Graph partitioning

Prof. Richard Vuduc
Georgia Institute of Technology
CSE/CS 8803 PNA: Parallel Numerical Algorithms
[L.27] Tuesday, April 22, 2008
Today’s sources

- CS 194/267 at UCB (Yelick/Demmel)
- “Intro to parallel computing” by Grama, Gupta, Karypis, & Kumar
Review: Dynamic load balancing
Parallel efficiency: 4 scenarios

Consider load balance, concurrency, and overhead
Summary

- Unpredictable loads $\rightarrow$ online algorithms
- Fixed set of tasks with unknown costs $\rightarrow$ self-scheduling
- Dynamically unfolding set of tasks $\rightarrow$ work stealing
- Theory $\Rightarrow$ randomized should work well

Other scenarios: What if…
- … locality is of paramount importance? $\Rightarrow$ Diffusion-based models?
- … processors are heterogeneous? $\Rightarrow$ Weighted factoring?
- … task graph is known in advance? $\Rightarrow$ Static case; graph partitioning (today)
Graph partitioning
Problem definition

- Weighted graph

\[ G = (V, E, W_V, W_E) \]

- Find partitioning of nodes s.t.:
  - Sum of node-weights \( \sim \) even
  - Sum of inter-partition edge-weights minimized

\[ V = V_1 \cup V_2 \cup \cdots \cup V_p \]
Partitioning a Sparse Symmetric Matrix

\[ A = \begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\
\end{bmatrix} \]
Cost of graph partitioning

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

\[
\left( \frac{n}{\frac{n}{2}} \right) \approx \sqrt{\frac{2}{\pi n}} \cdot 2^n
\]

- Problem is NP-Complete, so need heuristics

Sample Graph Partitionings

# Edge Crossings = 6

# Edge Crossings = 10
First heuristic: Repeated graph bisection

To get $2^k$ partitions, bisect $k$ times
Edge vs. vertex separators

- **Edge separator**: $E_s \subset E$, s.t. removal creates two disconnected components

- **Vertex separator**: $V_s \subset V$, s.t. removing $V_s$ and its incident edges creates two disconnected components

\[ E_s \rightarrow V_s: \quad |V_s| \leq |E_s| \]

\[ V_s \rightarrow E_s: \quad |E_s| \leq d \cdot |V_s|, \quad d = \text{max degree} \]
Overview of bisection heuristics

- **With** nodal coordinates: Spatial partitioning
- **Without** nodal coordinates
- **Multilevel** acceleration: Use coarse graphs
Partitioning with nodal coordinates
Intuition:
Planar graph theory

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

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

1. $V = V_1 \cup V_s \cup V_2$
2. $|V_1|, |V_2| \leq \frac{2}{3}|V|$
3. $|V_s| \leq \sqrt{8}|V|$
Inertial partitioning
Inertial partitioning

Choose line $L$
Inertial partitioning

Choose line $L$
Inertial partitioning

Choose line $L$
Inertial partitioning

Choose line $L$
Inertial partitioning

Choose line $L$

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

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

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

- Project points onto $L$
Inertial partitioning

- Choose line $L$

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

$a^2 + b^2 = 1$

- Project points onto $L$
Inertial partitioning

- Choose line $L$

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

$a^2 + b^2 = 1$

- Project points onto $L$

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

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

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

- Compute median and separate
  $$\bar{s} = \text{median}(s_1, \ldots, s_n)$$
How to choose L?
How to choose L?

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

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

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

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

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

\[ \Rightarrow \bar{x} = \frac{1}{n} \sum_k x_k \]

\[ \bar{y} = \frac{1}{n} \sum_k y_k \]

\[ \begin{pmatrix} a \\ b \end{pmatrix} = \text{Eigenvector of smallest eigenvalue of } A \]
What about 3D (or higher dimensions)?

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

\[
|V| = n^3 \\
|V_s| = n^2 \\
= O\left(|V|^{\frac{2}{3}}\right) = O\left(|E|^{\frac{2}{3}}\right)
\]
Random spheres

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

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

Example: 3-ply system
Random spheres

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

Definition: A **k-ply neighborhood system in d dimensions** = set \{D_1, ..., 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

Definition: An \((\alpha, k)\) overlap graph, for \(\alpha \geq 1\) and a k-ply neighborhood:

- Node = \(D_j\)
- Edge \(j \rightarrow i\) if expanding radius of smaller disk by \(\alpha\) causes two disks to overlap

Example: \((1,1)\) overlap graph for a 2D mesh.
Random spheres (cont’d)

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

$$V = V_1 \cup V_s \cup V_2$$

$$|V_1|, |V_2| < \frac{d + 1}{d + 2} \cdot n$$

$$|V_s| = O\left(\alpha \cdot k^{\frac{1}{d}} \cdot n^{\frac{d-1}{d}}\right)$$

- In 2D, same as Lipton & Tarjan
Random spheres: An algorithm

- Choose a sphere $S$ in $\mathbb{R}^d$
- Edges that $S$ “cuts” form edge separator $E_s$
- Build $V_s$ from $E_s$
- Choose $S$ “randomly,” s.t. satisfies theorem with high probability
Random spheres algorithm

Partition 1: All disks inside S

Partition 2: All disks outside S

Separator
Choosing a random sphere: Stereographic projections

Given \( p \) in plane, project to \( p' \) on sphere.

1. Draw line from \( p \) to north pole.
2. \( p' = \) intersection.

\[
\begin{align*}
    p &= (x, y) \\
    p' &= \frac{2x, 2y, x^2 + y^2 - 1}{x^2 + y^2 + 1}
\end{align*}
\]
Random spheres separator algorithm (Miller, et al.)

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

- Other variations exist
- Algorithms are efficient: $O(\text{points})$
- Implicitly assume nearest neighbor connectivity: Ignores edges!
  - Common for graphs from physical models
  - Good “initial guess” for other algorithms
  - Poor performance on non-spatial graphs
Partitioning without nodal coordinates
A coordinate-free algorithm: Breadth-first search

Choose root $r$ and run BFS, which produces:

- Subgraph $T$ of $G$ (same nodes, subset of edges)
- $T$ rooted at $r$
- Level of each node = distance from $r$
Kernighan/Lin (1970): Iteratively refine

Given edge-weighted graph and partitioning:

\[ G = (V, E, W_E) \]

\[ V = A \cup B, \quad |A| = |B| \]

\[ E_s = \{(u, v) \in E : u \in A, v \in B\} \]

\[ T \equiv \text{cost}(A, B) \equiv \sum_{e \in E_s} w(e) \]

Find equal-sized subsets X, Y of A, B s.t. swapping reduces cost

Need ability to quickly compute cost for many possible X, Y
K-L refinement: Definitions

**Definition:** “External” and “internal” costs of $a \in A$, and their difference; similarly for $B$:

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

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

$$D(a) \equiv E(a) - I(a)$$
Consider swapping two nodes

- Swap $X = \{a\}$ and $Y = \{b\}$:
  
  $A' = (A - a) \cup b$
  
  $B' = (B - b) \cup a$

- Cost changes:
  
  $T' = T - (D(a) + D(b) - 2w(a, b))$
  
  $\equiv T - \text{gain}(a, b)$
KL-refinement-algorithm (A, B):

Compute \( T = \text{cost}(A,B) \) for initial A, B \( \ldots \) cost = \( O(|V|^2) \)

Repeat

\( \ldots \) One pass greedily computes \(|V|/2 \) possible \( X,Y \) to swap, picks best

Compute costs \( D(v) \) for all \( v \) in \( V \) \( \ldots \) cost = \( O(|V|^2) \)

Unmark all nodes in \( V \) \( \ldots \) cost = \( O(|V|) \)

While there are unmarked nodes

Find an unmarked pair \((a,b)\) maximizing \( \text{gain}(a,b) \) \( \ldots \) cost = \( O(|V|^2) \)

Mark \( a \) and \( b \) (but do not swap them) \( \ldots \) cost = \( O(1) \)

Update \( D(v) \) for all unmarked \( v \),

as though \( a \) and \( b \) had been swapped \( \ldots \) cost = \( O(|V|) \)

Endwhile

\( \ldots \) At this point we have computed a sequence of pairs

\( \ldots \) \((a_1,b_1), \ldots , (a_k,b_k)\) and gains \( \text{gain}(1),\ldots , \text{gain}(k) \)

\( \ldots \) where \( k = |V|/2 \), numbered in the order in which we marked them

Pick \( m \) maximizing \( \text{Gain} = \sum_{k=1}^{m} \text{gain}(k) \) \( \ldots \) cost = \( O(|V|) \)

\( \ldots \) Gain is reduction in cost from swapping \((a_1,b_1)\) through \((a_m,b_m)\)

If \( \text{Gain} > 0 \) then \( \ldots \) it is worth swapping

Update \( \text{newA} = A - \{ a_1,\ldots ,a_m \} \cup \{ b_1,\ldots ,b_m \} \) \( \ldots \) cost = \( O(|V|) \)

Update \( \text{newB} = B - \{ b_1,\ldots ,b_m \} \cup \{ a_1,\ldots ,a_m \} \) \( \ldots \) cost = \( O(|V|) \)

Update \( T = T - \text{Gain} \) \( \ldots \) cost = \( O(1) \)

endif

Until \( \text{Gain} \leq 0 \)
Comments

- Expensive: $O(n^3)$

- Some gain(k) may be negative, but can still get positive final gain
  - Escape local minima

- Outer loop iterations?
  - On very small graphs, $|V| \leq 360$, KL show convergence after 2-4 sweeps
  - For random graphs, probability of convergence in 1 sweep goes down like $2^{-|V|/30}$
Spectral bisection

- Motivation: Vibrating string
- Computation: Compute eigenvector
  - To optimize sparse matrix-vector multiply, partition graph
  - To graph partition, find eigenvector of matrix associated with graph
  - To find eigenvector, do sparse matrix-vector multiply
A physical intuition:
Vibrating strings

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

- **Definition:** Incidence matrix \( \text{In}(G) \) = \(|V| \times |E|\) matrix, s.t. edge \( e = (i, j) \) →
  - \( \text{In}(G)[i, e] = +1 \)
  - \( \text{In}(G)[j, e] = -1 \)
  - Ambiguous (multiply by -1), but doesn’t matter

- **Definition:** Laplacian matrix \( \text{L}(G) \) = \(|V| \times |V|\) matrix, s.t. \( \text{L}(G)[i, j] = \ldots \)
  - degree of node \( i \), if \( i == j \);
  - -1, if there is an edge \( (i, j) \) and \( i \neq j \)
  - 0, otherwise
Examples of incidence and Laplacian matrices

Graph G

Incidence Matrix In(G)

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

Laplacian Matrix L(G)

\[
\begin{bmatrix}
1 & 1 & -1 \\
1 & 2 & -1 \\
1 & 2 & 2 & -1 \\
1 & 2 & -1 & 2 \\
1 & 1 & & & 2
\end{bmatrix}
\]

Nodes numbered in black
Edges numbered in blue
Theorem: Properties of $L(G)$

- $L(G)$ is symmetric $\Rightarrow$ eigenvalues are real, eigenvectors real & orthogonal
- $L(G) = In(G) * In(G)^T$
- Eigenvalues are non-negative, i.e., $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$
- Number of connected components of $G = $ number of 0 eigenvalues

- Definition: $\lambda_2(G) =$ algebraic connectivity of $G$
  - Magnitude measures connectivity
  - Non-zero if and only if $G$ is connected
Spectral bisection algorithm

- Algorithm:
  - Compute eigenpair \((\lambda_2, q_2)\)
  - For each node \(v\) in \(G\):
    - if \(q_2(v) < 0\), place in partition \(N_-\)
    - else, place in partition \(N_+\)

- Why?
Why the spectral bisection is “reasonable”: Fiedler’s theorems

- **Theorem 1:**
  - $G$ connected $\implies N_-$ connected
  - All $q_2(v) \neq 0$ $\implies N_+$ connected

- **Theorem 2:** Let $G_1$ be “less-connected” than $G$, i.e., has same nodes & subset of edges. Then $\lambda_2(G_1) \leq \lambda_2(G)$

- **Theorem 3:** $G$ is connected and $V = V_1 \cup V_2$, with $|V_1| \sim |V_2| \sim |V|/2$
  - $\implies |E_s| \geq 1/4 \cdot |V| \cdot \lambda_2(G)$
Spectral bisection: Key ideas

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

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

\[ (V^+, V^-) \leftarrow \text{Multilevel\_Partition} (G = (V, E)) \]

- If \(|V|\) is “small”, partition directly
- Else:
  - **Coarsen** \( G \to G_c = (V_c, E_c) \)
  - \((V_c^+, V_c^-) \leftarrow \text{Multilevel\_Partition} (V_c, E_c)\)
  - **Expand** \((V_c^+, V_c^-) \to (V^+, V^-)\)
  - **Improve** \((V^+, V^-)\)
  - Return \((V^+, V^-)\)
Algorithm 1: Multilevel Kernighan-Lin

- Coarsen and expand using maximal matchings
  - Definition: Matching = subset of edges s.t. no two edges share an endpoint
  - Use greedy algorithm
- Improve partitions using KL-refinement
Expanding a partition from coarse-to-fine graph

Converting a coarse partition to a fine partition

Partition shown in green
Multilevel spectral bisection

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

Maximal Independent Subset $N_i$ of $N$

- nodes of $N$
- nodes of $N_i$
Multilevel software

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

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