

# High Performance Computing: Tools and Applications

Edmond Chow  
School of Computational Science and Engineering  
Georgia Institute of Technology

Lecture 17

# Graph and hypergraph partitioning

## References:

- ▶ Graph Partitioning for High Performance Scientific Simulations, Kirk Schloegel, George Karypis, and Vipin Kumar, 2000
- ▶ Hypergraph-Partitioning-Based Decomposition for Parallel Sparse-Matrix Vector Multiplication, Umit V. Catalyurek and Cevdet Aykanat, 1999

Some figures below are from the above references.

# Distributed sparse matrix-vector multiplication

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ A_{21} & A_{22} & \cdots & A_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & \cdots & A_{pp} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}$$

Processor  $i$  stores  $A_{i*}$ ,  $x_i$  and computes  $y_i$   
(block row-wise partitioning).

# Distributed sparse matrix-vector multiplication

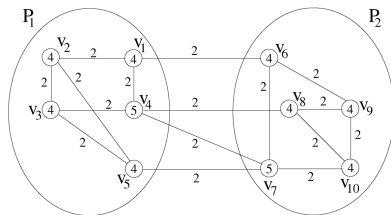
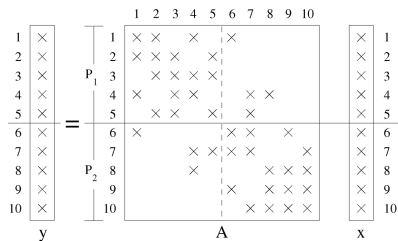
$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ A_{21} & A_{22} & \cdots & A_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & \cdots & A_{pp} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}$$

On processor  $i$ :

1. Send components of  $x_j$  needed by other processors
2. Compute  $y_{local} = A_{ij}x_j$
3. Receive components of  $x_j$  from other processors  $j$  needed by processor  $i$
4. Compute  $y_{ext} = A_{i,ext}x_{ext}$
5. Form  $y_i = y_{local} + y_{ext}$

If processor  $i$  contains nonzeros in column  $k$ , then processor  $i$  needs the  $k$ th scalar component of  $x$ .

# Graph model (for $p = 2$ )



The graph model shows which nonzeros (edges) induce communication in SpMV.

# Graph partitioning problem

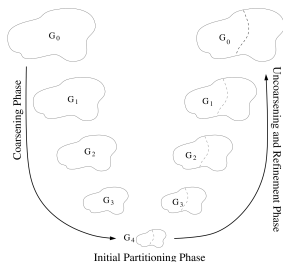
Given  $G = (V, E)$ , and number of parts  $p$ , find the partitioning of the vertices  $P_1, P_2, \dots, P_p$ , where  $P_i$  contains a subset of the vertices in  $V$ , such that the  $P_i$  are disjoint and complete and

$$|P_i| \leq \left( \frac{1}{k} \sum |P_j| \right) (1 + \epsilon)$$

for all  $i$  (number of vertices in each partition is approximately equal) and the number of **cut edges** is minimized. A cut edge straddles two partitions.

This problem is NP-hard.

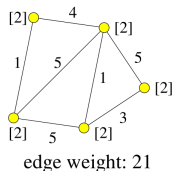
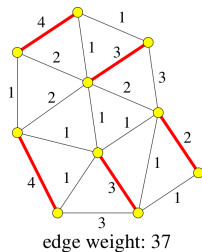
# Multilevel graph partitioning



1. Coarsen the graph by collapsing heavy edges. Vertex weights and edge weights are updated. Vertex and edge weights are initially 1.
2. Repeat until the graph is small enough, then partition the coarsest level graph by some method.
3. Uncoarsen. After each uncoarsening step, refine the partition (using, e.g., Kernighan-Lin).

Software: METIS, Chaco, SCOTCH.

# Graph coarsening

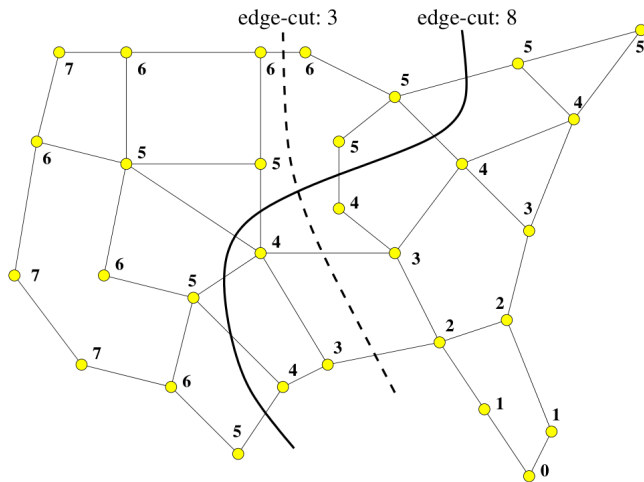


- ▶ Edges are collapsed by using a heavy-edge matching.
- ▶ A matching is a set of edges such that no two edges share a vertex.



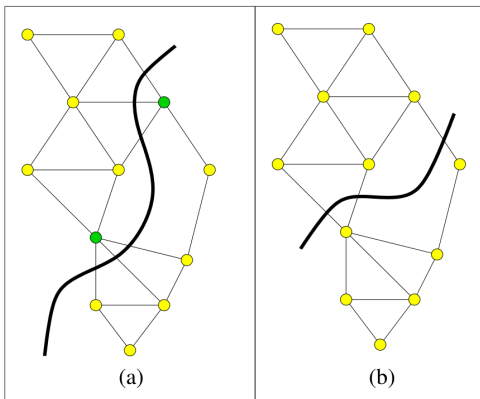
# Levelized nested dissection

Possibility for partitioning the coarsest level graph.



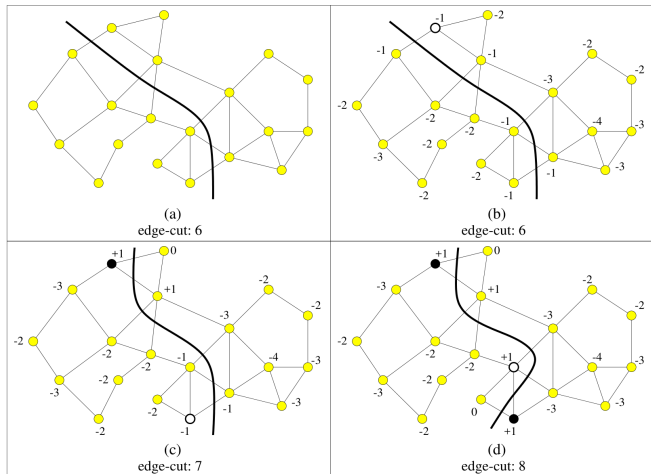
# Kernighan-Lin

Find pairs of nodes on opposite sides of the partition that can change partitions so that the edge-cut is improved. Repeat until all nodes have been moved and then restore the best configuration. Repeat above procedure until the best configuration found is unchanged.

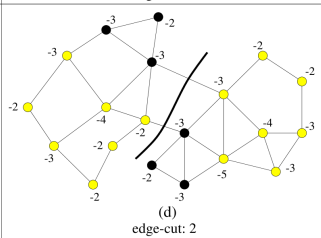
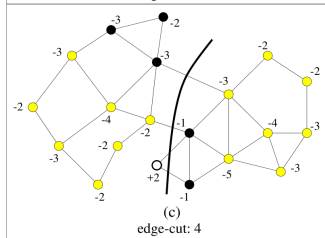
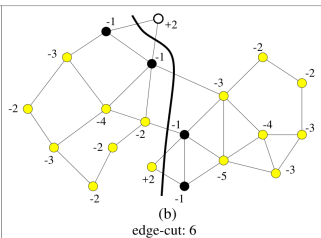
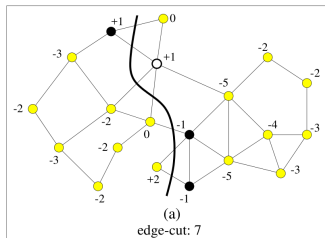


# Fiduccia-Mattheyses

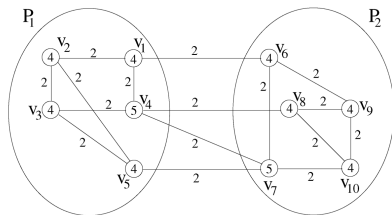
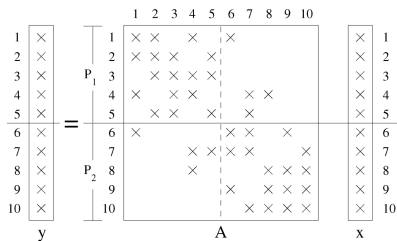
Compute gain for each node. Change partition of node with largest gain. Update gains. Gains are stored in priority queue and a node that has moved will not move again until all others have moved.



# Fiduccia-Mattheyses



# Deficiency with graph partitioning



The cut size does not measure the communication volume.

This deficiency can be addressed with hypergraph partitioning.

# Hypergraph

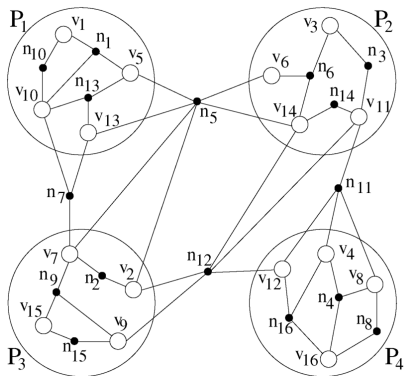
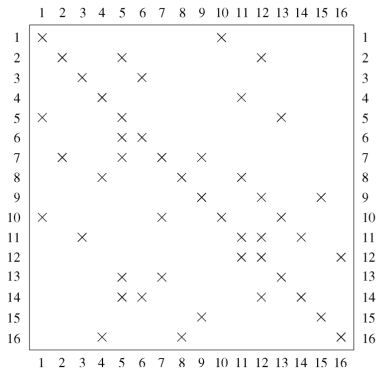
Hypergraph  $H = (V, N)$ , where  $V$  is the set of vertices and  $N$  is the set of hyperedges (or nets).

A hyperedge can connect more than 2 vertices.

## Hypergraph of a sparse matrix

- ▶ Rows are vertices.
- ▶ Columns are hyperedges. Nonzeros in column  $j$  correspond to the vertices connected by the hyperedge.

# Hypergraph



# Hypergraph partitioning problem

Group the vertices into approximately equal partitions such that

$$\min \sum_{n_i \text{ is cut}} (\lambda_i - 1)$$

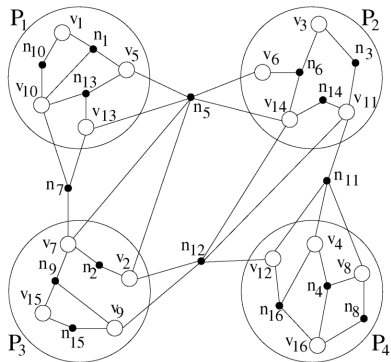
where  $\lambda_i$  is the number of parts connected by hyperedge  $i$ .

Each hyperedge belongs in a partition, so the number of neighbors is  $\lambda_i - 1$ .

NP-hard problem.



# Hypergraph partitioning



	10	13	5	1	6	14	11	3	2	15	7	9	8	16	12	4	
P <sub>1</sub>	10	×	×	×						×							10
	13		×	×							×						13
	5		×	×	×												5
	1	×			×												1
P <sub>2</sub>	6			×	×												6
	14			×		×	×								×		14
	11					×	×	×								×	11
	3					×		×									3
P <sub>3</sub>	2			×					×							×	2
	15									×		×					15
	7			×					×		×	×					7
	9									×		×					9
P <sub>4</sub>	8					×							×			×	8
	16												×	×	×	×	16
	12						×							×	×		12
	4						×							×		×	4

# Hypergraph partitioning

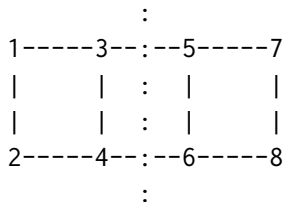
Multilevel algorithm: coarsen hyperedges, refinement with hyperedges.

Software: PaToH, hMETIS.

For FEM meshes, regular graph partitioning is usually good enough. For very unstructured meshes, e.g., from circuit simulation, hypergraph partitioning is needed.

## Exercise 8 - due Wed. Nov. 2, 10 pm

Consider this graph consisting of 8 vertices and a partition into two parts (separated by the dotted line):



What is the hypergraph corresponding to this graph? How many hyperedges are cut by the given partitioning?

# Final Project Options

## Jacobi-Schwarz

1. Implement Jacobi-Schwarz in 3D using Global Arrays. Global Arrays provides a shared memory programming style for distributed memory machines. You will have to be able to download and install Global Arrays.
2. Implement Jacobi-Schwarz in 2D using MPI remote memory access (a.k.a. one-sided) functions. Also test the use of multiple threads per MPI process (e.g., by using controlling the number of threads used by MKL).

# Final Project Options

## **Brownian Dynamics**

3. Implement a BD code with steric and hydrodynamic interactions in shared memory using Cilk Plus and OpenMP. Compare their performance and ease of programming. Be sure to use vectorization in your implementations.
4. Implement a BD code with steric interactions (no HI) using MPI. Particles are partitioned among the MPI processes. Partition the particles spatially to reduce communication, and particles can change partitions after a time step.

# Final Project Report and Guidelines

- ▶ Options represent a baseline project. Make your project your own.
- ▶ Unlike the mini-projects, the main product is your report.
  - ▶ How did you design your code?
  - ▶ How did you optimize your code?
  - ▶ How did you measure performance?
  - ▶ How did your performance measurements suggest further optimizations?
- ▶ We will likely only quickly read and not compile your code.
- ▶ Think of yourselves like graduate researchers.

# Important Dates

- ▶ Class cancelled: Nov 15 and Nov 17
- ▶ Presentations: Tue Nov 26 and Thurs Dec 1
- ▶ Last class: Tue Dec 6
- ▶ Report due Friday Dec 16, 10 pm
- ▶ Grades due: Dec 19