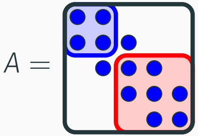
CS 5220: Graph Partitioning

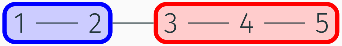
David Bindel

2017-11-07

Reminder: Sparsity and partitioning



Matrix



Graph

Want to partition sparse graphs so that

- Subgraphs are same size (load balance)
- Cut size is minimal (minimize communication)

Uses: parallel sparse matvec, nested dissection solves, …

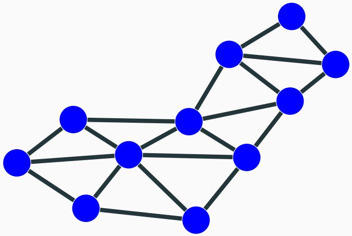
A common theme

Common idea: partition static data (or networked things):

- Physical network design (telephone layout, VLSI layout)
- Sparse matvec
- Preconditioners for PDE solvers
- Sparse Gaussian elimination
- Data clustering
- Image segmentation

Goal: Keep chunks big, minimize the “surface area” between

Graph partitioning

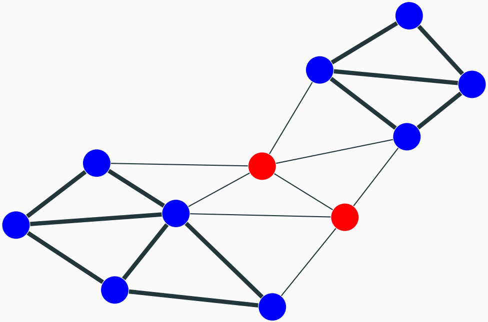


Given: $G = (V, E)$, possibly with weights and coordinates.
We want to partition G into k pieces such that

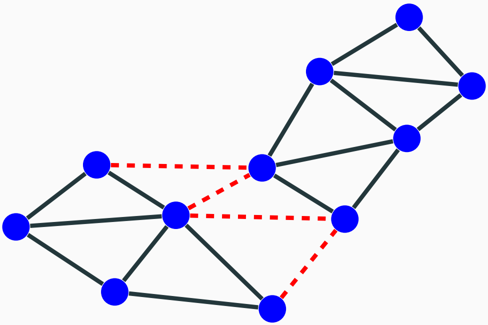
- Node weights are balanced across partitions.
- Weight of cut edges is minimized.

Important special case: $k = 2$.

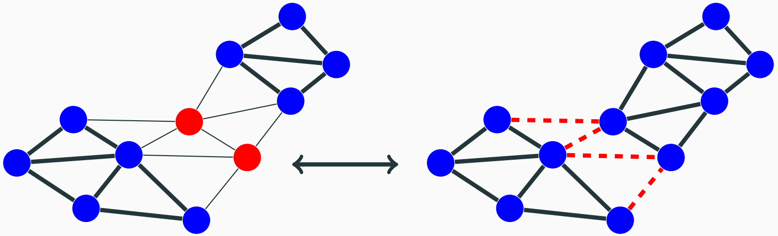
Graph partitioning: Vertex separator



Graph partitioning: Edge separator



Node to edge and back again



Can convert between node and edge separators

- Node to edge: cut all edges from separator to one side
- Edge to node: remove nodes on one side of cut edges

Fine if graph is degree bounded (e.g. near-neighbor meshes). Optimal vertex/edge separators very different for social networks!

How many partitionings are there? If n is even,

$$\binom{n}{n/2} = \frac{n!}{((n/2)!)^2} \approx 2^n \sqrt{2/(\pi n)}.$$

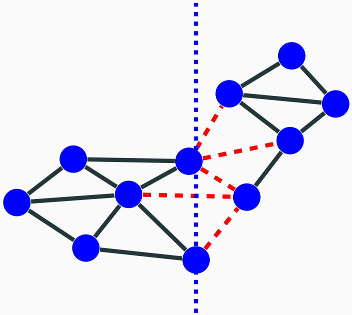
Finding the optimal one is NP-complete.

We need heuristics!

Partitioning with coordinates

- Lots of partitioning problems from "nice" meshes
  - Planar meshes (maybe with regularity condition)
  - k-ply meshes (works for $d > 2$)
  - Nice enough $\implies$ partition with $O(n^{1-1/d})$ edge cuts (Tarjan, Lipton; Miller, Teng, Thurston, Vavasis)
  - Edges link nearby vertices
- Get useful information from vertex density
- Ignore edges (but can use them in later refinement)

Recursive coordinate bisection



Idea: Cut with hyperplane parallel to a coordinate axis.

- Pro: Fast and simple
- Con: Not always great quality

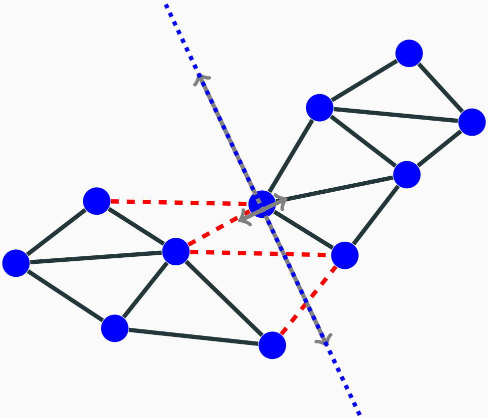
Inertial bisection

Idea: Optimize cutting hyperplane based on vertex density

$$
\begin{aligned}
\bar{\mathsf{x}} &= \frac{1}{n}\sum_{i=1}^{n} \mathsf{x}_i \\
\bar{\mathsf{r}}_i &= \mathsf{x}_i - \bar{\mathsf{x}} \\
\mathsf{I} &= \sum_{i=1}^{n} \left[\|\mathsf{r}_i\|^2 I - \mathsf{r}_i \mathsf{r}_i^T \right]
\end{aligned}
$$

Let $(\lambda_n, \mathbf{n})$ be the minimal eigenpair for the inertia tensor $\mathbf{I}$, and choose the hyperplane through $\bar{\mathsf{x}}$ with normal $\mathbf{n}$.

Inertial bisection



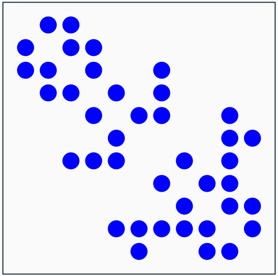
- Pro: Still simple, more flexible than coordinate planes
- Con: Still restricted to hyperplanes

Random circles (Gilbert, Miller, Teng)

- Stereographic projection
- Find centerpoint (any plane is an even partition)
  In practice, use an approximation.
- Conformally map sphere, moving centerpoint to origin
- Choose great circle (at random)
- Undo stereographic projection
- Convert circle to separator

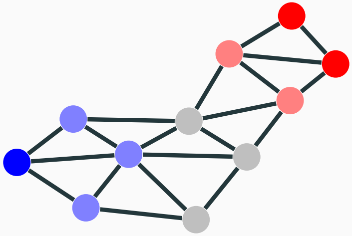
May choose best of several random great circles.

Coordinate-free methods



- Don't always have natural coordinates
  - Example: the web graph
  - Can sometimes add coordinates (metric embedding)
- So use edge information for geometry!

Breadth-first search



- Pick a start vertex v_0
    - Might start from several different vertices
- Use BFS to label nodes by distance from v_0
    - We've seen this before – remember RCM?
    - Could use a different order – minimize edge cuts locally (Karypis, Kumar)
- Partition by distance from v_0

Spectral partitioning

Label vertex i with $x_i = \pm 1$. We want to minimize

$$\text{edges cut} = \frac{1}{4} \sum_{(i,j) \in E} (x_i - x_j)^2$$

subject to the even partition requirement

$$\sum_i x_i = 0.$$

But this is NP hard, so we need a trick.

Spectral partitioning

Write

$$\text{edges cut} = \frac{1}{4} \sum_{(i,j)\in E} (x_i - x_j)^2 = \frac{1}{4}\|Cx\|^2 = \frac{1}{4}x^T L x$$

where C is the incidence matrix and $L = C^T C$ is the graph Laplacian:

$$C_{ij} = \begin{cases} 1, & e_j = (i,k) \\ -1, & e_j = (k,i) \\ 0, & \text{otherwise,} \end{cases} \qquad L_{ij} = \begin{cases} d(i), & i = j \\ -1, & i \neq j, (i,j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

Note that $Ce = 0$ (so $Le = 0$), $e = (1, 1, 1, \ldots, 1)^T$.

Spectral partitioning

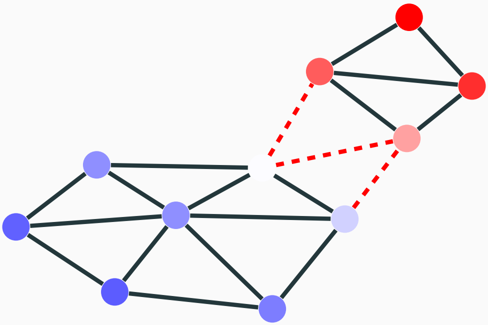
Now consider the *relaxed* problem with $x \in \mathbb{R}^n$:

$$\text{minimize } x^T L x \text{ s.t. } x^T e = 0 \text{ and } x^T x = 1.$$

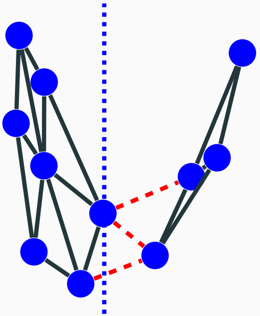
Equivalent to finding the second-smallest eigenvalue λ_2 and corresponding eigenvector x, also called the *Fiedler vector*. Partition according to sign of x_i.

How to approximate x? Use a Krylov subspace method (Lanczos)! Expensive, but gives high-quality partitions.

Spectral partitioning



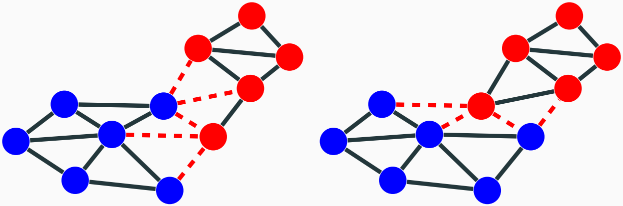
Spectral coordinates



Alternate view: define a coordinate system with the first d non-trivial Laplacian eigenvectors.

- Spectral partitioning = bisection in spectral coordinates
- Can cluster in other ways as well (e.g. k-means)

Refinement by swapping



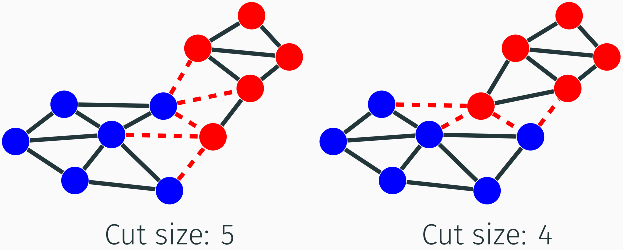
Cut size: 5 Cut size: 4

Gain from swapping (a, b) is $D(a) + D(b) - 2w(a, b)$, where D is external - internal edge costs:

$$D(a) = \sum_{b' \in B} w(a, b') - \sum_{a' \in A, a' \neq a} w(a, a')$$

$$D(b) = \sum_{a' \in A} w(b, a') - \sum_{b' \in B, b' \neq b} w(b, b')$$

Greedy refinement



Start with a partition $V = A \cup B$ and refine.

- $\mathrm{gain}(a, b) = D(a) + D(b) - 2w(a, b)$
- Purely greedy strategy: until no positive gain
  - Choose swap with most gain
  - Update D in neighborhood of swap; update gains
- Local minima are a problem.

Kernighan-Lin

In one sweep:

While no vertices marked

  Choose (a, b) with greatest gain

  Update $D(v)$ for all unmarked v as if (a, b) were swapped

  Mark a and b (but don't swap)

Find j such that swaps $1, \ldots, j$ yield maximal gain

Apply swaps $1, \ldots, j$

Usually converges in a few (2-6) sweeps. Each sweep is $O(|V|^3)$.
Can be improved to $O(|E|)$ (Fiduccia, Mattheyses).

Further improvements (Karypis, Kumar): only consider vertices on boundary, don't complete full sweep.

Multilevel ideas

Basic idea (same will work in other contexts):

- Coarsen
- Solve coarse problem
- Interpolate (and possibly refine)

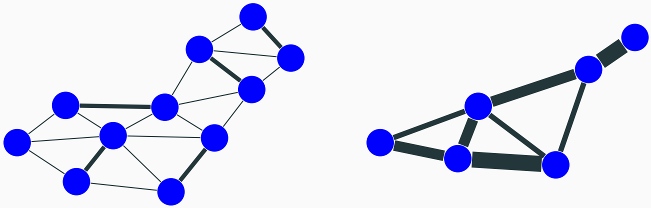
May apply recursively.

Maximal matching

One idea for coarsening: maximal matchings

- *Matching* of $G = (V, E)$ is $E_m \subset E$ with no common vertices.
- *Maximal*: cannot add edges and remain matching.
- Constructed by an obvious greedy algorithm.
- Maximal matchings are non-unique; some may be preferable to others (e.g. choose heavy edges first).

Coarsening via maximal matching



- Collapse nodes connected in matching into coarse nodes
- Add all edge weights between connected coarse nodes

Software

All these use some flavor(s) of multilevel:

- METIS/ParMETIS (Kapyris)
- PARTY (U. Paderborn)
- Chaco (Sandia)
- Scotch (INRIA)
- Jostle (now commercialized)
- Zoltan (Sandia)

Graph partitioning: Is this it?

Consider partitioning just for sparse matvec:

- Edge cuts $\neq$ communication volume
- Should we minimize *max* communication volume?
- Looked at communication volume – what about latencies?

Some go beyond graph partitioning (e.g. hypergraph in Zoltan).

Graph partitioning: Is this it?

Additional work on:

- Partitioning power law graphs
- Covering sets with small overlaps

Also: Classes of graphs with no small cuts (expanders)

Graph partitioning: Is this it?

Recall: partitioning for matvec *and* preconditioner

- Block Jacobi (or Schwarz) – relax on each partition
- Want to consider edge cuts *and physics*
  - E.g. consider edges = beams
  - Cutting a stiff beam worse than a flexible beam?
  - Doesn't show up from just the topology
- Multiple ways to deal with this
  - Encode physics via edge weights?
  - Partition geometrically?
- Tradeoffs are why we need to be *informed* users

Graph partitioning: Is this it?

So far, considered problems with *static* interactions

- What about particle simulations?
- Or what about tree searches?
- Or what about...?

Next time: more general *load balancing* issues