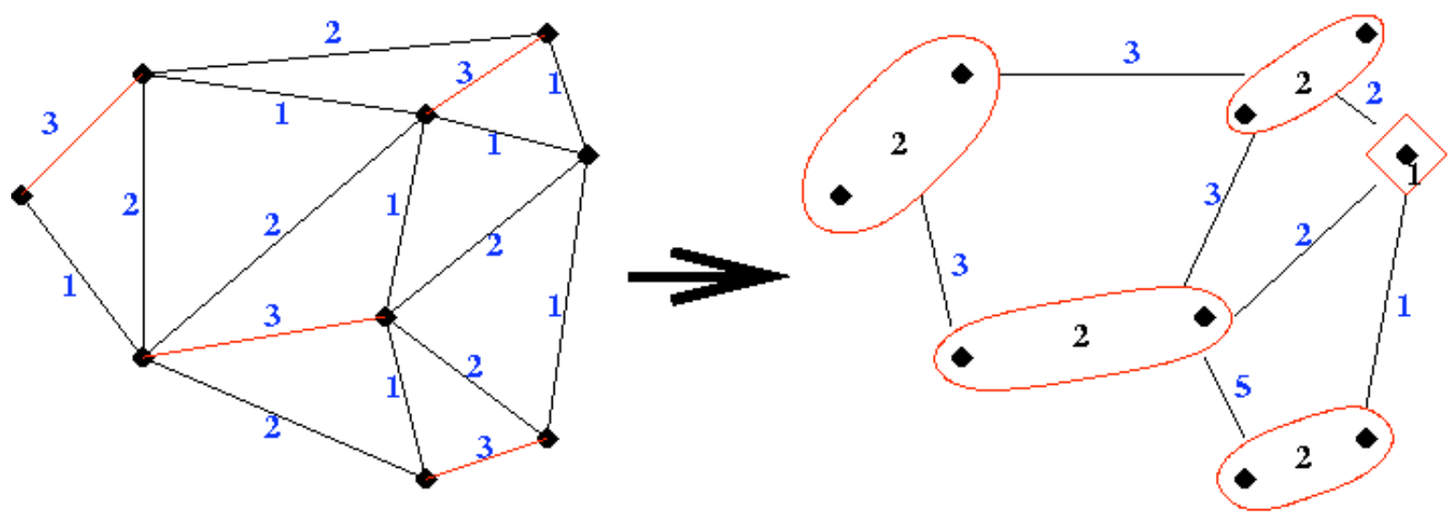
Familiar idea: Multilevel partitioning "V-cycle"

- $(V^+, V^-) \leftarrow$ **Multilevel_Partition** $(G = (V, E))$
  - If $|V|$ is "small", partition directly
  - Else:
    - **Coarsen** $G \rightarrow G_c = (V_c, E_c)$
    - $(V_c^+, V_c^-) \leftarrow$ **Multilevel_Partition** (V_c, E_c)
    - **Expand** $(V_c^+, V_c^-) \rightarrow (V^+, V^-)$
    - **Improve** (V^+, V^-)
    - Return (V^+, V^-)

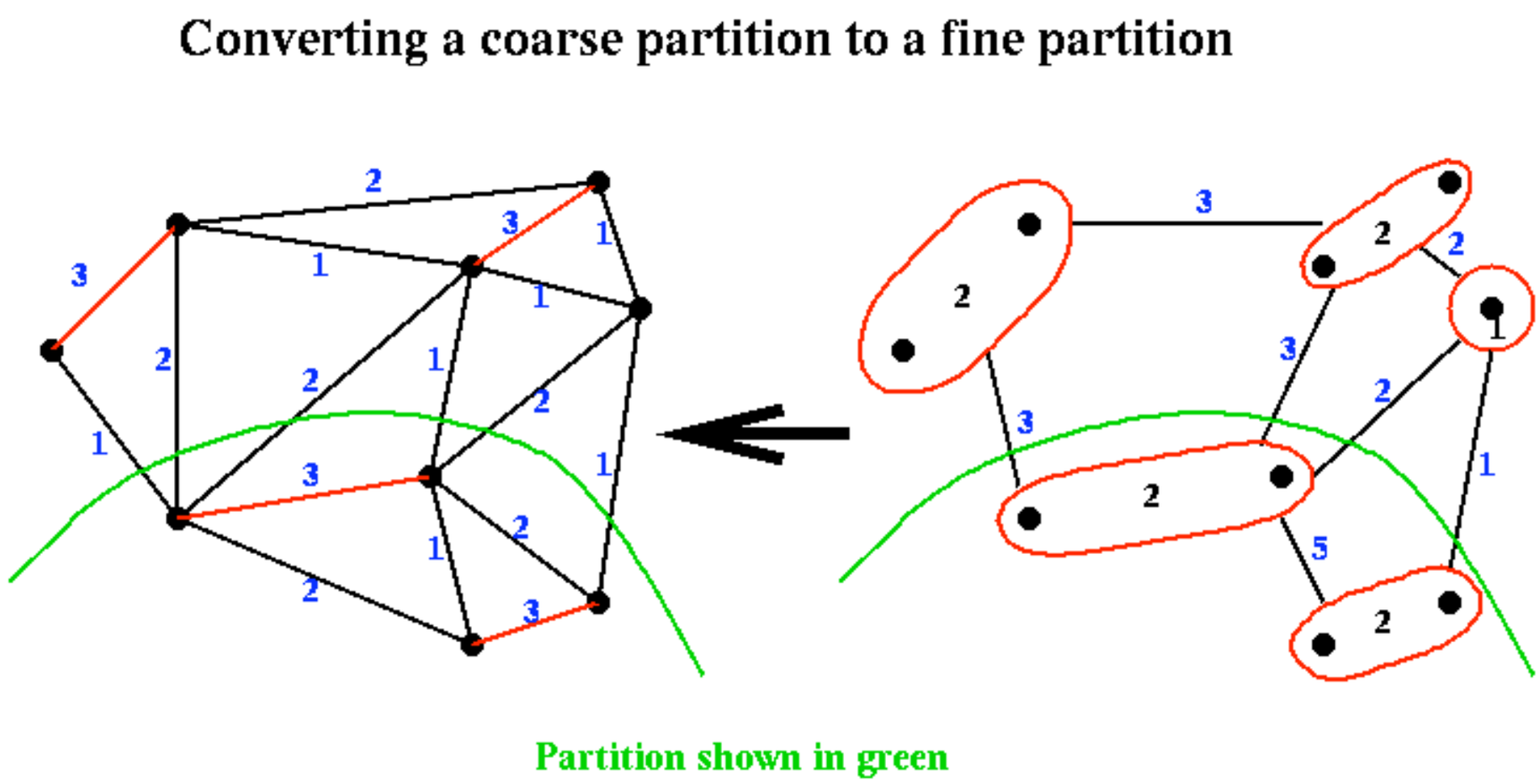


Algorithm 1: Multilevel Kernighan-Lin

- Coarsen and expand using maximal matchings
  - Definition: Matching = subset of edges s.t. no two edges share an endpoint
  - Use greedy algorithm
- Improve partitions using KL-refinement

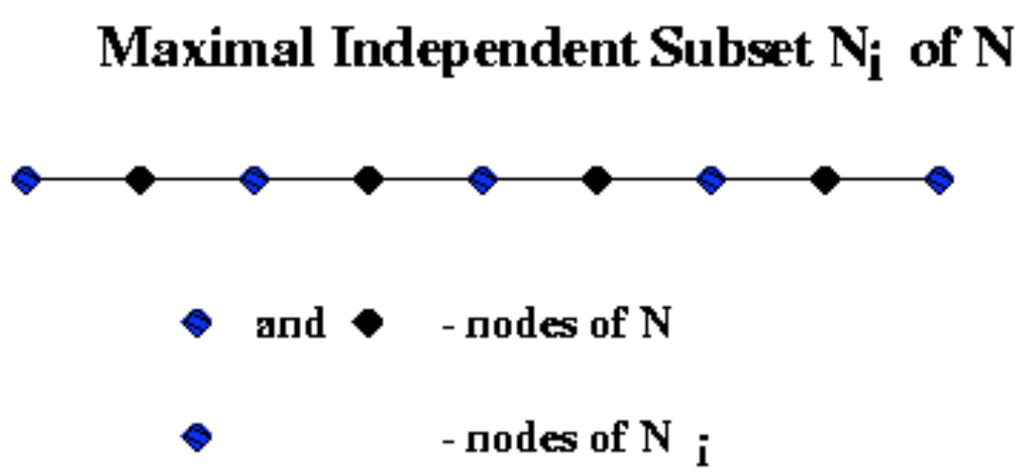


Expanding a partition from coarse-to-fine graph



Multilevel spectral bisection

- Coarsen and expand using maximal independent sets
  - Definition: Independent set = subset of unconnected nodes
  - Use greedy algorithm to compute
- Improve partition using Rayleigh-Quotient iteration



Multilevel software

- Multilevel Kernighan/Lin: METIS and ParMETIS
- Multilevel spectral bisection:
  - Barnard & Simon
  - Chaco (Sandia)
- Hybrids possible
- Comparisons: Not up to date, but what was known…
  - No one method “best”, but multilevel KL is fast
  - Spectral better for some apps, *e.g.*, normalized cuts in image segmentation

“In conclusion…”

Ideas apply broadly

- Physical sciences, *e.g.*,
  - Plasmas
  - Molecular dynamics
  - Electron-beam lithography device simulation
  - Fluid dynamics
- “Generalized” n-body problems: Talk to your classmate, Ryan Riegel

Backup slides