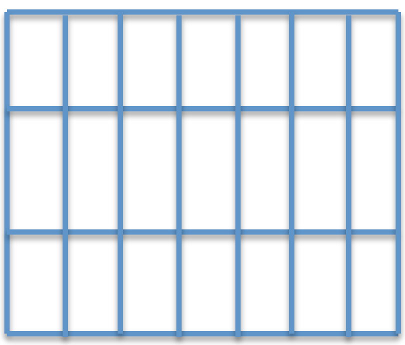
Graph Partitioning

Abhiram Ranade
IIT Bombay

A load balancing problem

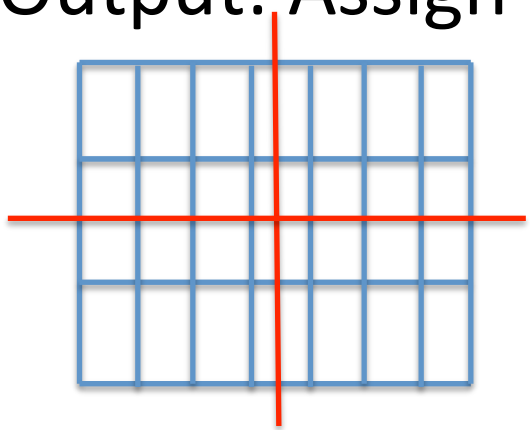
- Input:
  - undirected graph G. Vertices = processes. Edges = interprocess communication.
  - integer k = number of available processors.
- Output: Assign processes to processors



, 4

A load balancing problem

- Input:
  - undirected graph G. Vertices = processes. Edges = interprocess communication.
  - integer k = number of available processors.
- Output: Assign processes to processors

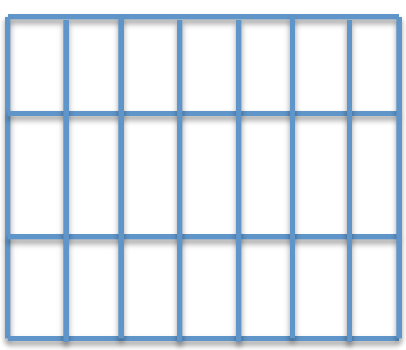


, 4

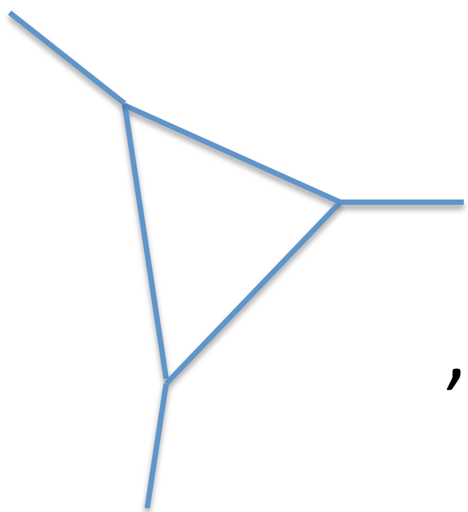
Give equal number of processes to each processor. Minimize actual communication

The graph partitioning problem

- Input: undirected graph G, integer k
- Output: Partition into subgraphs $G_1, G_2, \ldots, G_k$ with equal number of vertices, by removing minimum number of edges.



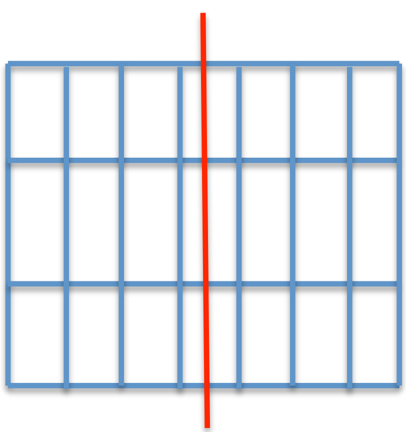
, 2



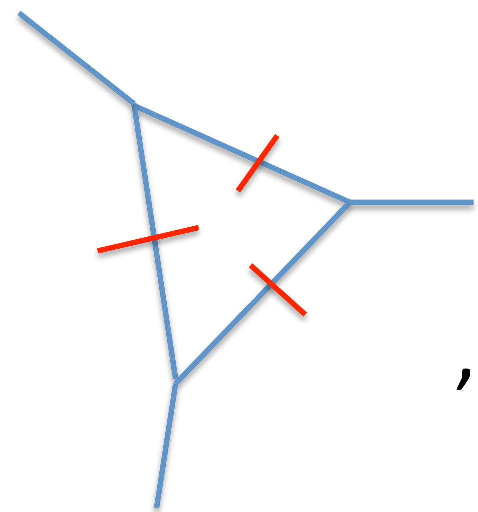
, 3

The graph partitioning problem

- Input: undirected graph G, integer k
- Output: Partition into subgraphs G_1, G_2, …, G_k with equal number of vertices, by removing minimum number of edges.



, 2



, 3

Other motivations

- Divide and conquer algorithms on graphs.
  - Partition into balanced subgraphs.
  - Recurse on subgraphs.
  - Cost of combine step : high if large number of edges between subgraphs.
- Identify bottlenecks in graph for purpose of fault tolerance etc.
- Nice connections to geomtry.

Rest of this talk: k=2, i.e. Graph bisection

Outline

- Basic results
- Partitioning special classes of graphs
  - Planar graphs
  - FEM (finite element method) graphs
- Partitioning arbitrary graphs
  - Spectral Methods
  - More advanced methods
  - Heuristics

Some basic results

- NP-hard even for k=2.
  - Reduction from MAX-CUT.
- Polytime algorithm known for trees
  - Moderately hard exercise. Dynamic programming
- Other graph classes: next
- Approximation algorithm: $O(\log^{1.5} n)$ factor known.
  - intro to some of the ideas: later

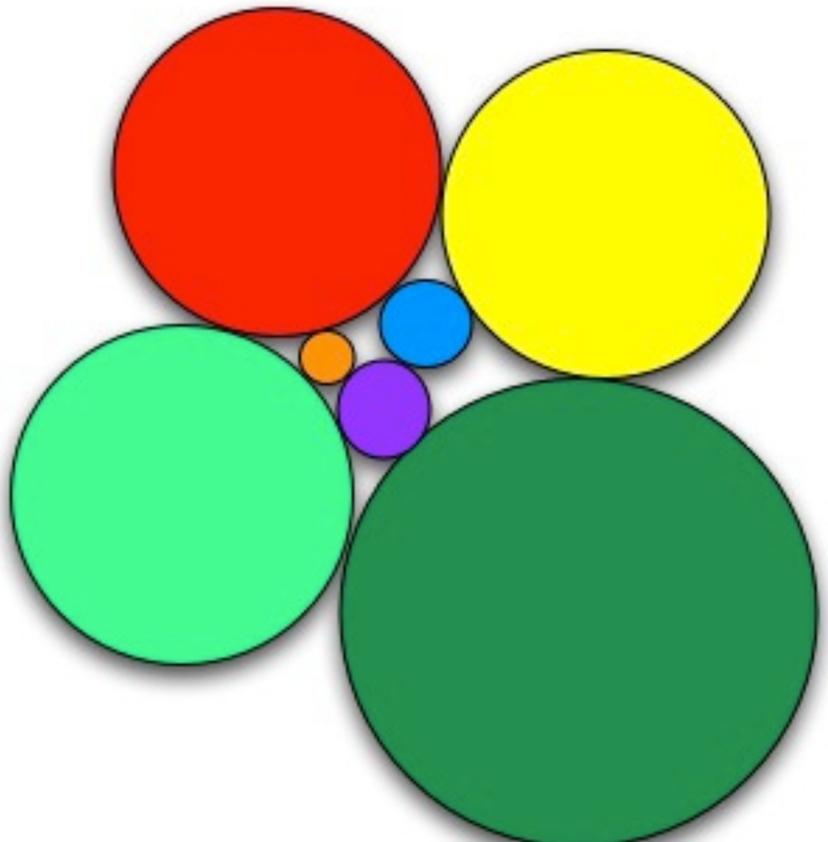
Planar graphs

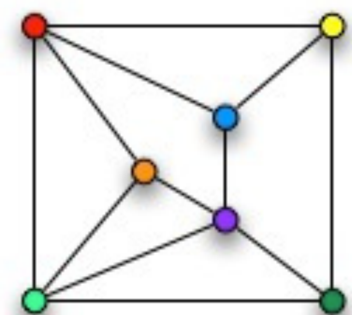
- NP-complete? Not known.
- Lipton-Tarjan Planar Separator Algorithm:
  - Any planar graph with n vertices can be bisected by removing $O(\sqrt{n})$ vertices and incident edges.
  - $O(\sqrt{n})$ edges for bounded degree graphs.
  - Algorithm may give $O(\sqrt{n})$ separator even if smaller exists.
  - Optimal for "Natural planar graphs" which cannot be bisected without removing $\Omega(\sqrt{n})$ vertices.

Planar separator theorem

- Every planar graph can be bisected by removing $O(\sqrt{n})$ vertices.
- Lipton-Tarjan proof:
  - uses clever breadth first search.
- Today: Alternate proof
  - Elegant connection to geometry. Extends to other geometrically embedded graphs, e.g. FEM graphs.

Representing planar graphs by "kissing" disks





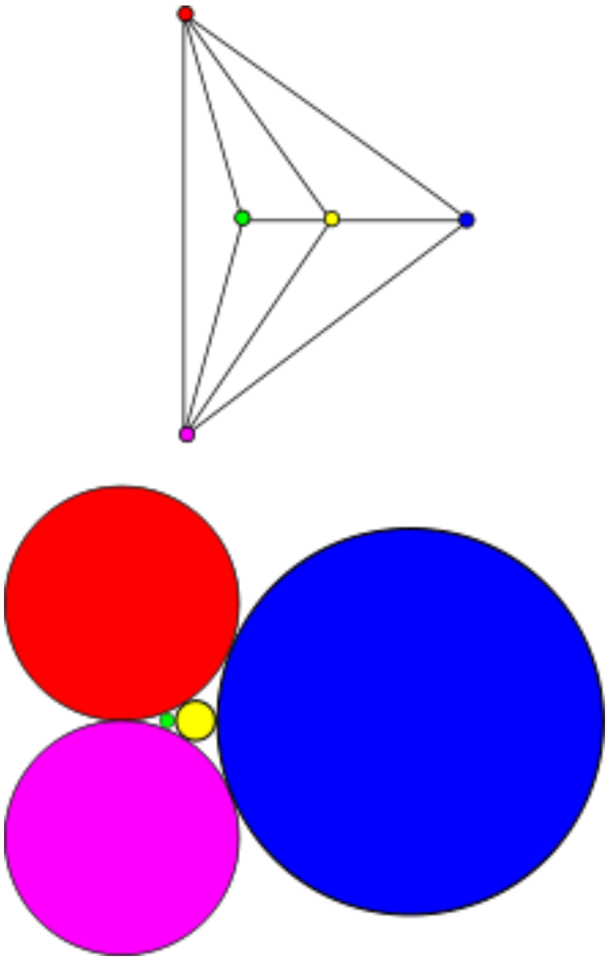
http://isabelandbenson.wordpress.com/2012/02/12/a-planar-separation-proof-using-koebes-theorem/

Circle packing theorem

Theorem [Koebe 1936] For every planar graph with n vertices, there exist disjoint circles $c_{1,}$ c_2, …, c_n such that c_i and c_j are tangential iff v_i, v_j are connected by an edge.

…Kissing disk embedding

Circle packing example



(Wikipedia)

Algorithm to find a Planar Separator

1. Get a kissing disk embedding.
2. Stereographically project it onto a sphere.
3. Find "center point" of disk centers.
4. Adjust so that center point moves to sphere center
5. Separator = random great circle through center.

Theorem:

Each hemisphere has $\theta(n)$ vertices.

Expected number of edges cut = $O(\sqrt{n})$.

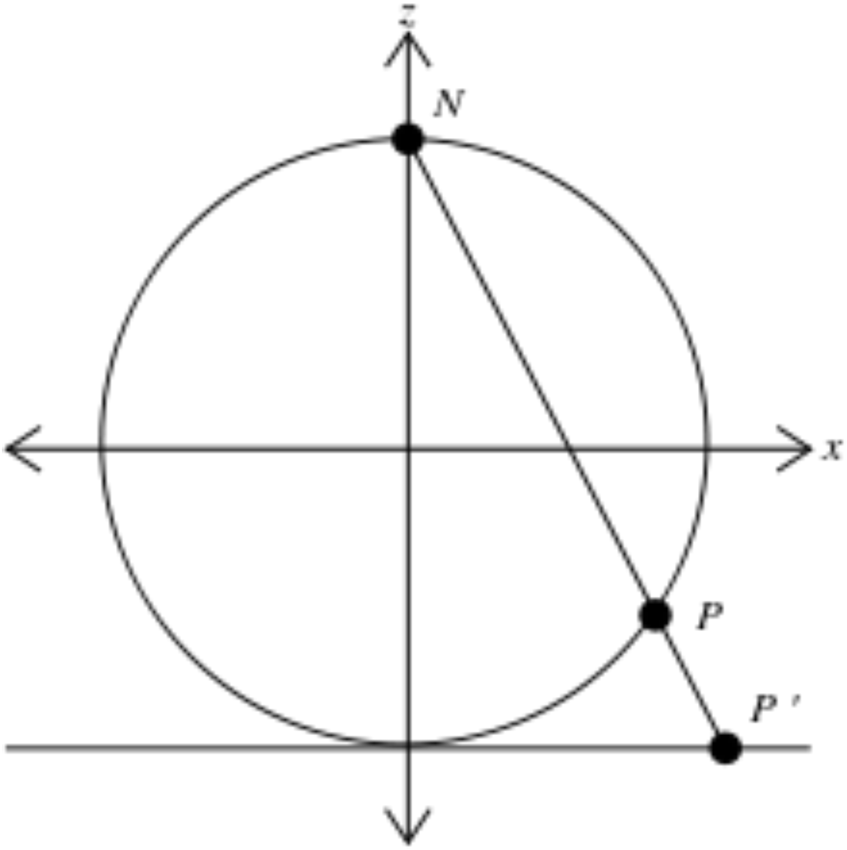
Remark

- Algorithm may not produce exact bisection, but say, ¼, ¾ split of vertices.
- Repeat on larger partition if balance not achieved. ¾ splits into 9/16, 3/16
- Combine ¼ + 3/16 = 7/16.
- Now we have better balance: 7/16, 9/16.
- Repeating log times suffices. Number of edges cut increase only by constant factor.

Stereographic projection

Input: Disk embedding on horizontal plane.

- Place a sphere on the plane.
- Point P' on plane mapped to P on sphere such that PP' passes through North Pole.
- Circles map to circles!



(Wikipedia)

Algorithm to find a Planar Separator

1. Get a kissing disk embedding. ✓
2. Stereographically project it onto a sphere. ✓
3. Find "center point" of disk centers.
4. Adjust so that center point moves to sphere center
5. Separator = random great circle through center.

Centerpoint C of n points

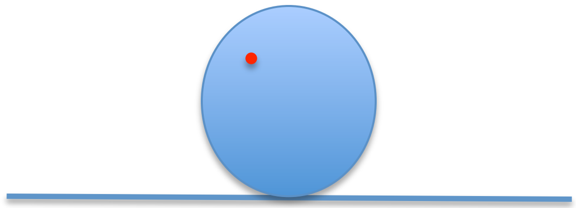
C = point such that any plane through it has at least n/4 points on both sides.

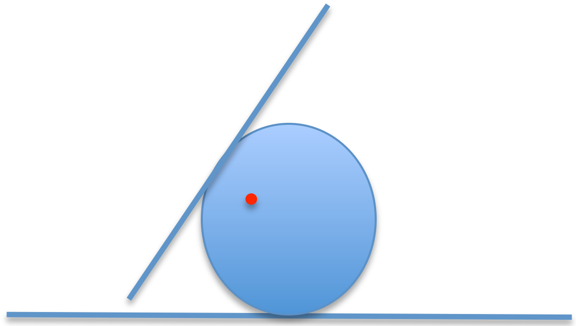
- Centerpoints exist, and may not be unique.
- Approximate centerpoints are easily found. Will guarantee at least n/Q points on both sides, where Q maybe > 4.

Algorithm to find a Planar Separator

1. Get a kissing disk embedding. ✓
2. Stereographically project it onto a sphere. ✓
3. Find "center point" of disk centers. ✓
4. Adjust so that center point moves to sphere center
5. Separator = random great circle through center.

Adjusting the centerpoint





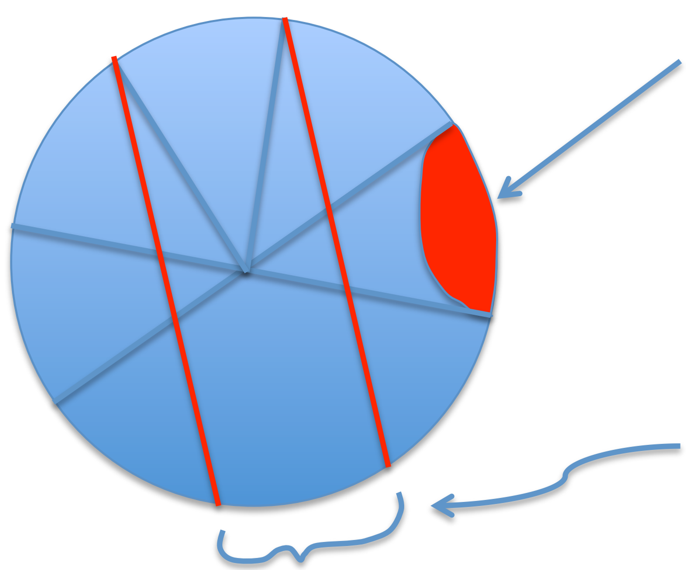
- Centerpoint shown in red.
- Project back onto inclined plane.
- Use a bigger sphere having same point of contact.
- How does red point move?

Where are we?

- We have a sphere with a disk on it for each vertex.
- Disks touch if corresponding vertices have an edge.
- Center of sphere = centerpoint of vertices.
- What will happen if we partition using some great circle?
- Centerpoint: $\theta(n)$ vertices will be on each side
- Will too many disks intersect with great circle?

How many disks intersect with a random great circle?

- Pick a random great circle = pick a random point as its north pole.
- Probability = Band area/sphere area $= 2\pi rR/4\pi R^2 = r/2R$



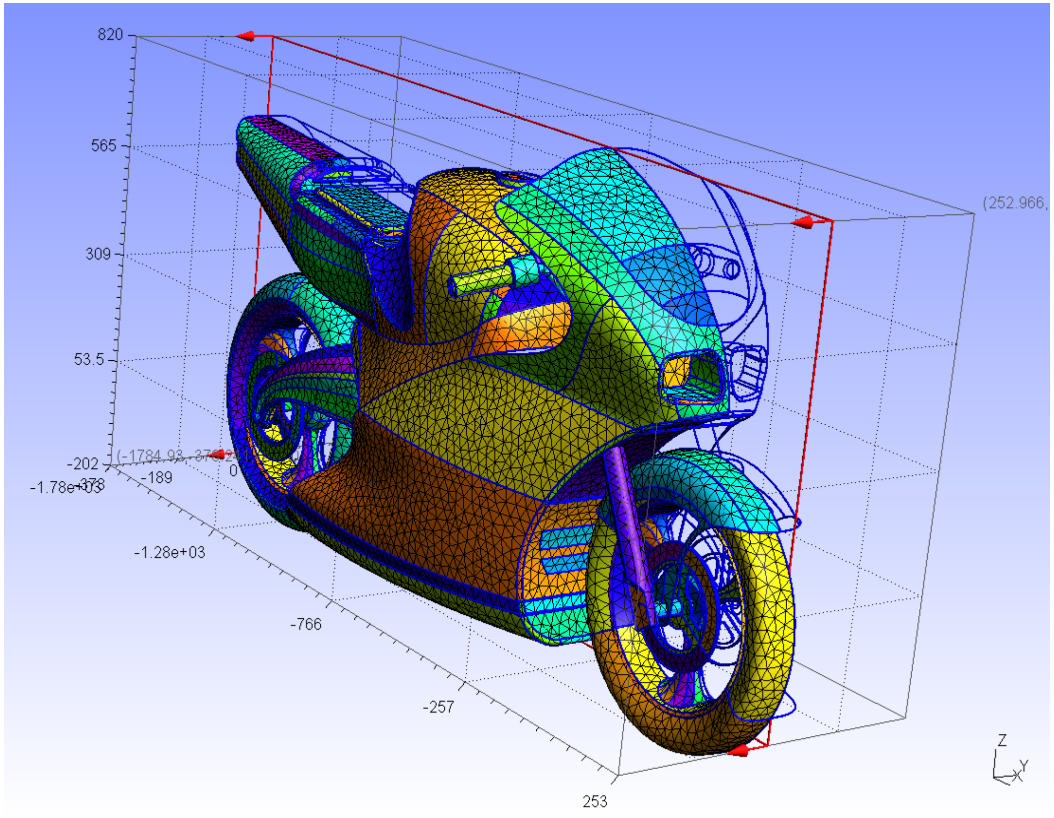
Expected number of intersections

- $E = r_1/2R + r_2/2R + \dots r_n/2R$
- We know: $\pi r_1^2 + \pi r_2^2 + \dots + \pi r_n^2 \leq 4\pi R^2$
- E is maximized when all r_i are equal, $r_i \leq 2R/\sqrt{n}$
- $E \leq n * (2R/\sqrt{n})/2R = \sqrt{n}$
- Plane has at least n/4 vertices on each side
- By removing incident edges on √n vertices we get ¼ - ¾ partition of vertices.
- Clearly such a partition exists! QED!

Remarks

- Nice connection between graph theory and geometry.
- Great circle when projected back, becomes a circle on the graph. “Circle separator”
- Do we need circles? Can we hope to get good partitions using a straight line?
  - There exist graphs for which this is not possible.
- In practice, Lipton-Tarjan’s original algorithm will be faster than above algorithm.

Finite Element Method (FEM) Graphs



http://www.geuz.org/gmsh/gallery/bike.png

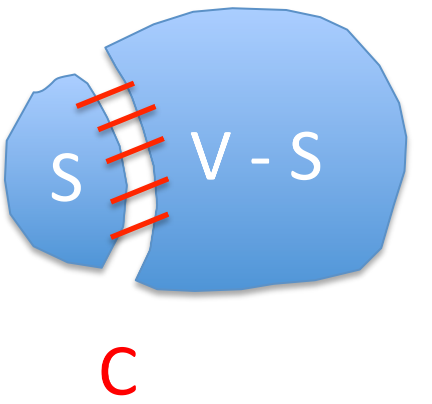
FEM Graphs

- 3 dimensional object represented by collection of volumes, “elements”
- Element size:
  - Small where physical properties vary a lot.
  - Large where physical properties vary less.
- “Aspect ratio” of elements is small, i.e. elements are not flat/narrow.
  - Useful for to reduce floating point overflow..

FEM Graphs continued

- Effectively we have a “kissing disk” embedding of elements in 3 dimensional space.
- Theorem[Miller-Thurston 1990] A 3d FEM mesh with n elements of small aspect ratio can be bisected by removing $O(n^{2/3})$ edges.
- Partitioning FEM graphs is useful for load balancing on parallel computers.

Partitioning General Graphs



- Don't insist on bisection.
- Sparsity = |C|/|S|
- Cut ratio = |C|/(|S||V-S|)
- Sparsity/C.Ratio ≈ |V|
- We study algorithms that attempt to find cuts of minimum ratio.
- We can get bisection by repeating if necessary.

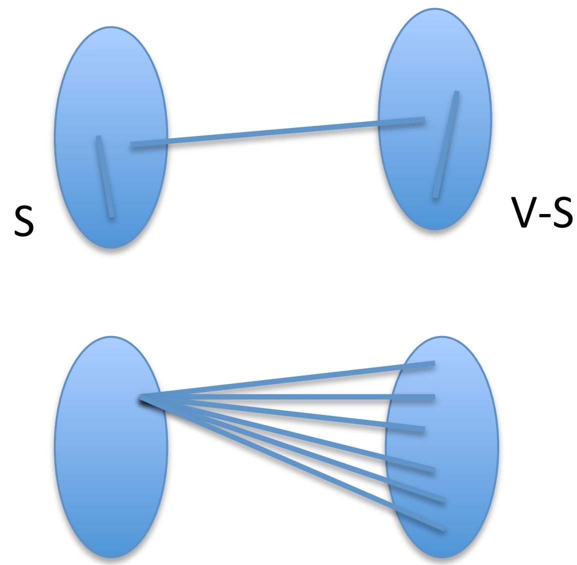
Strategy

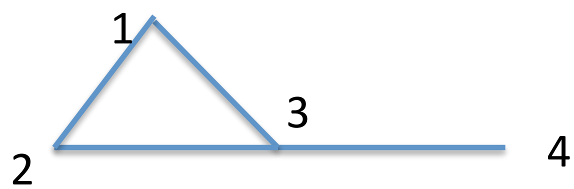
Our optimization problem is expressed in the language of graphs.

1. Pose it as a problem on integer variables.
   – NP-complete.
2. Find a similar problem on real variables that can be solved in polynomial time.
3. Exploiting the symmetry, get back an integer solution. Hope: it is close to the optimal.

Algebraic definition of cut ratio

- x = bit vector. x_i = 1 iff ith vertex is in S.
- No. of crossing edges:
$$\sum_{(i,j)\in E} |x_i - x_j|$$
- |S||V-S| = $\sum_{i<j} |x_i - x_j|$
- Want to minimize
$$\frac{\sum_{(i,j)\in E} |x_i - x_j|}{\sum_{i<j} |x_i - x_j|}$$





$$\min_{x_i\in\{0,1\}} \frac{|x_1 - x_2| + |x_1 - x_3| + |x_2 - x_3| + |x_3 - x_4|}{|x_1 - x_2| + |x_1 - x_3| + |x_1 - x_4| + |x_2 - x_3| + |x_2 - x_4| + |x_3 - x_4|}$$

General Algorithmic Strategy: Relaxation

- Problem 1: minimize $3x^2 - 5x$, x is an integer.
- Problem 2: minimize $3x^2 - 5x$, x is real.
  - Easier to solve, make derivative = 0 etc.
  - min(Problem 1) ≥ min(Problem 2)
- Problem 2 = Relaxation of problem 1
- Strategy to solve problem 1: solve problem 2 and then take a nearby solution.
- Minimizing cut ratio: similar idea..

Relaxation to "solve" for cut ratio

- Want to find
$$r = \min_{x_i \in \{0,1\}} \frac{\sum_{(i,j)\in E} |x_i - x_j|}{\sum_{i<j} |x_i - x_j|} = \min_{x_i \in \{0,1\}} \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\sum_{i<j} (x_i - x_j)^2}$$
- Allow real values
  - Suppose we can solve for x.
  - We will "round" solution
  - high = 1. Low = 0.
$$r \geq \min_{x_i \in R} \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\sum_{i<j} (x_i - x_j)^2}$$
- Insist that x_i add to 1
$$r \geq \min_{x \cdot 1 = 0} \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\sum_{i<j} (x_i - x_j)^2}$$

contd.

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

L_{ij} = -1 if (i,j) is an edge,
= degree(i) if i=j,
=0 otherwise
L = “Laplacian of the graph”

$$Deno\min ator = \sum_{i<j} (x_i - x_j)^2 = \sum_{i<j} x_i^2 - 2x_i x_j + x_j^2 = (n-1)\sum_i x_i^2 - 2\sum_{i>j} x_i x_j = n\sum_i x_i^2$$

$$r \geq \min_{x\cdot 1=0} \frac{\sum_{(i,j)\in E} (x_i - x_j)^2}{\sum_{i<j} (x_i - x_j)^2} = \min_{x\cdot 1=0} \frac{x^T L x}{n\sum_i x_i^2} = \frac{\sigma_2}{n}$$

σ_2 is second smallest “Eigenvalue of L”
Can be calculated quickly.
minimizing x can also be calculated.

Algorithm for finding good ratio cut

1. Construct Laplacian matrix
2. Find second smallest eigenvector
3. Choose threshold t.
4. Partition = $\{i \mid x_i \leq t\}, \{i \mid x_i > t\}$

How to choose t? Try all possible choices, and take one having best ratio.

"Spectral Method" : eigenvectors involved.

How good is the partition

- Cheeger's Theorem:
  - Partition found by previous algorithm will have cut ratio at most $\frac{\sqrt{8d\sigma_2}}{n}$

  Difficult proof

  - Optimal partition will have cut ratio at least $\frac{\sigma_2}{n}$

  Just proved!

- Seems to work well in practice

Examples

- Cycle on n vertices: Spectral method finds optimal partition.
- Bounded degree planar graphs: Spectral method finds $O(\sqrt{n})$ sized bisector.
- Binary hypercube: Spectral methods will find cut of ratio $\sqrt{4 \log n} / n$, whereas opt ratio = $1/n$

Remarks about intuition

- How did we steer the analysis towards the Laplacian?
  - Experience
  - Laplacian arises in solving differential equation, where second eigenvalue gives how graph may “vibrate”. Vibration mode points to cuts.
  - Laplacian is related to SVD, singular value decomposition. SVD of a point cloud identifies length, breadth, width of cloud. Small cuts are perpendicular to length.

Advanced methods

- Starting point:

$$r = \min_{x_i \in \{0,1\}} \frac{\sum_{(i,j)\in E} |x_i - x_j|}{\sum_{i<j} |x_i - x_j|}$$

- Different ways to relax.
  - If you relax too much, you may be able to solve the new problem more easily, but the solution will be too far.
- Arora-Rao-Vazirani 04 : stricter relaxation. Always give cut with ratio = O(√log n) times optimal.

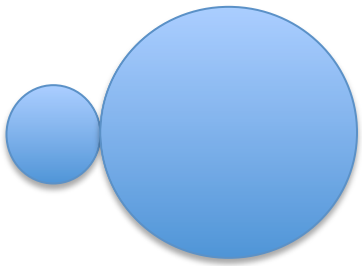
Heuristics

- Graph coarsening [Metis 04]:
  - Pick disjoint edges in graph.
  - Merge endpoints of each edge into a single vertex. (fast)
  - Repeat several times.
  - Find partition for final “coarse” graph using sophisticated method. (slow, but on small graph)
  - Final partition induces partition on original graph.

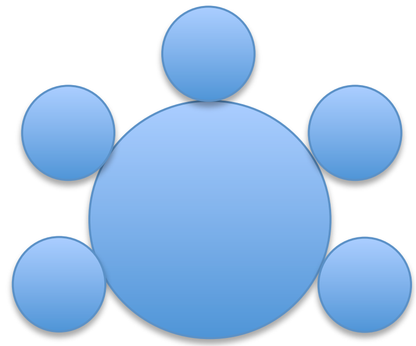
Concluding Remarks

- Important problem
- Many applications
- Many interesting methods. Theory + Heuristic.
- Deep connections to geometry.

Generalization of bisection



- We may need to cut many edges to bisect, but by cutting very few we may get ¼ - ¾ split.
- May be useful to know such “bottlenecks”



- Even for bisection, it might be easier to get one piece at a time

c-balanced partitioning

- For a fixed c, remove as few edges as possible to get subgraphs of size cn, (1-c)n.
- If c=O(1), we can use repeatedly to achieve bisection. Example: c = 0.1
  - After 1 application we have 0.1 – 0.9 split.
  - Apply once more on 0.9 : 0.1, 0.09, 0.81
  - Apply log times on larger piece and extract n/2 vertices. Bisection!