Definition of Graph Partitioning

- Given a graph \( G = (N, E, W_N, W_E) \)
  - \( N \) = nodes (or vertices),
  - \( W_N \) = node weights
  - \( E \) = edges
  - \( W_E \) = 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 \( N = N_1 \cup N_2 \cup \ldots \cup N_P \) such that
  - The sum of the node weights in each \( N_i \) is "about the same"
  - The sum of all edge weights of edges connecting all different pairs \( N_i \) and \( N_j \) is minimized
- Ex: balance the work load, while minimizing communication
- Special case of \( N = N_1 \cup N_2 \): Graph Bisection

Outline of Graph Partitioning Lecture

- Review definition of Graph Partitioning problem
- Overview of heuristics
- Partitioning with Nodal Coordinates
  - Ex: In finite element models, node at point in \((x, y)\) or \((x, y, z)\) space
- Partitioning without Nodal Coordinates
  - Ex: In model of WWW, nodes are web pages
- Multilevel Acceleration
  - BIG IDEA, appears often in scientific computing
- Comparison of Methods and Applications
- Beyond Graph Partitioning: Hypergraphs
Some Applications

- Telephone network design
  - Original application, algorithm due to Kernighan
- Load Balancing while Minimizing Communication
- Sparse Matrix times Vector Multiplication (SpMV)
  - Solving PDEs
  - \( N = \{1, \ldots, n\}, \quad (j,k) \in E \text{ if } A(j,k) \text{ nonzero,} \)
  - \( W(j) = \# \text{nonzeros in row } j, \quad W(j,k) = 1 \)
- VLSI Layout
  - \( N = \{\text{units on chip}\}, \quad E = \{\text{wires}\}, \quad W(j,k) = \text{wire length} \)
- 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
- Image Segmentation

Sparse Matrix Vector Multiplication \( y = y + A^\top x \)

Partitioning a Sparse Symmetric Matrix

Cost of Graph Partitioning

- Many possible partitionings to search
- Just to divide in 2 parts there are:
  \[ n \choose n/2 = n!/(n/2)! \sim (2/(\pi n))^{1/2} \cdot 2^n \text{ possibilities} \]

- Choosing optimal partitioning is NP-complete
  - (NP-complete = we can prove it is a 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

Outline of Graph Partitioning Lectures

- Review definition of Graph Partitioning problem
- Overview of heuristics
  - Partitioning with Nodal Coordinates
    - Ex: In finite element models, node at point in \((x,y)\) or \((x,y,z)\) space
  - Partitioning without Nodal Coordinates
    - Ex: In model of WWW, nodes are web pages
  - Multilevel Acceleration
    - BIG IDEA, appears often in scientific computing
  - Comparison of Methods and Applications
  - Beyond Graph Partitioning: Hypergraphs
**First Heuristic: Repeated Graph Bisection**

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

---

**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 $N_2$

\[ G = (N, E), \text{Nodes } N \text{ and Edges } E \]
\[ E_s = \text{green edges or blue edges} \]
\[ N_s = \text{red vertices} \]

- 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 $N_s$
  - $|E_s| \leq d \cdot |N_s|$ 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
    - $A(j,k) =$ # times keyword $j$ appears in URL $k$
- Multilevel acceleration (BIG IDEA)
  - Approximate problem by "coarse graph," do so recursively

---

**Outline of Graph Partitioning Lectures**

- Review definition of Graph Partitioning problem
- Overview of heuristics
- **Partitioning with Nodal Coordinates**
  - Ex: In finite element models, node at point in $(x,y)$ or $(x,y,z)$ space
  - Partitioning without Nodal Coordinates
  - Ex: In model of WWW, nodes are web pages
- Multilevel Acceleration
  - BIG IDEA, appears often in scientific computing
- Comparison of Methods and Applications
- Beyond Graph Partitioning: Hypergraphs
### Nodal Coordinates: How Well Can We Do?

- A planar graph can be drawn in plane without edge crossings
- Ex: m x m grid of m^2 nodes: ∃ vertex separator N_s with |N_s| = m = |N|\(^1/2\) (see earlier slide for m=5)
- **Theorem** (Tarjan, Lipton, 1979): If G is planar, ∃ N_s such that
  - N = N_1 \cup N_2 \cup N_2 is a partition,
  - |N_1| <= 2/3 |N| and |N_2| <= 2/3 |N|
  - |N_s| <= (8 * |N|)^1/2

• Theorem motivates intuition of following algorithms

### 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 a line L⊥ 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 \(\perp\) 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 \(Sbar = median(S_1,\ldots,S_n)\)
  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, 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

\[
\sum_j (\text{length of j-th green line})^2 = S_j^2 (x_j-xbar)^2 + (y_j-ybar)^2 - (-b^2(x_j-xbar) + a^2(y_j-ybar))^2
\]

Pythagorean Theorem

\[
= a^2 S_j^2 (x_j-xbar)^2 + b^2 \sum_j (x_j-xbar) + b^2 \sum_j (y_j-ybar)^2
\]

Minimized by choosing

\[
(xbar,ybar) = (\bar{x}, \bar{y}) / n = \text{center of mass} (a,b) = \text{eigenvector of smallest eigenvalue of}
\]

\[
\begin{bmatrix}
 X_1 & X_2 & X_3
\end{bmatrix}
\]
Nodal Coordinates: Random Spheres

- Generalize nearest neighbor idea of a planar graph to higher dimensions
  - Any graph can fit in 3D without edge crossings
  - Capture intuition of planar graphs of being connected to "nearest neighbors" but in higher than 2 dimensions
- For intuition, consider 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 \( n^2 = |N| = n^3 \) = \( O(|E|^{2/3}) \) edges
- For the general graphs
  - Need a notion of "well-shaped" like mesh

Random Spheres: Well Shaped Graphs

- Approach due to Miller, Teng, Thurston, Vavasis
- Def: A k-ply neighborhood system in d dimensions is a set \( \{D_1, \ldots, D_n\} \) of closed disks in \( \mathbb{R}^d \) such that no point in \( \mathbb{R}^d \) 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 \( \{D_1, \ldots, D_n\} \): There is a node for each \( D_j \) 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 \( d = 2 \), similar to Lipton/Tarjan
- Algorithm:
  - Choose a sphere \( S \) in \( \mathbb{R}^d \)
  - Edges that \( S \) "cuts" form edge separator \( E_S \)
  - Build \( N_S \) from \( E_S \)
  - Choose \( S \) "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
  - \( p = (x, y) \)
  - \( p' = \frac{(2x, 2y, x^2 + y^2 - 1)}{x^2 + y^2 + 1} \)
  - Similar in higher dimensions
Choosing a Random Sphere

- Do stereographic projection from $\mathbb{R}^d$ to sphere $S$ in $\mathbb{R}^{d+1}$
- 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 $(0,\ldots,0, r)$ for some $r$
  - Dilate points (unproject, multiply by $((1-r)/(1+r))^{1/2}$, project)
    - this maps centerpoint to origin $(0,\ldots,0)$, spreads points around $S$
- Pick a random plane through origin
- Intersection of plane and sphere $S$ is “circle”
- Unproject circle
  - yields desired circle $C$ in $\mathbb{R}^d$
- Create $N_s$: $j$ belongs to $N_s$ if $\alpha^*D_j$ 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
- Ignores edges, but can be used as good starting guess for subsequent partitioners that do examine edges
- Can do poorly if graph connectivity is not spatial:

- Details at
  - www.cs.berkeley.edu/~demmel/cs267/lecture18/lecture18.html
  - www.cs.ucsb.edu/~gilbert
  - www-bcf.usc.edu/~shanghua/
Outline of Graph Partitioning Lectures

- Review definition of Graph Partitioning problem
- Overview of heuristics
- Partitioning with Nodal Coordinates
  - Ex: In finite element models, node at point in (x,y) or (x,y,z) space
- Partitioning without Nodal Coordinates
  - Ex: In model of WWW, nodes are web pages
- Multilevel Acceleration
  - BIG IDEA, appears often in scientific computing
- Comparison of Methods and Applications
- Beyond Graph Partitioning: Hypergraphs

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

BFS partition of a 2D Mesh using center as root:

N1 = levels 0, 1, 2, 3
N2 = levels 4, 5, 6

Breadth First Search (details)

- Queue (First In First Out, or FIFO)
  - Enqueue(x,Q) adds x to back of Q
  - x = Dequeue(Q) removes x from front of Q
- Compute Tree T(N, E)

BFS partitioning heuristic

\[ N = N_1 \cup N_2, \text{ where} \]
\[ N_1 = \{\text{nodes at level } \leq L\}, \]
\[ N_2 = \{\text{nodes at level } > L\} \]
Choose L so \( |N_1| \) close to \( |N_2| \)

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?)
Coordinate-Free: Kernighan/Lin

- Take an initial partition and iteratively improve it
  - Kernighan/Lin (1970), cost = $O(|N|^2)$ 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|
- $T = \text{cost}(A,B) = \sum \{W(e) \text{ where } e \text{ connects nodes in } A \text{ and } B\}$
- Consider swapping $X$ and $Y$ if it decreases cost:
  - $\text{newA} = (A - X) \cup Y$ and $\text{newB} = (B - Y) \cup X$
  - $\text{newT} = \text{cost}(\text{newA}, \text{newB}) < T = \text{cost}(A,B)$
- Need to compute $\text{newT}$ efficiently for many possible $X$ and $Y$, choose smallest (best)

Kernighan/Lin: Preliminary Definitions

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

Kernighan/Lin Algorithm

- Compute $T = \text{cost}(A, B)$ for initial $A, B$
- Repeat
  - $\text{One pass greedily computes } |N|/2 \text{ possible } X, Y \text{ to swap, picks best}$
  - $\text{Compute costs } D(n) \text{ for all } n \in N$
  - $\text{Unmark all nodes in } N$
  - While there are unmarked nodes
    - $\text{Find an unmarked pair } (a,b) \text{ maximizing } \text{gain}(a,b)$
    - $\text{Mark } a \text{ and } b \text{ (but do not swap them)}$
    - $\text{Update } D(n) \text{ for all unmarked } n,$
      as though $a$ and $b$ had been swapped
    - $\text{Endwhile}$
  - $\text{At this point we have computed a sequence of pairs}$
    - $(a_1,b_1), \ldots, (a_k,b_k)$ and gains $\text{gain}(1), \ldots, \text{gain}(k)$
    - $\text{where } k = |N|/2; \text{ numbered in the order in which we marked them}$
- Pick $m$ maximizing $\text{Gain} = \sum_{k=1}^{m} \text{gain}(k)$
  - $\text{Gain is reduction in cost from swapping } (a_1,b_1) \text{ through } (a_m,b_m)$
- If $\text{Gain} > 0$ then ...
  - $\text{it is worth swapping}$
  - $\text{Update } \text{newA} = A - \{a_1,\ldots,a_m\} \cup \{b_1,\ldots,b_m\}$
  - $\text{Update } \text{newB} = B - \{b_1,\ldots,b_m\} \cup \{a_1,\ldots,a_m\}$
  - $\text{Update } T = T - \text{Gain}$
- $\text{Endif}$
- $\text{Until } \text{Gain} <= 0$

Comments on Kernighan/Lin Algorithm

- Most expensive line shown in red, $O(n^3)$
- 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|=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 $2^{-|N|/30}$
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 ...

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)

Basic Definitions

• Definition: The incidence matrix In(G) of a graph G(N,E) is an INI by IEI 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 INI by INI symmetric matrix, with one row and column for each node. It is defined by
    • L(G) (i,i) = degree of node i (number of incident edges)
    • L(G) (i,j) = -1 if i ≠ j and there is an edge (i,j)
    • L(G) (i,j) = 0 otherwise

Example of In(G) and L(G) for Simple Meshes
Properties of Laplacian Matrix

- **Theorem 1:** Given G, L(G) has the following properties
  (proof on 1996 CS267 web page)
  - L(G) is symmetric.
  - This means the eigenvalues of L(G) are real and its eigenvectors are real and orthogonal.
  - ln(G) * (ln(G))^T = L(G)
  - The eigenvalues of L(G) are nonnegative:
    - 0 = \lambda_1 \leq \lambda_2 \leq \ldots \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.

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 1996 CS267 web page)
    - Recall \lambda_2(L(G)) is the algebraic connectivity of G
    - **Theorem 3 (Fiedler):** Let G_1(N,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 1996 CS267 web page)

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

- **Lower Frequency Mode:**
  - First frequency of vibration
- **Second Frequency Mode:**
  - Second frequency of vibration
- **Third Frequency Mode:**
  - Third frequency of vibration
**Details for Vibrating String Analogy**

- Force on mass \( j \) = \( k^*\{x(j-1) - x(j)\} + k^*\{x(j+1) - x(j)\} \)
- \( F=ma \) yields \( m^*x''(j) = -k^*\{-x(j-1) + 2x(j) - x(j+1)\} \)
- Writing (*) for \( j=1,2,\ldots,n \) yields

\[
\begin{pmatrix}
  x(1) \\
  2x(1) - x(2) \\
  2x(2) - x(1) \\
  \vdots \\
  x(n)
\end{pmatrix}
\begin{pmatrix}
  2 & -1 & 0 & \cdots & 0 \\
  -1 & 2 & -1 & \cdots & 0 \\
  0 & -1 & 2 & \cdots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & -1 & 2 \\
\end{pmatrix}
\begin{pmatrix}
  x(1) \\
  x(2) \\
  x(3) \\
  \vdots \\
  x(n)
\end{pmatrix}
\]

\[\Rightarrow \frac{(-m/k)}{m} x'' = L^*x\]

**Details for Vibrating String (continued)**

- \(-m/k\) \( x'' = L^*x \), where \( x = [x_1, x_2, \ldots, x_n]^T \)
- Seek solution of form \( x(t) = \sin(\alpha t) \cdot x_0 \)
- \( L x_0 = \frac{(m/k)}{\lambda} x_0 \)
- For each integer \( i \), get \( \lambda_i = 2^2(1-\cos(i\pi/(n+1))) \), \( x_i = \sin(1^i\pi/(n+1)) \cdot \sin(2^i\pi/(n+1)) \cdot \sin(3^i\pi/(n+1)) \cdot \cdots \cdot \sin(n^i\pi/(n+1)) \)
- Thus \( x_0 \) is a sine curve with frequency proportional to \( i \)
- Thus \( \alpha_i = \sqrt{2^2(k/m)(1-\cos(i\pi/(n+1)))} \approx (k/m)^{1/2} \cdot \pi \cdot n/(n+1) \)
- \( L = \begin{pmatrix}
  2 & -1 & \cdots & \cdots & -1 \\
  -1 & 2 & -1 & \cdots & 0 \\
  \vdots & \vdots & \ddots & \ddots & \vdots \\
  \cdots & \cdots & \cdots & -1 & 2 \\
  -1 & 0 & \cdots & 0 & 2
\end{pmatrix} \)

\[\text{not quite Laplacian of 1D mesh, but we can fix that...}\]

**Eigenvectors of \( L(1D \text{ mesh}) \)**

- Eigenvector 1 (all ones)
- Eigenvector 2
- Eigenvector 3
Computing $v_2$ and $\lambda_2$ of $L(G)$ using Lanczos

- Given any $n$-by-$n$ symmetric matrix $A$ (such as $L(G)$), Lanczos computes a $k$-by-$k$ “approximation” $T$ by doing $k$ matrix-vector products, $k \ll n$.

Choose an arbitrary starting vector $r$

- $b(0) = \|r\|$

repeat

- $j := j + 1$
- $q(j) = r/b(j-1)$  
  ... scale a vector (BLAS1)
- $r = A\cdot q(j)$  
  ... matrix vector multiplication, the most expensive step
- $r = r - b(j-1)\cdot v(j-1)$  
  ... “axpy”, or scalar*vector + vector (BLAS1)
- $a(j) = v(j)\cdot r$  
  ... dot product (BLAS1)
- $r = r - a(j)\cdot v(j)$  
  ... “axpy” (BLAS1)
- $b(j) = \|r\|$  
  ... compute vector norm (BLAS1)

until convergence  
  ... details omitted

$T = \begin{bmatrix} a(1) & b(1) \\ b(1) & a(2) \\ \vdots & \vdots \\ b(k-1) & a(k-1) \\ b(k-1) & a(k) \end{bmatrix}$

- Approximate $A$’s eigenvalues/vectors using $T$’s

Spectral Bisection: Summary

- Laplacian matrix represents graph connectivity
- Second eigenvector gives a graph bisection
  - Roughly equal “weights” in two parts
  - Weak connection in the graph will be separator
- 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
- Have we made progress?
  - The first matrix-vector multiplies are slow, but use them to learn how to make the rest faster
Outline of Graph Partitioning Lectures

- Review definition of Graph Partitioning problem
- Overview of heuristics
- Partitioning with Nodal Coordinates
  - Ex: In finite element models, node at point in (x,y) or (x,y,z) space
- Partitioning without Nodal Coordinates
  - Ex: In model of WWW, nodes are web pages
- Multilevel Acceleration
  - BIG IDEA, appears often in scientific computing
- Comparison of Methods and Applications
- Beyond Graph Partitioning: Hypergraphs

Introduction to Multilevel Partitioning

- If we want to partition G(N,E), but it is too big to do efficiently, what can we do?
  - 1) Replace G(N,E) by a coarse approximation Gc(Nc,Ec), and partition Gc instead
  - 2) Use partition of Gc to get a rough partitioning of G, and then iteratively improve it
- What if Gc still too big?
  - Apply same idea recursively

Multilevel Partitioning - High Level Algorithm

(N+N-) = Multilevel_Partition(N, E)
...recursive partitioning routine returns N+ and N- where N = N+ U N-
if |N| is small
(1) Partition G = (N,E) directly to get N = N+ U N-
    Return (N+, N-)
else
(2) Coarsen G to get an approximation Gc = (Nc, Ec)
(3) (Nc+, Nc-) = Multilevel_Partition(Nc, Ec)
(4) Expand (Nc+, Nc-) to a partition (N+, N-) of N
(5) Improve the partition (N+, N-)
    Return (N+, N-)
endif

How do we
- Coarsen?
- Expand?
- Improve?

Multilevel Kernighan-Lin

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

Maximal Matching: Example

![Maximal Matching Example Diagram]

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 shown in black
```

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

Coarsening using a maximal matching (details)

1) Construct a maximal matching E_m of G(N,E)
   for all edges e=(j,k) in E_m
   \[ W(n(e)) = W(j) + W(k) \]  … gray statements update node/edge weights
   for all nodes n in N not incident on an edge in E_m
   \[ W(n) \]  … do not change W(n)

   2) Collapse matched nodes into a single one
   For each other edge e'=(j,r) or (k,r) in E
   Put edge ee' = (n(e),n(r)) in E_c
   W(ee') = W(e')

   3) Add unmatched nodes
   For all edges e=(j,k) in E
   Put n in N_c

   4) Connect two nodes in N_c if nodes inside them are connected in E
   For all edges e''=(j,r) or (k,r) in E
   Put edge ee'' = (n(e),n(r)) in E_c
   W(ee'') = W(e'')

   If there are multiple edges connecting two nodes in N_c, collapse them,... adding edge weights.
Expanding a partition of $G_n$ to a partition of $G$

Converting a coarse partition to a fine partition

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
  ```

Multilevel Spectral Bisection

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

Example of Coarsening

Computing $G_c$ from $G$
Coarsening using Maximal Independent Sets (details)

... 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 $E_c = \emptyset$ 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 $D(m_k)$ and $D(m_j)$) mark $e$ add edge $(m_k, m_j)$ 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

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
      - $v(j) = \text{average of } v_C(k) \text{ for all neighbors } k \text{ of } j \text{ in } N_c$
  - End if

Improve eigenvector: Rayleigh Quotient Iteration

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

Outline of Graph Partitioning Lectures

- Review definition of Graph Partitioning problem
- Overview of heuristics
- Partitioning with Nodal Coordinates
  - Ex: In finite element models, node at point in (x,y) or (x,y,z) space
- Partitioning without Nodal Coordinates
  - Ex: In model of WWW, nodes are web pages
- Multilevel Acceleration
  - BIG IDEA, appears often in scientific computing
- Comparison of Methods and Applications
- Beyond Graph Partitioning: Hypergraphs

Available Implementations

- Multilevel Kernighan/Lin
  - METIS and ParMETIS (glaros.dtc.umn.edu/gkhome/views/metis)
  - SCOTCH and PT-SCOTCH (www.labri.fr/perso/pelegrin/scotch/)
- Multilevel Spectral Bisection
- Hybrids possible
  - Ex: Using Kernighan/Lin to improve a partition from spectral bisection
- Recent package, collection of techniques
  - Zoltan (www.cs.sandia.gov/Zoltan)
- See www.cs.sandia.gov/~bahendr/partitioning.html

Comparison of methods

- Compare only methods that use edges, not nodal coordinates
  - See www-users.cs.umn.edu/~karypis/metis/publications/main.html
  - Ex: Normalized Cuts and Image Segmentation by J. Malik, J. Shi
### Number of edges cut for a 64-way partition, by METIS

<table>
<thead>
<tr>
<th>Graph</th>
<th># of Nodes</th>
<th># of Edges</th>
<th># cuts for 64-way partition</th>
<th>Expected # cuts for 32 mesh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>144</td>
<td>144649</td>
<td>1074393</td>
<td>88806</td>
<td>8427</td>
<td>3D FE Mesh</td>
</tr>
<tr>
<td>4ELT</td>
<td>15606</td>
<td>45878</td>
<td>2965</td>
<td>2111</td>
<td>2D FE Mesh</td>
</tr>
<tr>
<td>ADD32</td>
<td>4960</td>
<td>9462</td>
<td>1190</td>
<td>7208</td>
<td>32 bit adder</td>
</tr>
<tr>
<td>AUTO</td>
<td>446995</td>
<td>3314611</td>
<td>76747</td>
<td>5327</td>
<td>3D FE Mesh</td>
</tr>
<tr>
<td>BBMAT</td>
<td>38754</td>
<td>993481</td>
<td>113520</td>
<td>57647</td>
<td>3D FE Mesh</td>
</tr>
<tr>
<td>FINAN512</td>
<td>74752</td>
<td>261120</td>
<td>11388</td>
<td>4620</td>
<td>3D FE Mesh</td>
</tr>
<tr>
<td>LHR10</td>
<td>19672</td>
<td>209093</td>
<td>59784</td>
<td>20481</td>
<td>3D FE Mesh</td>
</tr>
<tr>
<td>MAP1</td>
<td>267241</td>
<td>334931</td>
<td>8736</td>
<td>74878</td>
<td>2D Stiffness M.</td>
</tr>
<tr>
<td>MEMPLUS</td>
<td>17758</td>
<td>54196</td>
<td>2252</td>
<td>7856</td>
<td>Lin. Prog.</td>
</tr>
<tr>
<td>SHYY161</td>
<td>76480</td>
<td>152002</td>
<td>4365</td>
<td>20796</td>
<td>Memory circuit</td>
</tr>
<tr>
<td>TORSO</td>
<td>201142</td>
<td>1479989</td>
<td>7579</td>
<td>39623</td>
<td>3D FE Mesh</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Graph</th>
<th># of Nodes</th>
<th># of Edges</th>
<th># cuts for 64-way partition</th>
<th>Expected # cuts for 32 mesh</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>144</td>
<td>144649</td>
<td>1074393</td>
<td>88806</td>
<td>8427</td>
<td>3D FE Mesh</td>
</tr>
<tr>
<td>4ELT</td>
<td>15606</td>
<td>45878</td>
<td>2965</td>
<td>2111</td>
<td>2D FE Mesh</td>
</tr>
<tr>
<td>ADD32</td>
<td>4960</td>
<td>9462</td>
<td>1190</td>
<td>7208</td>
<td>32 bit adder</td>
</tr>
<tr>
<td>AUTO</td>
<td>446995</td>
<td>3314611</td>
<td>76747</td>
<td>5327</td>
<td>3D FE Mesh</td>
</tr>
<tr>
<td>BBMAT</td>
<td>38754</td>
<td>993481</td>
<td>113520</td>
<td>57647</td>
<td>3D FE Mesh</td>
</tr>
<tr>
<td>FINAN512</td>
<td>74752</td>
<td>261120</td>
<td>11388</td>
<td>4620</td>
<td>3D FE Mesh</td>
</tr>
<tr>
<td>LHR10</td>
<td>19672</td>
<td>209093</td>
<td>59784</td>
<td>20481</td>
<td>3D FE Mesh</td>
</tr>
<tr>
<td>MAP1</td>
<td>267241</td>
<td>334931</td>
<td>8736</td>
<td>74878</td>
<td>2D Stiffness M.</td>
</tr>
<tr>
<td>MEMPLUS</td>
<td>17758</td>
<td>54196</td>
<td>2252</td>
<td>7856</td>
<td>Lin. Prog.</td>
</tr>
<tr>
<td>SHYY161</td>
<td>76480</td>
<td>152002</td>
<td>4365</td>
<td>20796</td>
<td>Memory circuit</td>
</tr>
<tr>
<td>TORSO</td>
<td>201142</td>
<td>1479989</td>
<td>7579</td>
<td>39623</td>
<td>3D FE Mesh</td>
</tr>
</tbody>
</table>

**Kernighan/Lin much faster than Spectral Bisection!**

### Outline of Graph Partitioning Lectures

- Review definition of Graph Partitioning problem
- Overview of heuristics
- Partitioning with Nodal Coordinates
  - Ex: In finite element models, node at point in (x,y) or (x,y,z) space
- Partitioning without Nodal Coordinates
  - Ex: In model of WWW, nodes are web pages
- Multilevel Acceleration
  - BIG IDEA, appears often in scientific computing
- Comparison of Methods and Applications
- Beyond Graph Partitioning: Hypergraphs

### Beyond simple graph partitioning:

Representing a sparse matrix as a hypergraph

\[
\begin{bmatrix}
x & 0 & x & 0 \\
0 & x & 0 & 0 \\
0 & 0 & 0 & x \\
x & c1 & c2 & c3 \\
c2 & c1 & c2 & c3 \\
c3 & c2 & c3 & c4 \\
c4 & c3 & c4 & c1 \\
\end{bmatrix}
\]
But graph cut is 3!
\[ \Rightarrow \text{Cut size of graph partition may not accurately count communication volume} \]

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

Source vector entries corresponding to \( c_2 \) and \( c_3 \) are needed by both partitions - so total volume of communication is 2.

\[ \Rightarrow \text{Cut size of graph partition may not accurately count communication volume} \]

Two Different 2D Mesh Partitioning Strategies

<table>
<thead>
<tr>
<th>Graph: Cartesian Partitioning</th>
<th>Hypergraph: MeshPart Algorithm [Ucar, Catalyurek, 2010]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total SpMV communication volume = 64</td>
<td>Total SpMV communication volume = 58</td>
</tr>
</tbody>
</table>

Experimental Results: Hypergraph vs. Graph Partitioning

-8% reduction in total communication volume using hypergraph partitioning (PaToH) versus graph partitioning (METIS)
Further Benefits of Hypergraph Model: Nonsymmetric Matrices

- Graph model of matrix has edge \((i,j)\) if either \(A(i,j)\) or \(A(j,i)\) nonzero
- Same graph for \(A\) as \(|A| + |A^T|\)
- Ok for symmetric matrices, what about nonsymmetric?
  - Try \(A\) upper triangular

<table>
<thead>
<tr>
<th>Graph Partitioning (Metis)</th>
<th>Hypergraph Partitioning (PaToH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Communication Volume = 254</td>
<td>Total Communication Volume = 181</td>
</tr>
<tr>
<td>Load imbalance ratio = 6%</td>
<td>Load imbalance ratio = 0.1%</td>
</tr>
</tbody>
</table>

Summary: Graphs versus Hypergraphs

- Pros and cons
  - When matrix is non-symmetric, the graph partitioning model (using \(A+A^T\)) loses information, resulting in suboptimal partitioning in terms of communication and load balance.
  - Even when matrix is symmetric, graph cut size is not an accurate measurement of communication volume
  - Hypergraph partitioning model solves both these problems
  - However, hypergraph partitioning (PaToH) can be much more expensive than graph partitioning (METIS)

- Hypergraph partitioners: PaToH, HMETIS, ZOLTAN
- For more see Bruce Hendrickson’s web page
  - “Load Balancing Fictions, Falsehoods and Fallacies”