

# CS 267: Applications of Parallel Computers

## Graph Partitioning

Kathy Yelick

<http://www.cs.berkeley.edu/~yelick/cs267>

4/15/2004 CS267, Yelick 1

### Outline of Graph Partitioning Lectures

- Review definition of Graph Partitioning problem
- Overview of heuristics
- Partitioning with Nodal Coordinates
  - Planar graphs
  - How well can graphs be partitioned in theory?
  - Graphs in higher dimensions
- Partitioning without Nodal Coordinates

- Multilevel Acceleration
  - BIG IDEA**, appears often in scientific computing
- Comparison of Methods and Applications

### Definition of Graph Partitioning

- Given a graph  $G = (N, E, W_N, W_E)$ 
  - $N$  = nodes (or vertices),
  - $E$  = edges
  - $W_N$  = node weights
  - $W_E$  = edge weights

- Ex:  $N$  = {tasks},  $W_N$  = {task costs}, edge  $(j,k)$  in  $E$  means task  $j$  sends  $W_E(j,k)$  words to task  $k$
- Choose a partition  $N = N_1 \cup N_2 \cup \dots \cup N_p$  such that
  - The sum of the node weights in each  $N_j$  is "about the same"
  - The sum of all edge weights of edges connecting all different pairs  $N_j$  and  $N_k$  is minimized
- Ex: balance the work load, while minimizing communication
- Special case of  $N = N_1 \cup N_2$ : Graph Bisection

4/15/2004 CS267, Yelick 3

### Applications

- Telephone network design
  - Original application, algorithm due to Kernighan
- Load Balancing while Minimizing Communication
- Sparse Matrix times Vector Multiplication
  - Solving PDEs
  - $N = \{1, \dots, n\}$ ,  $(j,k)$  in  $E$  if  $A(j,k)$  nonzero,
  - $W_N(j) = \text{\#nonzeros in row } j$ ,  $W_E(j,k) = 1$
- VLSI Layout
  - $N$  = {units on chip},  $E$  = {wires},  $W_E(j,k)$  = wire length
- Sparse Gaussian Elimination
  - Used to reorder rows and columns to increase parallelism, and to decrease "fill-in"
- Data mining and clustering
- Physical Mapping of DNA

4/15/2004 CS267, Yelick 4

### Cost of Graph Partitioning

- Many possible partitionings to search
- Just to divide in 2 parts there are:
  - $n$  choose  $n/2$  ~
  - $\sqrt{2n/\pi} * 2^n$  possibilities

- Choosing optimal partitioning is NP-complete
  - (NP-complete = we can prove it is as hard as other well-known hard problems in a class Nondeterministic Polynomial time)
  - Only known exact algorithms have cost = exponential( $n$ )
- We need good heuristics

4/15/2004 CS267, Yelick 6

### First Heuristic: Repeated Graph Bisection

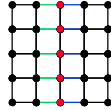
- To partition  $N$  into  $2^k$  parts
  - bisect graph recursively  $k$  times
- Henceforth discuss mostly graph bisection

4/15/2004 CS267, Yelick 7

### Edge Separators vs. Vertex Separators

- **Edge Separator:**  $E_s$  (subset of  $E$ ) separates  $G$  if removing  $E_s$  from  $E$  leaves two ~equal-sized, disconnected components of  $N$ :  $N_1$  and  $N_2$
- **Vertex Separator:**  $N_s$  (subset of  $N$ ) separates  $G$  if removing  $N_s$  and all incident edges leaves two ~equal-sized, disconnected components of  $N$ :  $N_1$  and  $N_2$

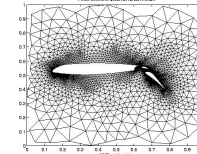
$G = (N, E)$ , Nodes  $N$  and Edges  $E$   
 $E_s$  = green edges or blue edges  
 $N_s$  = red vertices



- Making an  $N_s$  from an  $E_s$ : pick one endpoint of each edge in  $E_s$ 
  - $|N_s| \leq |E_s|$  ?
- Making an  $E_s$  from an  $N_s$ : pick all edges incident on  $N_s$ 
  - $|E_s| \leq d * |N_s|$  where  $d$  is the maximum degree of the graph ?
- We will find Edge or Vertex Separators, as convenient

### Overview of Bisection Heuristics

- Partitioning with Nodal Coordinates
  - Each node has  $x, y, z$  coordinates  $\rightarrow$  partition space



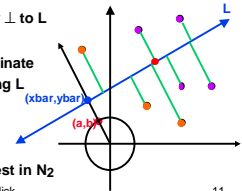
- Partitioning without Nodal Coordinates
  - E.g., Sparse matrix of Web documents
    - $A(i,k) = \#$  times keyword  $j$  appears in URL  $k$
- Multilevel acceleration (**BIG IDEA**)
  - Approximate problem by "coarse graph," do so recursively

### Nodal Coordinates: How Well Can We Do?

- Consider a special case:
  - A graph with nodal coordinates
  - The graph is planar
- A planar graph can be drawn in plane without edge crossings
- Ex:  $m \times m$  grid of  $m^2$  nodes:  $\exists$  vertex separator  $N_s$  with  $|N_s| = m = \sqrt{|N|}$  (see last slide for  $m=5$ )
- **Theorem** (Tarjan, Lipton, 1979): If  $G$  is planar,  $\exists N_s$  such that
  - $N = N_1 \cup N_s \cup N_2$  is a partition,
  - $|N_1| \leq 2/3 |N|$  and  $|N_2| \leq 2/3 |N|$
  - $|N_s| \leq \sqrt{8 * |N|}$
- Theorem motivates intuition of following algorithms

### Nodal Coordinates: Inertial Partitioning

- For a graph in 2D, choose line with half the nodes on one side and half on the other
  - In 3D, choose a plane, but consider 2D for simplicity
- Choose a line  $L$ , and then choose an  $L^\perp$  perpendicular to it, with half the nodes on either side
- 1. **Choose a line  $L$  through the points**  
 $L$  given by  $a^*(x-xbar)+b^*(y-ybar)=0$ , with  $a^2+b^2=1$ ;  $(a,b)$  is unit vector  $\perp$  to  $L$
- 2. **Project each point to the line**  
 For each  $n_j = (x_j, y_j)$ , compute coordinate  $S_j = -b^*(x_j-xbar) + a^*(y_j-ybar)$  along  $L$
- 3. **Compute the median**  
 Let  $Sbar = \text{median}(S_1, \dots, S_n)$
- 4. **Use median to partition the nodes**  
 Let nodes with  $S_j < Sbar$  be in  $N_1$ , rest in  $N_2$

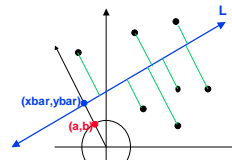


### Inertial Partitioning: Choosing L

- 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

### Inertial Partitioning: choosing L (continued)

$(a,b)$  is unit vector perpendicular to  $L$

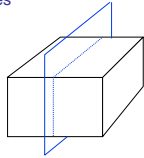


$$\begin{aligned}
 \Sigma_j & \text{ (length of } j\text{-th green line)}^2 \\
 &= \Sigma_j [(x_j - xbar)^2 + (y_j - ybar)^2 - (-b^*(x_j - xbar) + a^*(y_j - ybar))^2] \\
 & \dots \text{ Pythagorean Theorem} \\
 &= a^2 * \Sigma_j (x_j - xbar)^2 + 2*a*b * \Sigma_j (x_j - xbar)*(y_j - ybar) + b^2 * \Sigma_j (y_j - ybar)^2 \\
 &= a^2 * X1 + 2*a*b * X2 + b^2 * X3 \\
 &= [a \ b] * \begin{bmatrix} X1 & X2 \\ X2 & X3 \end{bmatrix} * \begin{bmatrix} a \\ b \end{bmatrix}
 \end{aligned}$$

Minimized by choosing  $(xbar, ybar) = (\Sigma_j x_j, \Sigma_j y_j) / N = \text{center of mass}$ ,  $(a,b) = \text{eigenvector of smallest eigenvalue of } \begin{bmatrix} X1 & X2 \\ X2 & X3 \end{bmatrix}$

**Nodal Coordinates: Random Spheres**

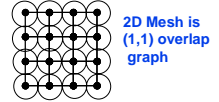
- Generalize nearest neighbor idea of a planar graph to higher dimensions
- For intuition, consider a the graph defined by a regular 3D mesh
- An  $n$  by  $n$  by  $n$  mesh of  $|N| = n^3$  nodes
  - Edges to 6 nearest neighbors
  - Partition by taking plane parallel to 2 axes
  - Cuts  $n^2 = |N|^{2/3} = O(|E|^{2/3})$  edges
- For the general graphs
  - Need a notion of well-shaped
  - (Any graph fits in 3D without crossings!)



**Random Spheres: Well Shaped Graphs**

- Approach due to Miller, Teng, Thurston, Vavasis
- **Def:** A  $k$ -ply neighborhood system in  $d$  dimensions is a set  $\{D_1, \dots, D_n\}$  of closed disks in  $R^d$  such that no point in  $R^d$  is strictly interior to more than  $k$  disks
- **Def:** An  $(\alpha, k)$  overlap graph is a graph defined in terms of  $\alpha \geq 1$  and a  $k$ -ply neighborhood system  $\{D_1, \dots, D_n\}$ : There is a node for each  $D_i$ , and an edge from  $j$  to  $i$  if expanding the radius of the smaller of  $D_j$  and  $D_i$  by  $>\alpha$  causes the two disks to overlap

Ex:  $n$ -by- $n$  mesh is a  $(1,1)$  overlap graph  
 Ex: Any planar graph is  $(\alpha, k)$  overlap for some  $\alpha, k$

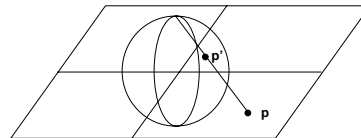


**Generalizing Lipton/Tarjan to Higher Dimensions**

- **Theorem** (Miller, Teng, Thurston, Vavasis, 1993): Let  $G=(N,E)$  be an  $(\alpha, k)$  overlap graph in  $d$  dimensions with  $n=|N|$ . Then there is a vertex separator  $N_s$  such that
  - $N = N_1 \cup N_s \cup N_2$  and
  - $N_1$  and  $N_2$  each has at most  $n^{(d+1)/(d+2)}$  nodes
  - $N_s$  has at most  $O(\alpha \cdot k^{1/d} \cdot n^{(d-1)/d})$  nodes
- When  $d=2$ , same as Lipton/Tarjan
- Algorithm:
  - Choose a sphere  $S$  in  $R^d$
  - Edges that  $S$  "cuts" form edge separator  $E_s$
  - Build  $N_s$  from  $E_s$
  - Choose "randomly", so that it satisfies Theorem with high probability

**Stereographic Projection**

- Stereographic projection from plane to sphere
  - In  $d=2$ , draw line from  $p$  to North Pole, projection  $p'$  of  $p$  is where the line and sphere intersect



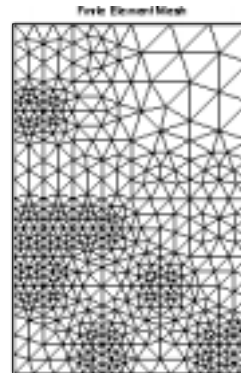
$p = (x,y)$       $p' = (2x, 2y, x^2 + y^2 - 1) / (x^2 + y^2 + 1)$

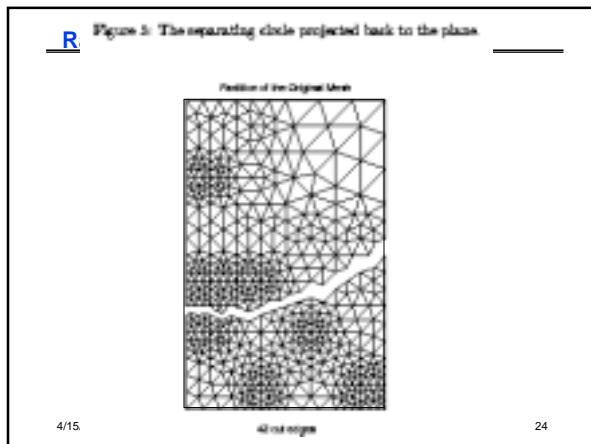
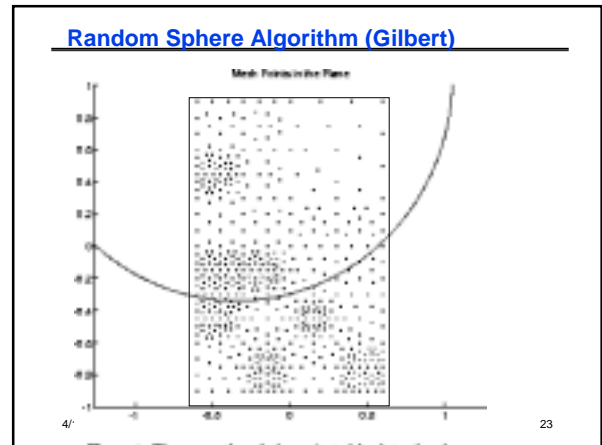
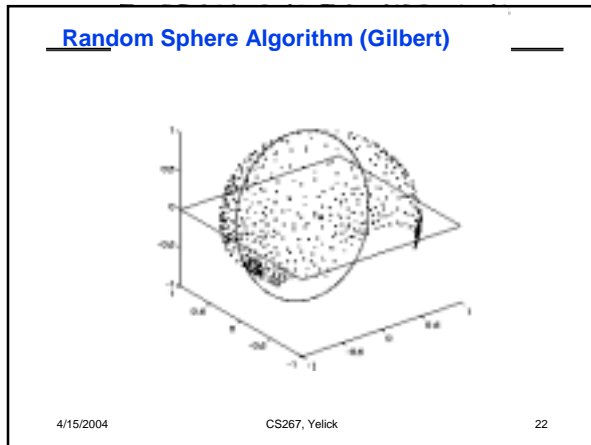
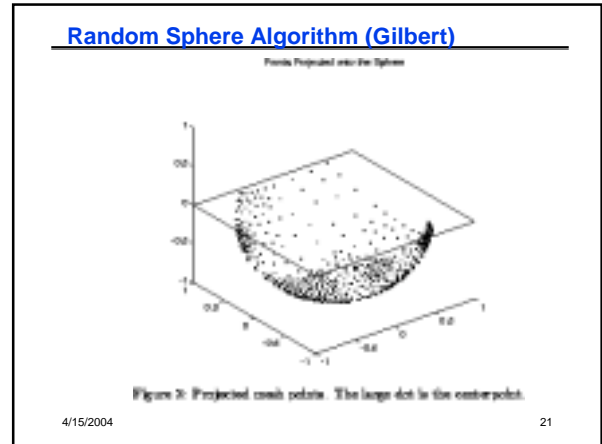
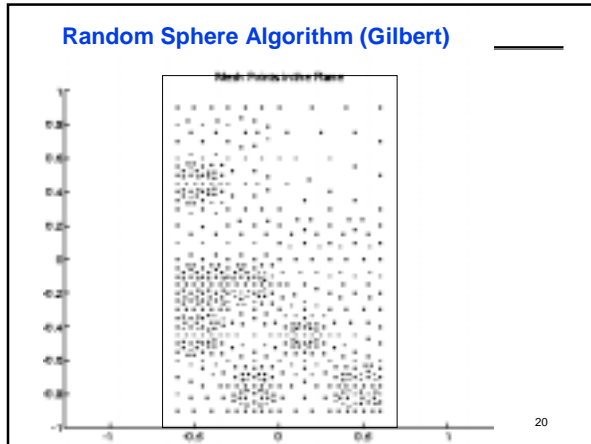
- Similar in higher dimensions

**Choosing a Random Sphere**

- Do stereographic projection from  $R^d$  to sphere in  $R^{d+1}$
- Find **centerpoint** of projected points
  - Any plane through centerpoint divides points ~evenly
  - There is a linear programming algorithm, cheaper heuristics
- **Conformally map** points on sphere
  - Rotate points around origin so centerpoint at  $(0, \dots, 0, r)$  for some  $r$
  - Dilate points (unproject, multiply by  $\sqrt{((1-r)/(1+r))}$ , project)
    - this maps centerpoint to origin  $(0, \dots, 0)$
- Pick a random plane through origin
  - Intersection of plane and sphere is circle
- Unproject circle
  - yields desired circle  $C$  in  $R^d$
- Create  $N_s$ :  $j$  belongs to  $N_s$  if  $\alpha \cdot D_j$  intersects  $C$

**Random Sphere Algorithm (Gilbert)**





### Nodal Coordinates: Summary

- Other variations on these algorithms
- Algorithms 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
- Ignore edges, but can be used as good starting guess for subsequent partitioners that do examine edges
- Can do poorly if graph connection is not spatial:
- Details at
  - [www.cs.berkeley.edu/~demmel/cs267/lecture18/lecture18.html](http://www.cs.berkeley.edu/~demmel/cs267/lecture18/lecture18.html)
  - [www.parc.xerox.com/spl/members/gilbert](http://www.parc.xerox.com/spl/members/gilbert) (tech reports and SW)
  - [www-sal.cs.uiuc.edu/~steng](http://www-sal.cs.uiuc.edu/~steng)

4/15/2004

CS267, Yelick

### Coordinate-Free: Breadth First Search (BFS)

- Given  $G(N,E)$  and a root node  $r$  in  $N$ , BFS produces
  - A subgraph  $T$  of  $G$  (same nodes, subset of edges)
  - $T$  is a tree rooted at  $r$
  - Each node assigned a level = distance from  $r$

4/15/2004 CS267, Yelick 26

### Breadth First Search

- Queue (First In First Out, or FIFO)
  - Enqueue( $x,Q$ ) adds  $x$  to back of  $Q$
  - $x = \text{Dequeue}(Q)$  removes  $x$  from front of  $Q$
- Compute Tree  $T(N_T, E_T)$

```

N_T = {(r,0)}, E_T = empty set ... Initially T = root r, which is at level 0
Enqueue((r,0),Q) ... Put root on initially empty Queue Q
Mark r ... Mark root as having been processed
While Q not empty ... While nodes remain to be processed
  (n,level) = Dequeue(Q) ... Get a node to process
  For all unmarked children c of n
    N_T = N_T U (c,level+1) ... Add child c to N_T
    E_T = E_T U (n,c) ... Add edge (n,c) to E_T
    Enqueue((c,level+1),Q) ... Add child c to Q for processing
    Mark c ... Mark c as processed
  Endfor
Endwhile
    
```

4/15/2004 CS267, Yelick 27

### Partitioning via Breadth First Search

- BFS identifies 3 kinds of edges
  - Tree Edges - part of  $T$
  - Horizontal Edges - connect nodes at same level
  - Interlevel Edges - connect nodes at adjacent levels
- No edges connect nodes in levels differing by more than 1 (why?)
- BFS partitioning heuristic
  - $N = N_1 \cup N_2$ , where
    - $N_1 = \{\text{nodes at level } \leq L\}$ ,
    - $N_2 = \{\text{nodes at level } > L\}$
  - Choose  $L$  so  $|N_1|$  close to  $|N_2|$

BFS partition of a 2D Mesh using center as root:  
 $N_1 = \text{levels } 0, 1, 2, 3$   
 $N_2 = \text{levels } 4, 5, 6$

4/15/2004 CS267, Yelick 28

### Coordinate-Free: Kernighan/Lin

- Take a initial partition and iteratively improve it
  - Kernighan/Lin (1970), cost =  $O(|N|^3)$  but easy to understand
  - Fiduccia/Mattheyses (1982), cost =  $O(|E|)$ , much better, but more complicated
- Given  $G = (N,E,W_E)$  and a partitioning  $N = A \cup B$ , where  $|A| = |B|$ 
  - $T = \text{cost}(A,B) = \sum \{W(e) \text{ where } e \text{ connects nodes in } A \text{ and } B\}$
  - Find subsets  $X$  of  $A$  and  $Y$  of  $B$  with  $|X| = |Y|$
  - Swapping  $X$  and  $Y$  should decrease cost:
    - $\text{newA} = A - X \cup Y$  and  $\text{newB} = B - Y \cup X$
    - $\text{newT} = \text{cost}(\text{newA}, \text{newB}) < \text{cost}(A,B)$
- Need to compute newT efficiently for many possible  $X$  and  $Y$ , choose smallest

4/15/2004 CS267, Yelick 29

### Kernighan/Lin: Preliminary Definitions

- $T = \text{cost}(A, B)$ ,  $\text{newT} = \text{cost}(\text{newA}, \text{newB})$
- Need an efficient formula for newT; will use
  - $E(a) = \text{external cost of } a \text{ in } A = \sum \{W(a,b) \text{ for } b \text{ in } B\}$
  - $I(a) = \text{internal cost of } a \text{ in } A = \sum \{W(a,a') \text{ for other } a' \text{ in } A\}$
  - $D(a) = \text{cost of } a \text{ in } A = E(a) - I(a)$
  - $E(b)$ ,  $I(b)$  and  $D(b)$  defined analogously for  $b$  in  $B$
- Consider swapping  $X = \{a\}$  and  $Y = \{b\}$ 
  - $\text{newA} = A - \{a\} \cup \{b\}$ ,  $\text{newB} = B - \{b\} \cup \{a\}$
- $\text{newT} = T - (D(a) + D(b) - 2 * w(a,b)) = T - \text{gain}(a,b)$ 
  - $\text{gain}(a,b)$  measures improvement gotten by swapping  $a$  and  $b$
- Update formulas
  - $\text{newD}(a') = D(a') + 2 * w(a',a) - 2 * w(a',b)$  for  $a'$  in  $A$ ,  $a' \neq a$
  - $\text{newD}(b') = D(b') + 2 * w(b',b) - 2 * w(b',a)$  for  $b'$  in  $B$ ,  $b' \neq b$

4/15/2004 CS267, Yelick 30

### Kernighan/Lin Algorithm

```

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

4/15/2004 CS267, Yelick 31

### Comments on Kernighan/Lin Algorithm

- Most expensive line show in red
- Some gain(k) may be negative, but if later gains are large, then final Gain may be positive
  - can escape "local minima" where switching no pair helps
- How many times do we Repeat?
  - K/L tested on very small graphs ( $|N| \leq 360$ ) and got convergence after 2-4 sweeps
  - For random graphs (of theoretical interest) the probability of convergence in one step appears to drop like  $2^{-|N|/30}$

4/15/2004 CS267, Yelick 32

### Coordinate-Free: Spectral Bisection

- Based on theory of Fiedler (1970s), popularized by Pothen, Simon, Liou (1990)
- Motivation, by analogy to a vibrating string
- Basic definitions
- Vibrating string, revisited
- Implementation via the Lanczos Algorithm
  - To optimize sparse-matrix-vector multiply, we graph partition
  - To graph partition, we find an eigenvector of a matrix associated with the graph
  - To find an eigenvector, we do sparse-matrix vector multiply
  - No free lunch ...

4/15/2004 CS267, Yelick 33

### Motivation for Spectral Bisection

- Vibrating string
- Think of  $G = 1D$  mesh as masses (nodes) connected by springs (edges), i.e. a string that can vibrate
- Vibrating string has modes of vibration, or harmonics
- Label nodes by whether mode - or + to partition into N- and N+
- Same idea for other graphs (eg planar graph ~ trampoline)

Modes of a Vibrating String

Lowest Frequency  $\lambda(1)$

Second Frequency  $\lambda(2)$

Third Frequency  $\lambda(3)$

4/15/2004 CS267, Yelick 34

### Basic Definitions

- **Definition:** The **incidence matrix**  $In(G)$  of a graph  $G(N,E)$  is an  $|N|$  by  $|E|$  matrix, with one row for each node and one column for each edge. If edge  $e=(i,j)$  then column  $e$  of  $In(G)$  is zero except for the  $i$ -th and  $j$ -th entries, which are  $+1$  and  $-1$ , respectively.
- Slightly ambiguous definition because multiplying column  $e$  of  $In(G)$  by  $-1$  still satisfies the definition, but this won't matter...
- **Definition:** The **Laplacian matrix**  $L(G)$  of a graph  $G(N,E)$  is an  $|N|$  by  $|N|$  symmetric matrix, with one row and column for each node. It is defined by
  - $L(G) (i,i) = \text{degree of node } i$  (number of incident edges)
  - $L(G) (i,j) = -1$  if  $i \neq j$  and there is an edge  $(i,j)$
  - $L(G) (i,j) = 0$  otherwise

4/15/2004 CS267, Yelick 35

### Example of $In(G)$ and $L(G)$ for Simple Meshes

Incidence and Laplacian Matrices

Graph G

Incidence Matrix  $In(G)$

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ -1 & & & \\ 1 & -1 & & \\ & 1 & -1 & \\ & & 1 & -1 \\ & & & 1 \end{bmatrix}$$

Laplacian Matrix  $L(G)$

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & & & \\ -1 & 1 & 1 & & \\ & -1 & 2 & 1 & \\ & & 1 & -2 & 1 \\ & & & -1 & 2 \end{bmatrix}$$

Nodes numbered in black  
Edges numbered in blue

4/15/2004 CS267, Yelick 36

### Properties of Laplacian Matrix

- **Theorem 1:** Given  $G$ ,  $L(G)$  has the following properties (proof on web page)
  - $L(G)$  is symmetric.
    - This means the eigenvalues of  $L(G)$  are real and its eigenvectors are real and orthogonal.
  - Rows of  $L$  sum to zero:
    - Let  $e = [1, \dots, 1]^T$ , i.e. the column vector of all ones. Then  $L(G) \cdot e = 0$ .
  - The eigenvalues of  $L(G)$  are nonnegative:
    - $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$
  - The number of connected components of  $G$  is equal to the number of  $\lambda_i$  equal to 0.
  - **Definition:**  $\lambda_2(L(G))$  is the algebraic connectivity of  $G$ 
    - The magnitude of  $\lambda_2$  measures connectivity
    - In particular,  $\lambda_2 \neq 0$  if and only if  $G$  is connected.

4/15/2004 CS267, Yelick 38

### Spectral Bisection Algorithm

- Spectral Bisection Algorithm:**
  - Compute eigenvector  $v_2$  corresponding to  $\lambda_2(L(G))$
  - For each node  $n$  of  $G$ 
    - if  $v_2(n) < 0$  put node  $n$  in partition  $N^-$
    - else put node  $n$  in partition  $N^+$
- Why does this make sense? First reasons...**
  - Theorem 2 (Fiedler, 1975):** Let  $G$  be connected, and  $N^-$  and  $N^+$  defined as above. Then  $N^-$  is connected. If no  $v_2(n) = 0$ , then  $N^+$  is also connected. (proof on web page)
  - Recall  $\lambda_2(L(G))$  is the algebraic connectivity of  $G$
  - Theorem 3 (Fiedler):** Let  $G_1(N_1, E_1)$  be a subgraph of  $G(N, E)$ , so that  $G_1$  is "less connected" than  $G$ . Then  $\lambda_2(L(G_1)) \leq \lambda_2(L(G))$ , i.e. the algebraic connectivity of  $G_1$  is less than or equal to the algebraic connectivity of  $G$ . (proof on web page)

4/15/2004 CS267, Yelick 40

### Motivation for Spectral Bisection (recap)

- Vibrating string has **modes of vibration**, or **harmonics**
- Modes computable as follows
  - Model string as masses connected by springs (a 1D mesh)
  - Write down  $F=ma$  for coupled system, get matrix  $A$
  - Eigenvalues and eigenvectors of  $A$  are frequencies and shapes of modes
- Label nodes by whether mode - or + to get  $N^-$  and  $N^+$
- Same idea for other graphs (eg planar graph ~ trampoline)

Modes of a Vibrating String

4/15/2004 41

### Details for Vibrating String Analogy

- Force on mass  $j = k*[x(j-1) - x(j)] + k*[x(j+1) - x(j)]$   
 $= -k*[-x(j-1) + 2*x(j) - x(j+1)]$
- $F=ma$  yields  $m*x''(j) = -k*[-x(j-1) + 2*x(j) - x(j+1)]$  (\*)
- Writing (\*) for  $j=1,2,...,n$  yields

$$m \cdot \frac{d^2}{dx^2} \begin{pmatrix} x(1) \\ x(2) \\ \dots \\ x(j) \\ \dots \\ x(n) \end{pmatrix} = -k \cdot \begin{pmatrix} 2x(1) - x(2) \\ -x(1) + 2x(2) - x(3) \\ \dots \\ -x(j-1) + 2x(j) - x(j+1) \\ \dots \\ 2x(n-1) - x(n) \end{pmatrix} = -k \cdot \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & & \dots & & \\ & & & -1 & 2 & -1 \\ & & & & & \dots \\ & & & & & & -1 & 2 \end{pmatrix} \cdot \begin{pmatrix} x(1) \\ x(2) \\ \dots \\ x(j) \\ \dots \\ x(n) \end{pmatrix} = -k \cdot L \cdot \begin{pmatrix} x(1) \\ x(2) \\ \dots \\ x(j) \\ \dots \\ x(n) \end{pmatrix}$$

**$(-m/k) x'' = L \cdot x$**

Vibrating Mass Spring System

4/15/2004 42

### Details for Vibrating String (continued)

- $(-m/k) x'' = L \cdot x$ , where  $x = [x_1, x_2, \dots, x_n]^T$
- Seek solution of form  $x(t) = \sin(\alpha \cdot t) \cdot x_0$ 
  - $L \cdot x_0 = (m/k) \cdot \alpha^2 \cdot x_0 = \lambda \cdot x_0$
  - For each integer  $i$ , get  $\lambda = 2^2(1 - \cos(i\pi/(n+1)))$ ,  $x_0 = \begin{pmatrix} \sin(1 \cdot i \cdot \pi / (n+1)) \\ \sin(2 \cdot i \cdot \pi / (n+1)) \\ \dots \\ \sin(n \cdot i \cdot \pi / (n+1)) \end{pmatrix}$
- Thus  $x_0$  is a sine curve with frequency proportional to  $i$
- Thus  $\alpha^2 = 2^2 \cdot k/m \cdot (1 - \cos(i\pi/(n+1)))$  or  $\alpha = \text{sqrt}(k/m) \cdot \pi \cdot i / (n+1)$
- $L = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & & \dots & & \\ & & & -1 & 2 \end{pmatrix}$  not quite  $L(1D \text{ mesh})$ , but we can fix that ...

4/15/2004 CS267, Yelick 43

### Motivation for Spectral Bisection

- Vibrating string has **modes of vibration**, or **harmonics**
- Modes computable as follows
  - Model string as masses connected by springs (a 1D mesh)
  - Write down  $F=ma$  for coupled system, get matrix  $A$
  - Eigenvalues and eigenvectors of  $A$  are frequencies and shapes of modes
- Label nodes by whether mode - or + to get  $N^-$  and  $N^+$
- Same idea for other graphs (eg planar graph ~ trampoline)

"Vibrating String" for Spectral Bisection

4/15/2004 CS267, Yelick 44

### Eigenvectors of L(1D mesh)

Graph: Partitioning a Chain; n=40.

4/15/2004 CS267, Yelick 45



**Multilevel Kernighan-Lin**

- Coarsen graph and expand partition using maximal matchings
- Improve partition using Kernighan-Lin

4/15/2004 CS267, Yelick 52

**Maximal Matching**

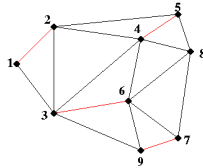
- Definition: A matching of a graph  $G(N,E)$  is a subset  $E_m$  of  $E$  such that no two edges in  $E_m$  share an endpoint
- Definition: A maximal matching of a graph  $G(N,E)$  is a matching  $E_m$  to which no more edges can be added and remain a matching
- A simple greedy algorithm computes a maximal matching:

```

let  $E_m$  be empty
mark all nodes in  $N$  as unmatched
for  $i = 1$  to  $|N|$  ... visit the nodes in any order
  if  $i$  has not been matched
    mark  $i$  as matched
    if there is an edge  $e=(i,j)$  where  $j$  is also unmatched,
      add  $e$  to  $E_m$ 
      mark  $j$  as matched
    endif
  endif
endfor
    
```

4/15/2004 CS267, Yelick 53

**Maximal Matching: Example**



4/15/2004 CS267, Yelick 54

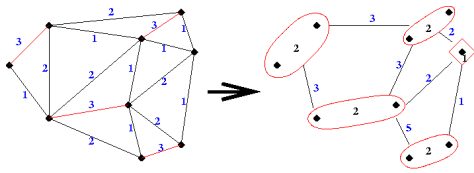
**Coarsening using a maximal matching**

- 1) Construct a maximal matching  $E_m$  of  $G(N,E)$
  - for all edges  $e=(j,k)$  in  $E_m$  2) collapse matches nodes into a single one
  - Put node  $n(e)$  in  $N_c$
  - $W(n(e)) = W(j) + W(k)$  ... gray statements update node/edge weights
  - for all nodes  $n$  in  $N$  not incident on an edge in  $E_m$  3) add unmatched nodes
  - Put  $n$  in  $N_c$  ... do not change  $W(n)$
  - ... Now each node  $r$  in  $N$  is "inside" a unique node  $n(r)$  in  $N_c$
  - ... 4) Connect two nodes in  $N_c$  if nodes inside them are connected in  $E$
  - for all edges  $e=(j,k)$  in  $E_m$
  - for each other edge  $e'=(j,r)$  in  $E$  incident on  $j$
  - Put edge  $ee = (n(e),n(r))$  in  $E_c$
  - $W(ee) = W(e')$
  - for each other edge  $e'=(r,k)$  in  $E$  incident on  $k$
  - Put edge  $ee = (n(r),n(e))$  in  $E_c$
  - $W(ee) = W(e')$
- If there are multiple edges connecting two nodes in  $N_c$ , collapse them, adding edge weights

4/15/2004 CS267, Yelick 55

**Example of Coarsening**

How to coarsen a graph using a maximal matching



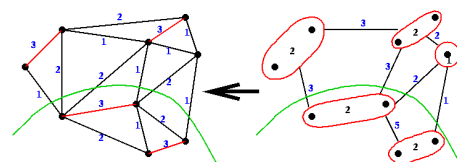
$G = (N, E)$   
 $E_m$  is shown in red  
 Edge weights shown in blue  
 Node weights are all one

$G_c = (N_c, E_c)$   
 $N_c$  is shown in red  
 Edge weights shown in blue  
 Node weights shown in black

4/15/2004 CS267, Yelick 56

**Expanding a partition of  $G_c$  to a partition of  $G$**

Converting a coarse partition to a fine partition



Partition shown in green

4/15/2004 CS267, Yelick 57

### Multilevel Spectral Bisection

- Coarsen graph and expand partition using maximal independent sets
- Improve partition using Rayleigh Quotient Iteration

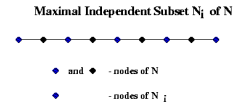
4/15/2004 CS267, Yelick 58

### Maximal Independent Sets

- Definition: An independent set of a graph  $G(N,E)$  is a subset  $N_i$  of  $N$  such that no two nodes in  $N_i$  are connected by an edge
- Definition: A maximal independent set of a graph  $G(N,E)$  is an independent set  $N_i$  to which no more nodes can be added and remain an independent set
- A simple greedy algorithm computes a maximal independent set:
 

```

let  $N_i$  be empty
for  $k = 1$  to  $|N|$  ... visit the nodes in any order
    if node  $k$  is not adjacent to any node already in  $N_i$ 
        add  $k$  to  $N_i$ 
    endif
endfor
            
```



4/15/2004 CS267, Yelick 59

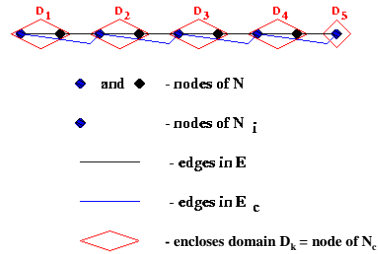
### Coarsening using Maximal Independent Sets

- ... Build "domains"  $D(k)$  around each node  $k$  in  $N_i$  to get nodes in  $N_c$
- ... Add an edge to  $E_c$  whenever it would connect two such domains
- $E_c =$  empty set
- for all nodes  $k$  in  $N_i$ 
  - $D(k) = \{k\}$ , empty set
  - ... first set contains nodes in  $D(k)$ , second set contains edges in  $D(k)$
- unmark all edges in  $E$
- repeat
  - choose an unmarked edge  $e = (k,j)$  from  $E$
  - if exactly one of  $k$  and  $j$  (say  $k$ ) is in some  $D(m)$ 
    - mark  $e$
    - add  $j$  and  $e$  to  $D(m)$
  - else if  $k$  and  $j$  are in two different  $D(m)$ 's (say  $D(m_i)$  and  $D(m_j)$ )
    - mark  $e$
    - add edge  $(m_i, m_j)$  to  $E_c$
  - else if both  $k$  and  $j$  are in the same  $D(m)$ 
    - mark  $e$
    - add  $e$  to  $D(m)$
  - else
    - leave  $e$  unmarked
- endif
- until no unmarked edges

4/15/2004 CS267, Yelick 60

### Example of Coarsening

#### Computing $G_c$ from $G$



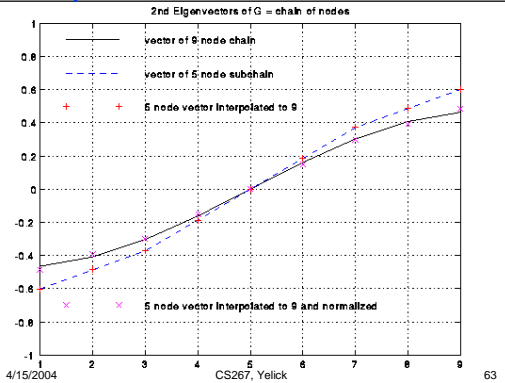
4/15/2004 CS267, Yelick 61

### Expanding a partition of $G_c$ to a partition of $G$

- Need to convert an eigenvector  $v_c$  of  $L(G_c)$  to an approximate eigenvector  $v$  of  $L(G)$
- Use interpolation:
  - For each node  $j$  in  $N$ 
    - if  $j$  is also a node in  $N_c$ , then
      - $v(j) = v_c(j)$  ... use same eigenvector component
    - else
      - $v(j) =$  average of  $v_c(k)$  for all neighbors  $k$  of  $j$  in  $N_c$
  - endif

4/15/2004 CS267, Yelick 62

### Example: 1D mesh of 9 nodes

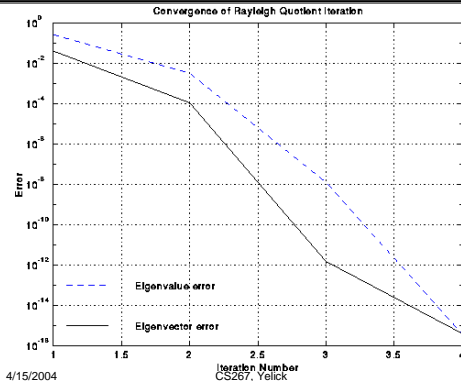


4/15/2004 CS267, Yelick 63

**Improve eigenvector: Rayleigh Quotient Iteration**

$j = 0$   
 pick starting vector  $v(0)$  ... from expanding  $v_c$   
 repeat  
    $j=j+1$   
    $r(j) = v^T(j-1) * L(G) * v(j-1)$   
   ...  $r(j) = \text{Rayleigh Quotient of } v(j-1)$   
   ... = good approximate eigenvalue  
    $v(j) = (L(G) - r(j)*I)^{-1} * v(j-1)$   
   ... expensive to do exactly, so solve approximately  
   ... using an iteration called SYMMLQ,  
   ... which uses matrix-vector multiply (no surprise)  
    $v(j) = v(j) / || v(j) ||$  ... normalize  $v(j)$   
 until  $v(j)$  converges  
 ... Convergence is very fast: cubic

**Example of convergence for 1D mesh**



**Available Implementations**

- Multilevel Kernighan/Lin
  - METIS ([www.cs.umn.edu/~metis](http://www.cs.umn.edu/~metis))
  - ParMETIS - parallel version
- Multilevel Spectral Bisection
  - S. Barnard and H. Simon, "A fast multilevel implementation of recursive spectral bisection ...", Proc. 6th SIAM Conf. On Parallel Processing, 1993
  - Chaco ([www.cs.sandia.gov/CRF/papers\\_chaco.html](http://www.cs.sandia.gov/CRF/papers_chaco.html))
- Hybrids possible
  - Ex: Using Kernighan/Lin to improve a partition from spectral bisection

**Comparison of methods**

- Compare only methods that use edges, not nodal coordinates
  - CS267 webpage and KK95a (see below) have other comparisons
- Metrics
  - Speed of partitioning
  - Number of edge cuts
  - Other application dependent metrics
- Summary
  - No one method best
  - Multi-level Kernighan/Lin fastest by far, comparable to Spectral in the number of edge cuts
    - [www-users.cs.umn.edu/~karypis/memis/publications/mail.html](http://www-users.cs.umn.edu/~karypis/memis/publications/mail.html)
    - see publications KK95a and KK95b
  - Spectral give much better cuts for some applications
    - Ex: image segmentation
    - [www.cs.berkeley.edu/~jshi/Grouping/overview.html](http://www.cs.berkeley.edu/~jshi/Grouping/overview.html)
    - see "Normalized Cuts and Image Segmentation"

**Number of edges cut for a 64-way partition**

For Multilevel Kernighan/Lin, as implemented in METIS (see KK95a)

Graph	# of Nodes	# of Edges	# Edges cut for 64-way partition	Expected # cuts for 2D mesh	Expected # cuts for 3D mesh	Description
144	144649	1074393	88806	6427	31805	3D FE Mesh
4ELT	15606	45878	2965	2111	7208	2D FE Mesh
ADD32	4960	9462	675	1190	3357	32 bit adder
AUTO	448695	3314611	194436	11320	67647	3D FE Mesh
BBMAT	38744	993481	55753	3326	13215	2D Stiffness M.
FINANS12	74752	261120	11388	4620	20481	Lin. Prog.
LHR10	10672	209093	58784	1746	5595	Chem. Eng.
MAP1	267241	334931	1388	8736	47887	Highway Net.
MEMPLUS	17758	54196	17894	2252	7856	Memory circuit
SHYY161	76480	152002	4365	4674	20796	Navier-Stokes
TORSO	201142	1479989	117997	7579	39623	3D FE Mesh

Expected # cuts for 64-way partition of 2D mesh of n nodes  
 $n^{1/2} + 2*(n/2)^{1/2} + 4*(n/4)^{1/2} + \dots + 32*(n/32)^{1/2} \sim 17 * n^{1/2}$

Expected # cuts for 64-way partition of 3D mesh of n nodes =  
 $n^{2/3} + 2*(n/2)^{2/3} + 4*(n/4)^{2/3} + \dots + 32*(n/32)^{2/3} \sim 11.5 * n^{2/3}$

**Speed of 256-way partitioning (from KK95a)**

Partitioning time in seconds

Graph	# of Nodes	# of Edges	Multilevel Spectral Bisection	Multilevel Kernighan/Lin	Description
144	144649	1074393	607.3	48.1	3D FE Mesh
4ELT	15606	45878	25.0	3.1	2D FE Mesh
ADD32	4960	9462	18.7	1.6	32 bit adder
AUTO	448695	3314611	2214.2	179.2	3D FE Mesh
BBMAT	38744	993481	474.2	25.5	2D Stiffness M.
FINANS12	74752	261120	311.0	18.0	Lin. Prog.
LHR10	10672	209093	142.6	8.1	Chem. Eng.
MAP1	267241	334931	850.2	44.8	Highway Net.
MEMPLUS	17758	54196	117.9	4.3	Memory circuit
SHYY161	76480	152002	130.0	10.1	Navier-Stokes
TORSO	201142	1479989	1053.4	63.9	3D FE Mesh

Kernighan/Lin much faster than Spectral Bisection!

### Coordinate-Free Partitioning: Summary

- Several techniques for partitioning without coordinates
  - Breadth-First Search – simple, but not great partition
  - Kernighan-Lin – good corrector given reasonable partition
  - Spectral Method – good partitions, but slow
- Multilevel methods
  - Used to speed up problems that are too large/slow
  - Coarsen, partition, expand, improve
  - Can be used with K-L and Spectral methods and others
- Speed/quality
  - For load balancing of grids, multi-level K-L probably best
  - For other partitioning problems (vision, clustering, etc.) spectral may be better
  - Good software available

4/15/2004

CS267, Yelick

70

### Is Graph Partitioning a Solved Problem?

- Myths of partitioning due to Bruce Hendrickson
  - ➔ 1. Edge cut = communication cost
  - ➔ 2. Simple graphs are sufficient
  - ➔ 3. Edge cut is the right metric
  4. Existing tools solve the problem
  5. Key is finding the right partition
  6. Graph partitioning is a solved problem
- Slides and myths based on Bruce Hendrickson's: "Load Balancing Myths, Fictions & Legends"

4/15/2004

CS267, Yelick

71

### Myth 1: Edge Cut = Communication Cost

- Myth1: The edge-cut deceit  
edge-cut = communication cost
- Not quite true:
  - #vertices on boundary is actual communication volume
    - Do not communicate same node value twice
  - Cost of communication depends on # of messages too ( $\alpha$  term)
  - Congestion may also affect communication cost
- Why is this OK for most applications?
  - Mesh-based problems match the model: cost is ~ edge cuts
  - Other problems (data mining, etc.) do not

4/15/2004

CS267, Yelick

72

### Myth 2: Simple Graphs are Sufficient

- Graphs often used to encode data dependencies
  - Do X before doing Y
- Graph partitioning determines data partitioning
  - Assumes graph nodes can be evaluated in parallel
  - Communication on edges can also be done in parallel
  - Only dependence is between sweeps over the graph
- More general graph models include:
  - Hypergraph: nodes are computation, edges are communication, but connected to a set ( $\geq 2$ ) of nodes
  - Bipartite model: use bipartite graph for directed graph
  - Multi-object, Multi-Constraint model: use when single structure may involve multiple computations with differing costs

4/15/2004

CS267, Yelick

73

### Myth 3: Partition Quality is Paramount

- When structure are changing dynamically during a simulation, need to partition dynamically
  - Speed may be more important than quality
  - Partitioner must run fast in parallel
  - Partition should be incremental
    - Change minimally relative to prior one
  - Must not use too much memory
- Example from Touheed, Selwood, Jimack and Bersins
  - 1 M elements with adaptive refinement on SGI Origin
  - Timing data for different partitioning algorithms:
    - Repartition time from 3.0 to 15.2 secs
    - Migration time : 17.8 to 37.8 secs
    - Solve time: 2.54 to 3.11 secs

4/15/2004

CS267, Yelick

74

### References

- Details of all proofs on Jim Demmel's 267 web page
- A. Pothen, H. Simon, K.-P. Liou, "Partitioning sparse matrices with eigenvectors of graphs", SIAM J. Mat. Anal. Appl. 11:430-452 (1990)
- M. Fiedler, "Algebraic Connectivity of Graphs", Czech. Math. J., 23:298-305 (1973)
- M. Fiedler, Czech. Math. J., 25:619-637 (1975)
- B. Parlett, "The Symmetric Eigenproblem", Prentice-Hall, 1980
- [www.cs.berkeley.edu/~ruhe/lantplht/lantplht.html](http://www.cs.berkeley.edu/~ruhe/lantplht/lantplht.html)
- [www.netlib.org/laso](http://www.netlib.org/laso)

4/15/2004

CS267, Yelick

75

## Summary

---

- Partitioning with nodal coordinates:
  - Inertial method
  - Projection onto a sphere
  - Algorithms are efficient
  - Rely on graphs having nodes connected (mostly) to "nearest neighbors" in space
- Partitioning without nodal coordinates:
  - Breadth-First Search – simple, but not great partition
  - Kernighan-Lin – good corrector given reasonable partition
  - Spectral Method – good partitions, but slow
- Today:
  - Spectral methods revisited
  - Multilevel methods

4/15/2004

CS267, Yelick

76