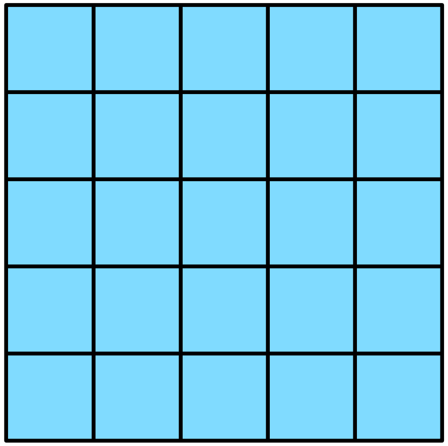
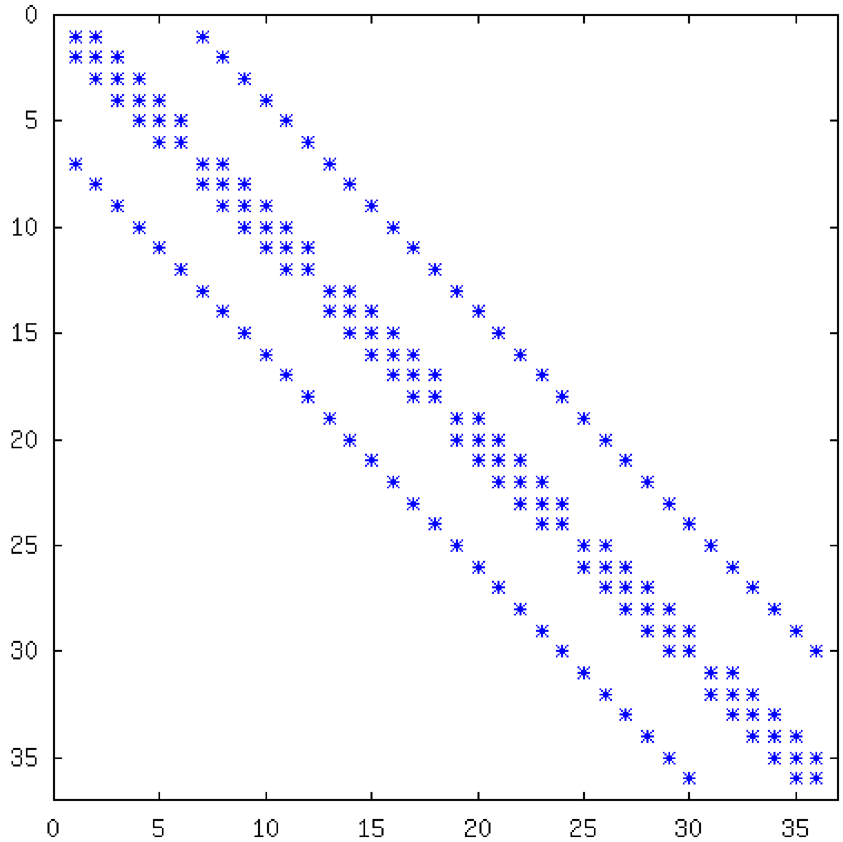
Partitioning: Motivational Problem: Computing the temperature distribution

- Use the “diffusion rule” that the average temperature at a grid point is the average temperature of its neighboring grid points
- Solve the equations by iteratively updating the temperatures at each grid point
- The equations can be expressed as $Ax = b$ where A is a **sparse matrix**.

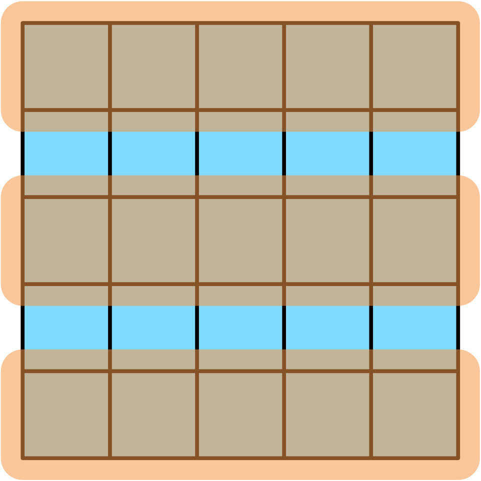
Pattern of the sparse matrix

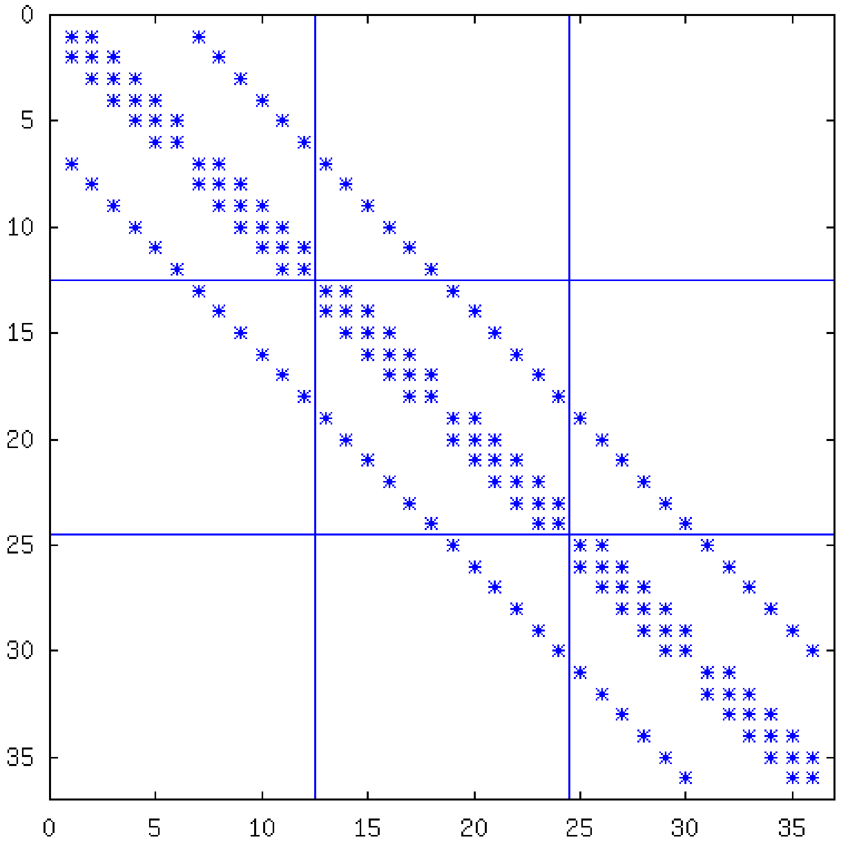




How would you partition this matrix for performing a sparse matrix-vector multiplication?

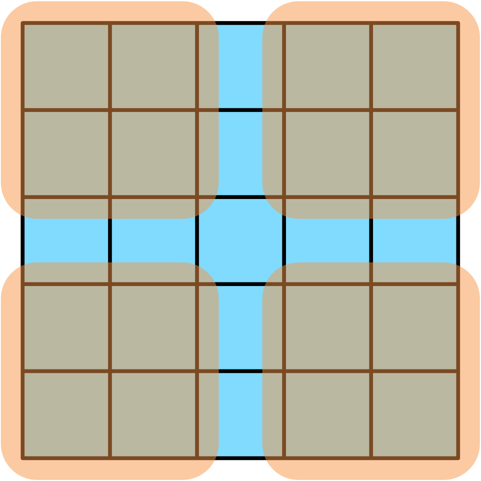
Partitioning for 3 processes

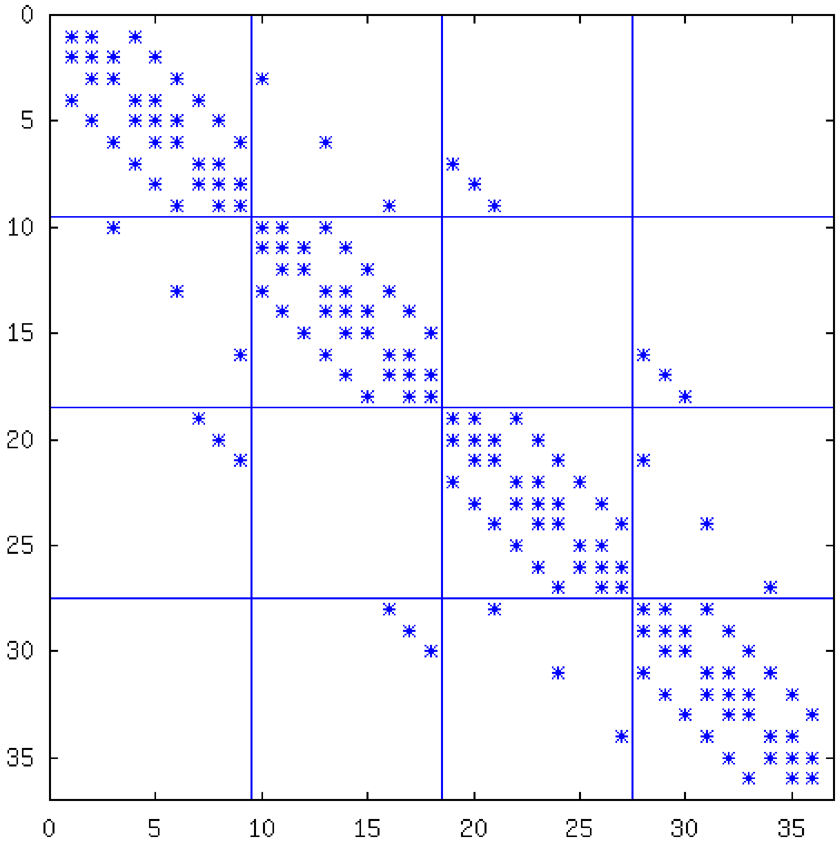




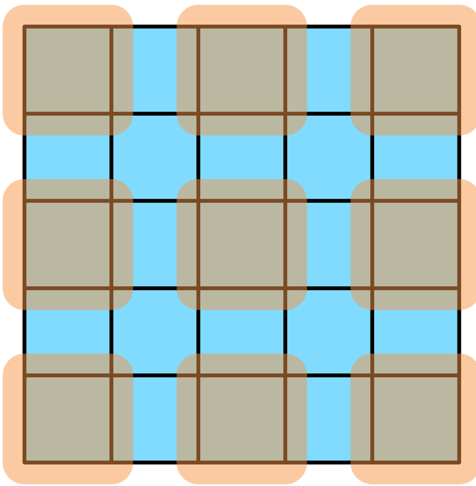
The vector is partitioned into 3 parts the same way.

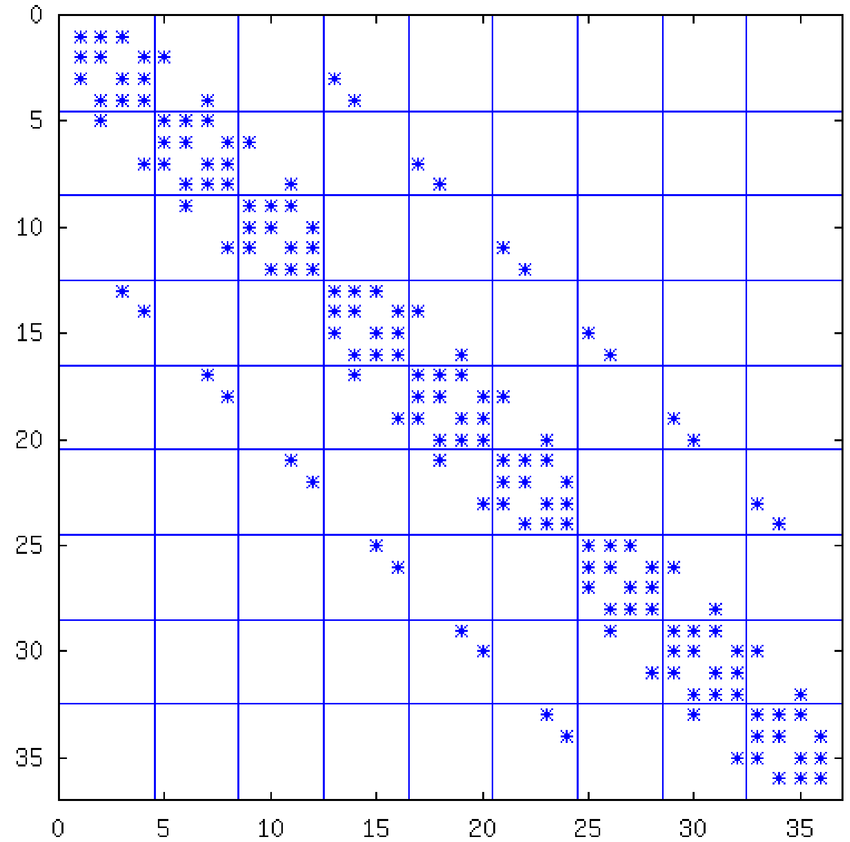
Partitioning for 4 processes



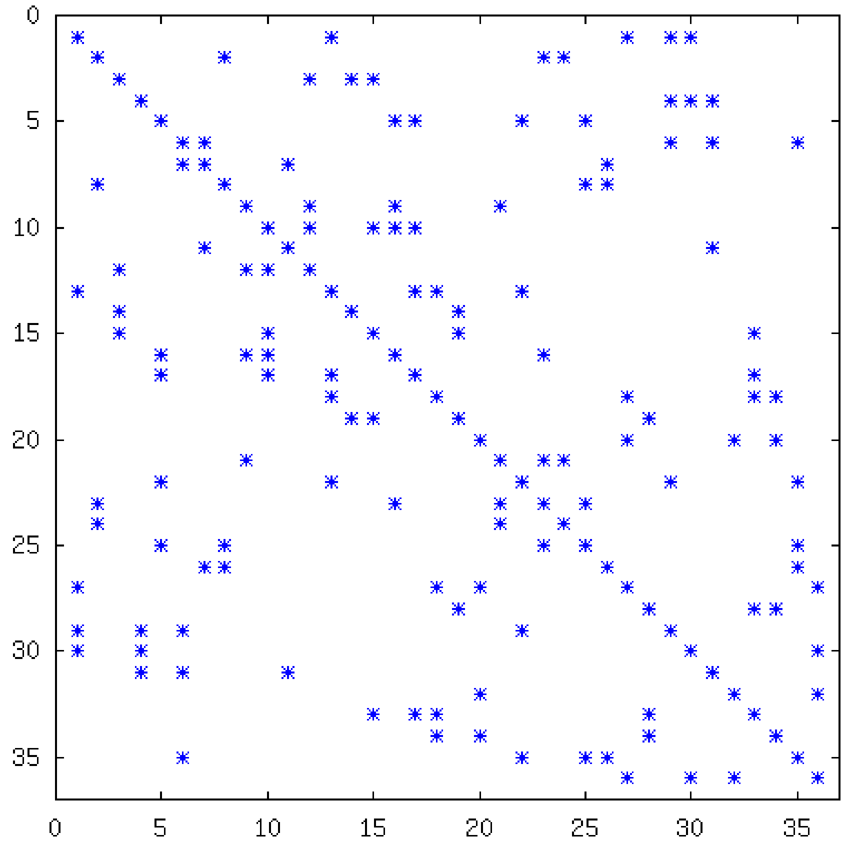


Partitioning for 9 processes





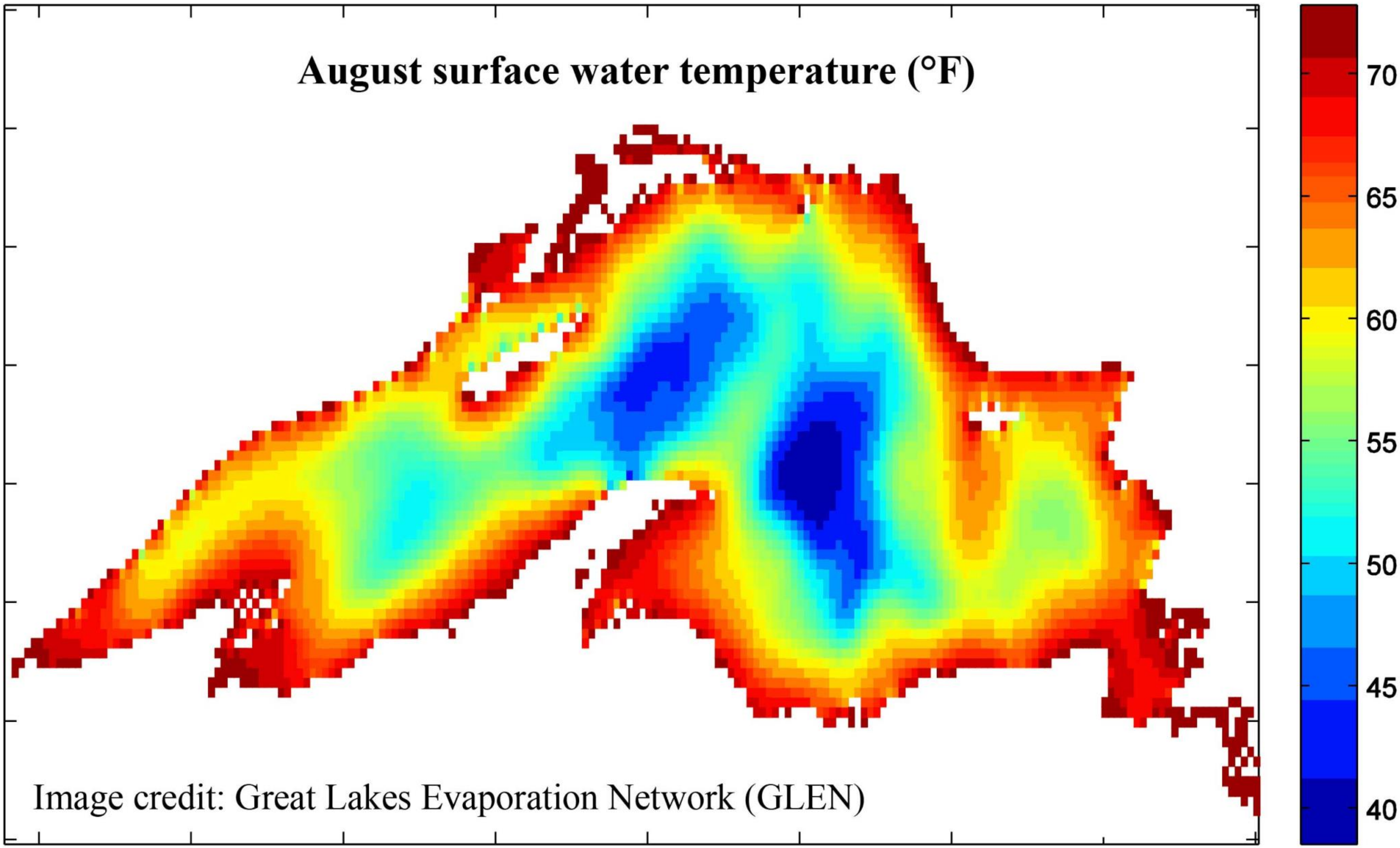
How to partition this matrix to reduce communication?

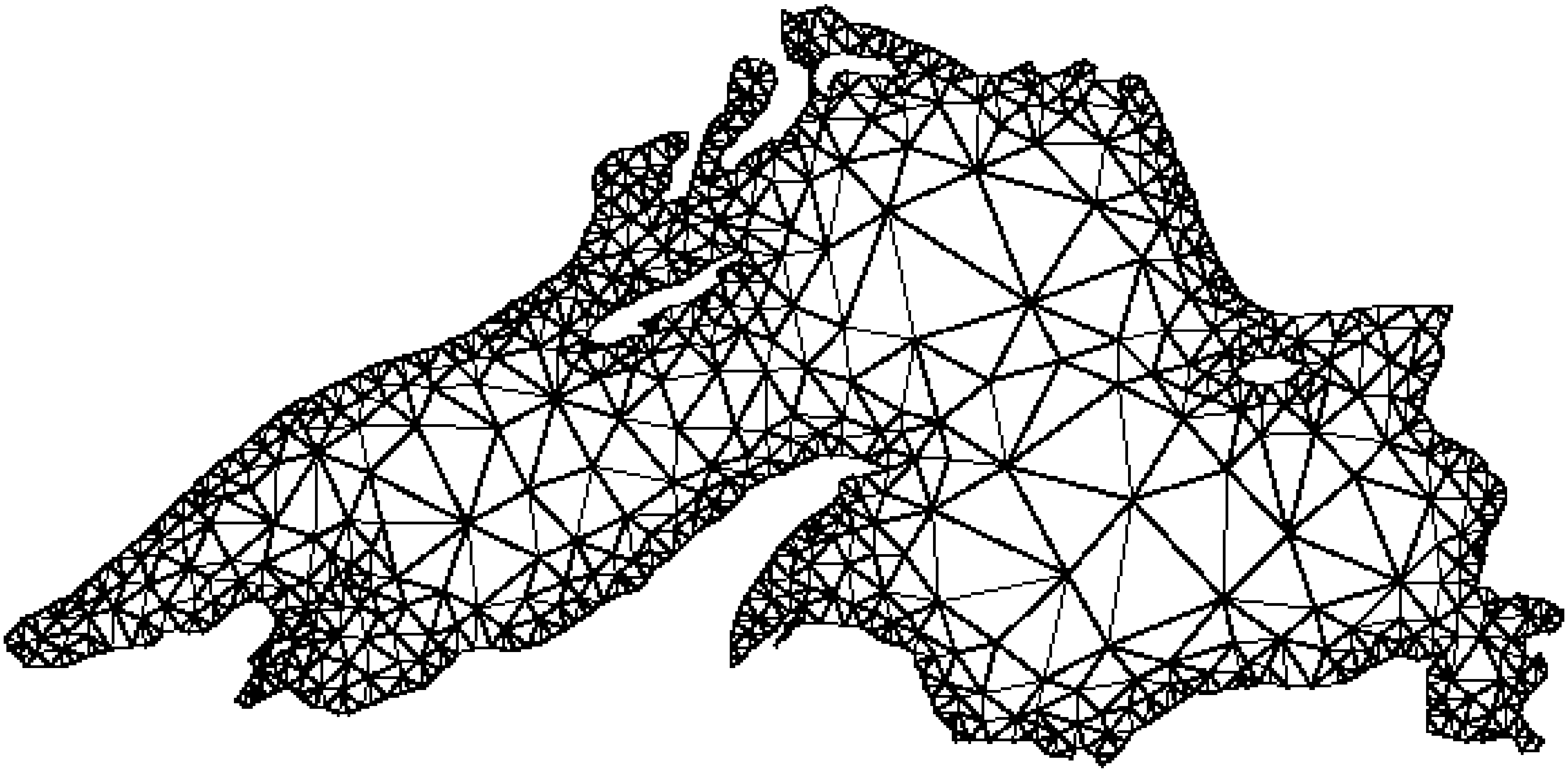


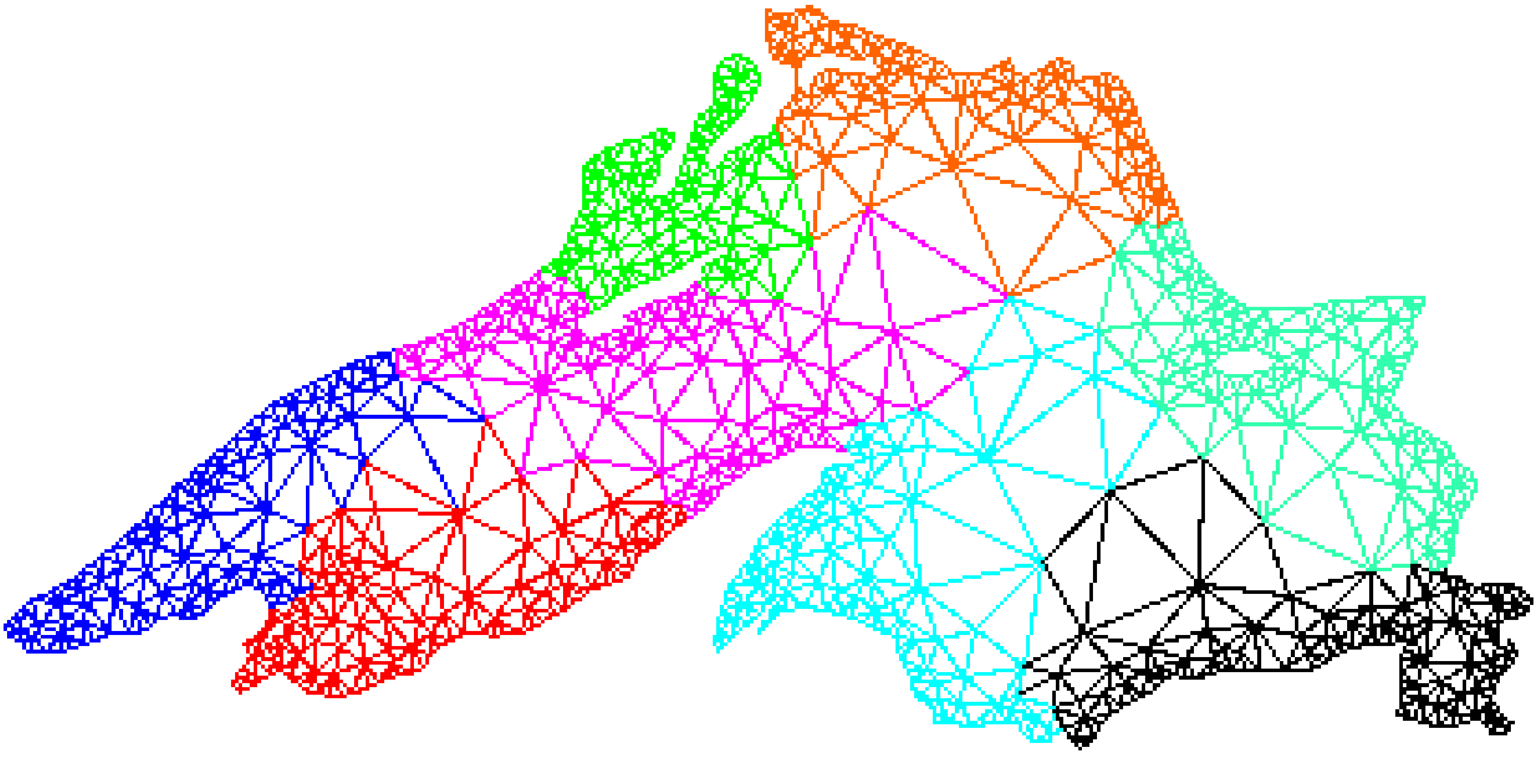
Partitioning sparse matrices

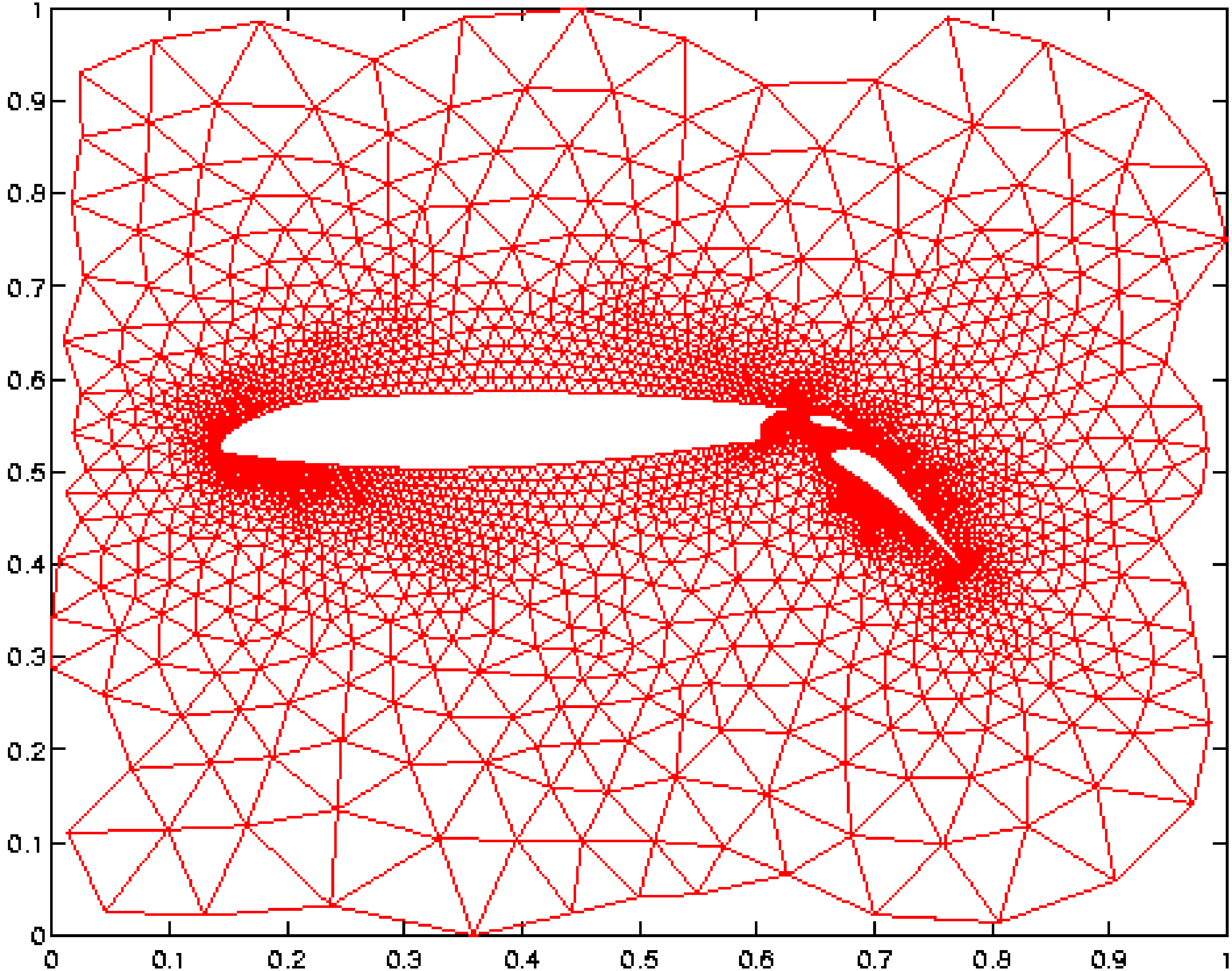
- The sparse matrix corresponds to a graph. Partition the graph in a way such that communication is reduced.
- In many scientific problems, the sparse matrix is not arbitrary: it corresponds to the discretization mesh.
- For a mesh partitioning, number of neighbors is bounded independent of the mesh size.

Lake Superior water temperature forecast (2014)



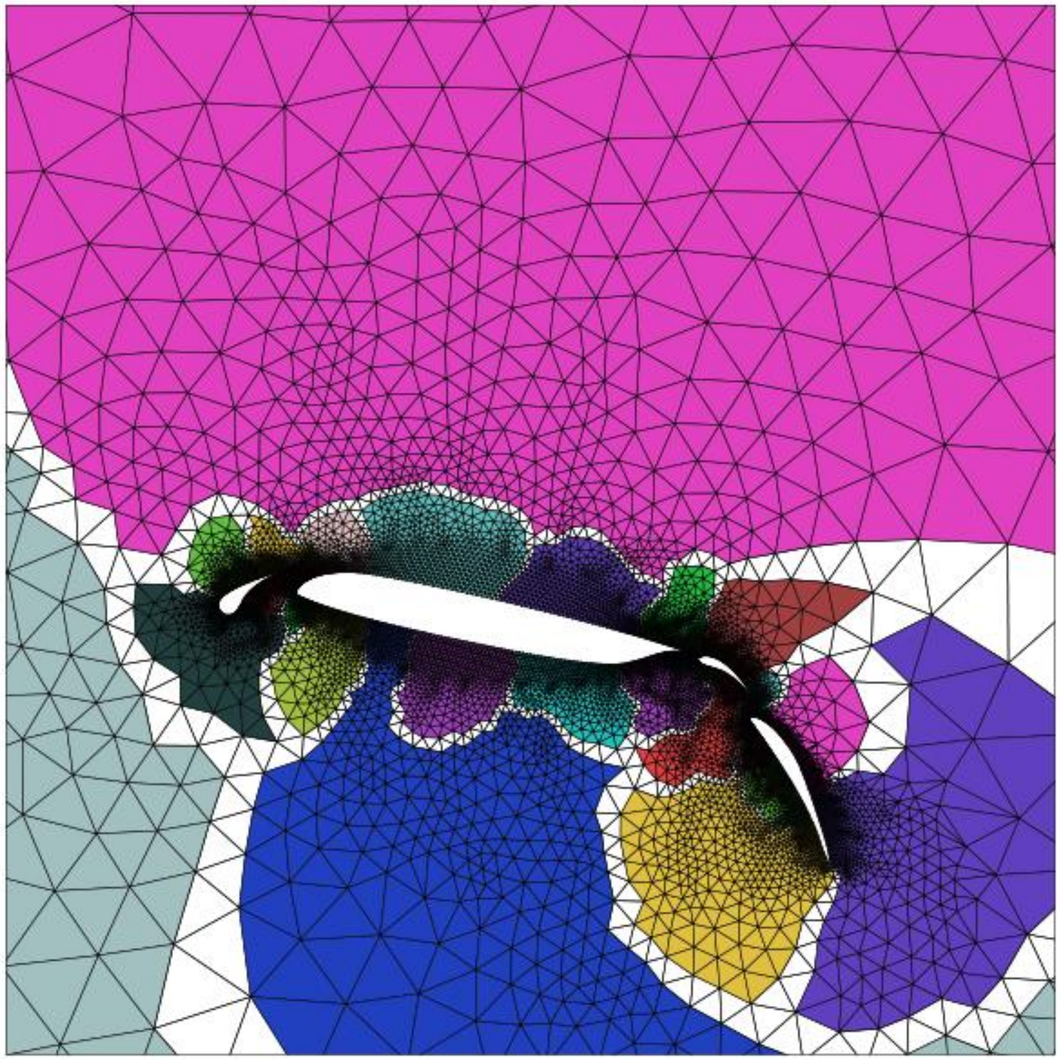






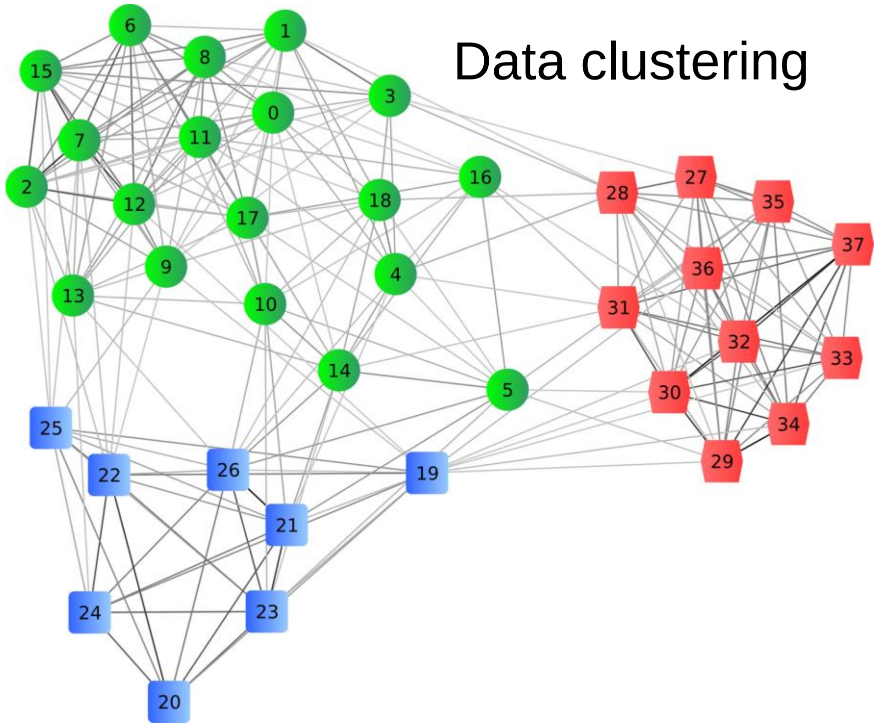
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Partitioning Unstructured Problems



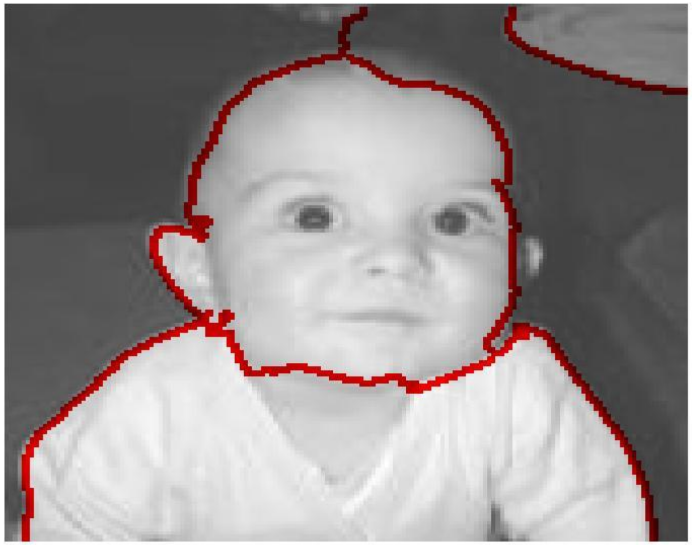
Many applications of graph partitioning

Data clustering



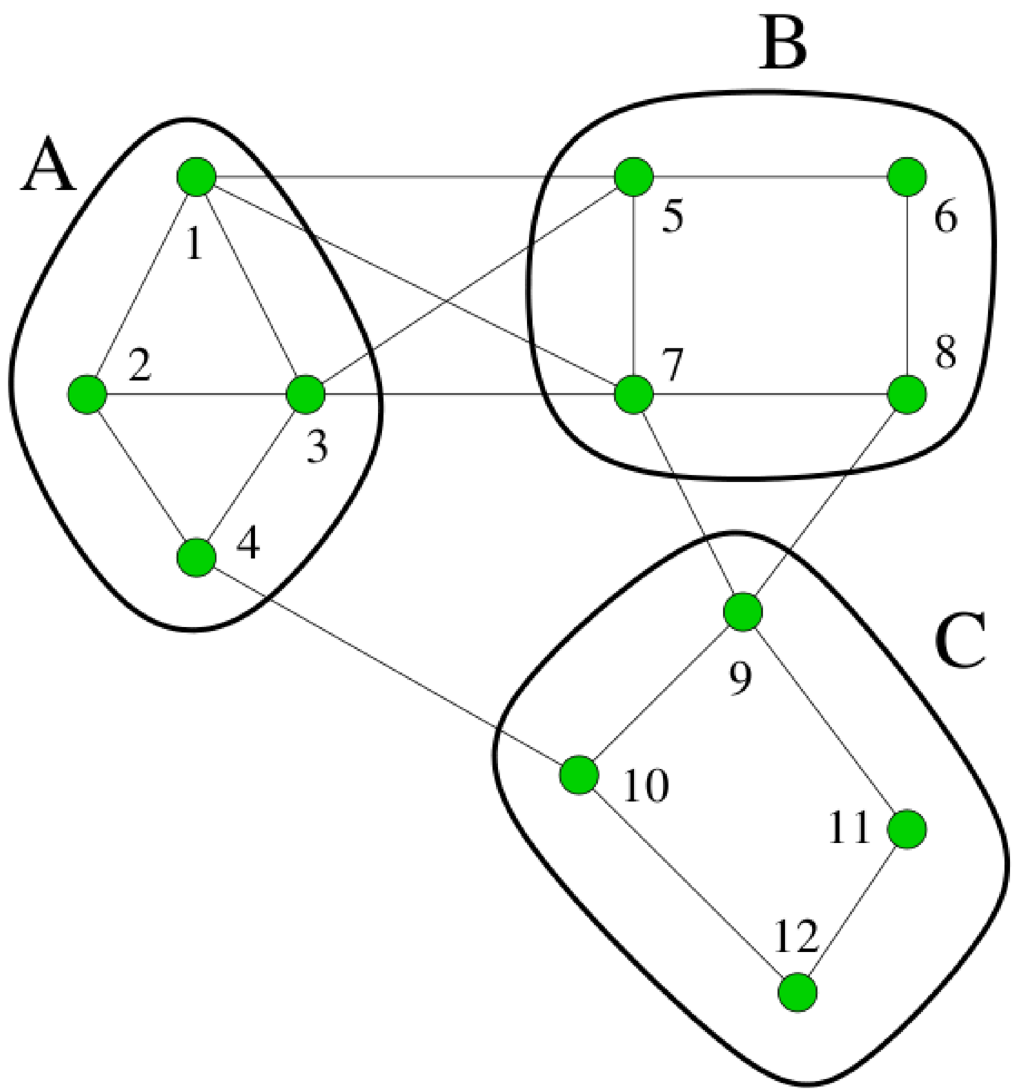
http://bio.informatik.uni-jena.de/peace/

"Normalized Cut" for image segmentation



J. Shi

Graph Formulation

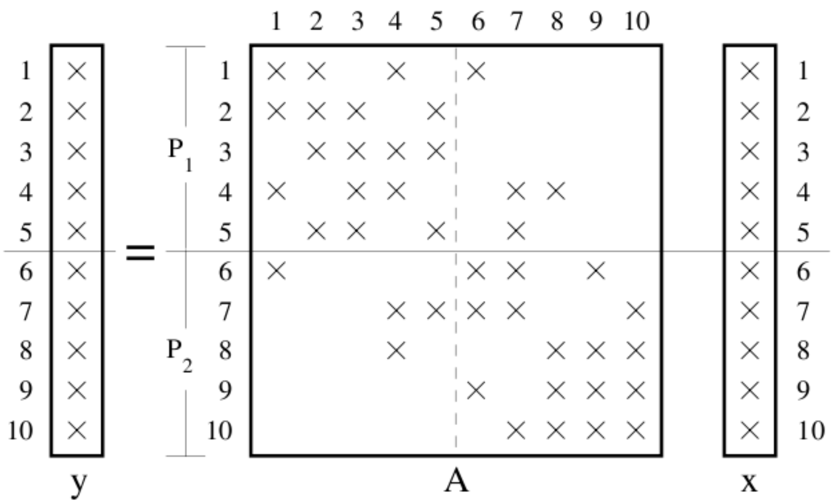


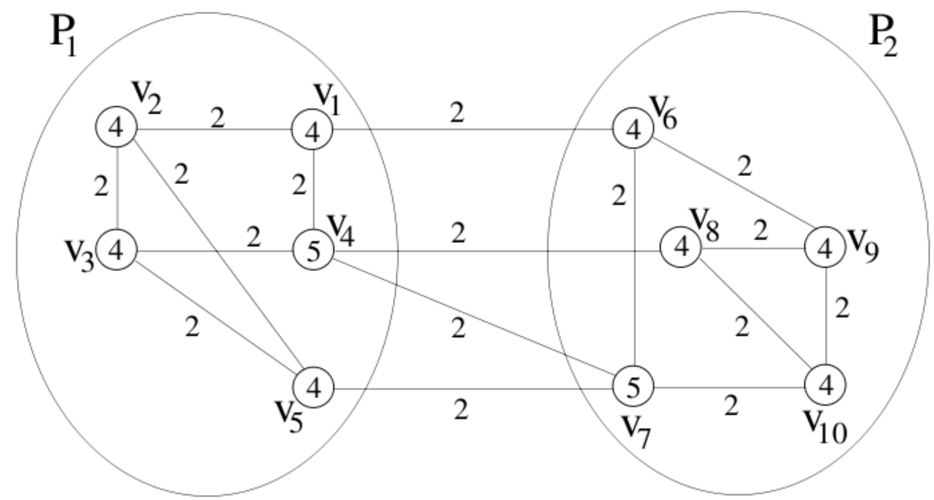
Graph Formulation (Symmetric)

Partitioning is a symmetric “reordering” of the rows and columns of the matrix

Sum of weights of cut edges = number of off-block-diagonal nonzeros

Traditional Graph Partitioning: Minimize “edge-cut” while keeping partitions balanced



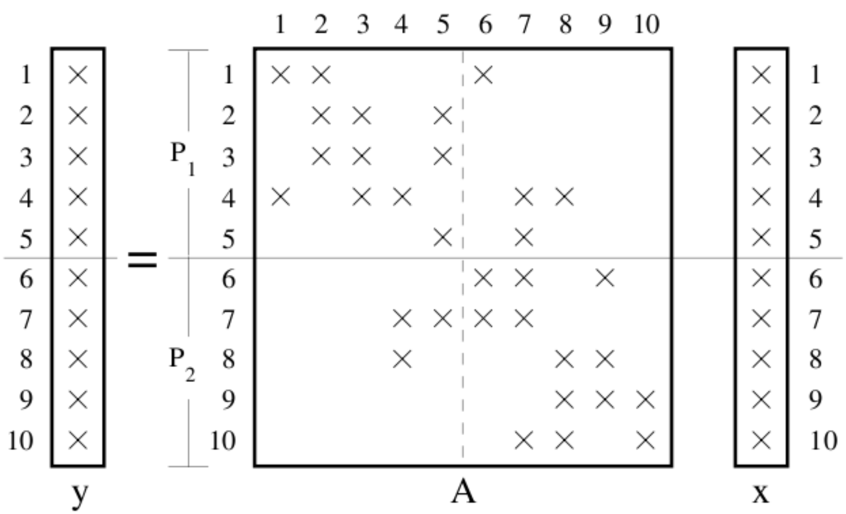


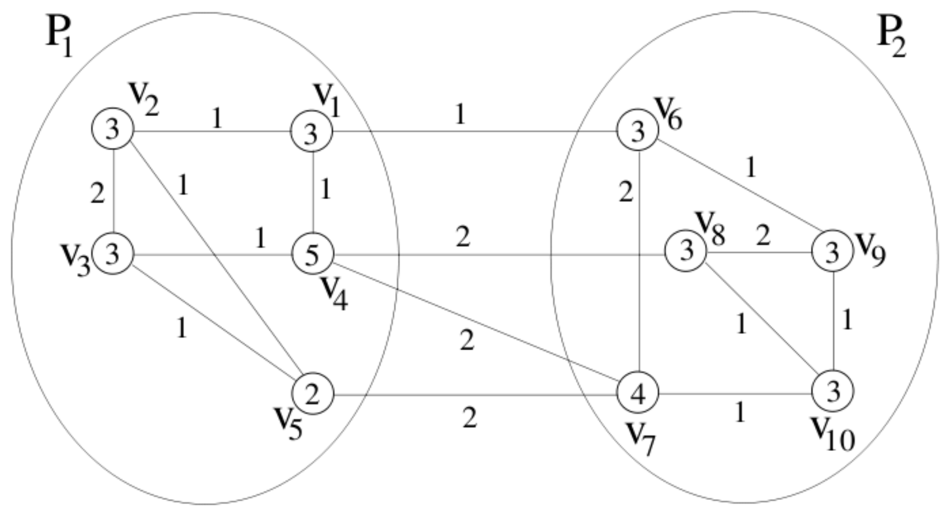
However, communication volume is proportional to the number of boundary vertices

Also should try to minimize the total number of messages (minimize num neighbors)

Minimize the maximum communication cost for among all processors

Graph Formulation (Nonsymmetric)





Graphs and Hypergraphs

- Graph = vertices and edges
  (edges join two vertices)
- Hypergraph = vertices and hyperedges
  (hyperedges join two or more vertices)

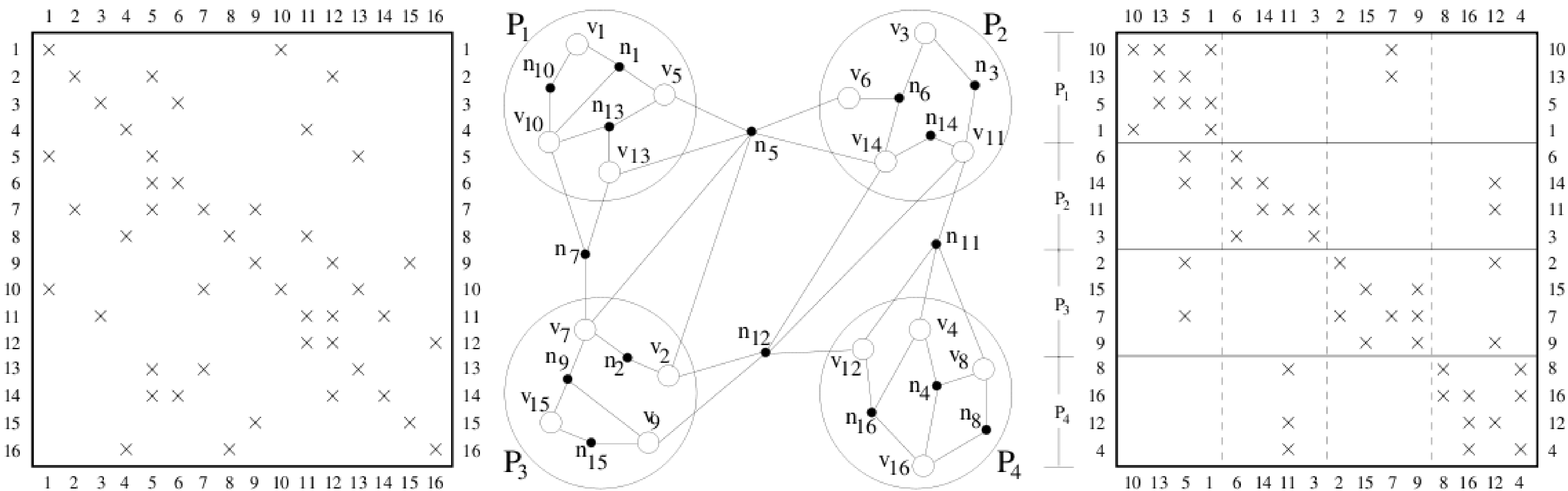
Graph vs. Hypergraph Partitioning

- For finite element meshes, graph partitioning is “good enough”
- For very unstructured meshes, hypergraph partitioning may be useful (e.g., PaToH library)

Hypergraph Formulation

Attempt to directly minimize the actual communication volume.
K-way hypergraph partitioning is common in circuit partitioning.

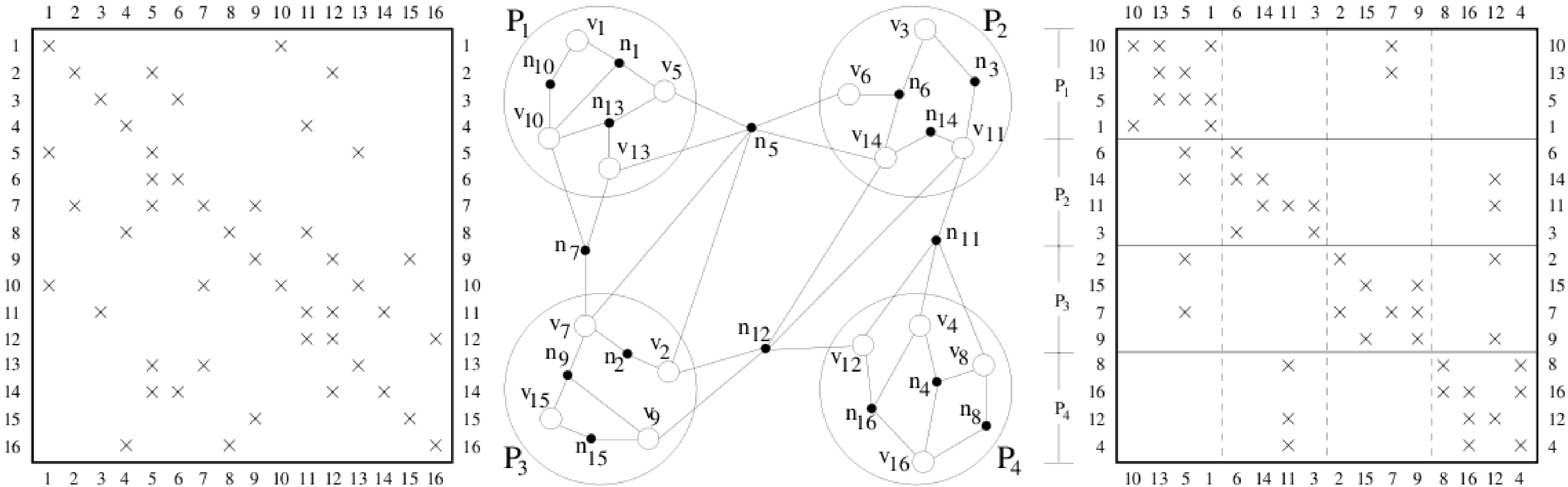
vertex=row
net=column



Hypergraph Formulation

Attempt to directly minimize the actual communication volume.
K-way hypergraph partitioning is common in circuit partitioning.

vertex=row
net=column



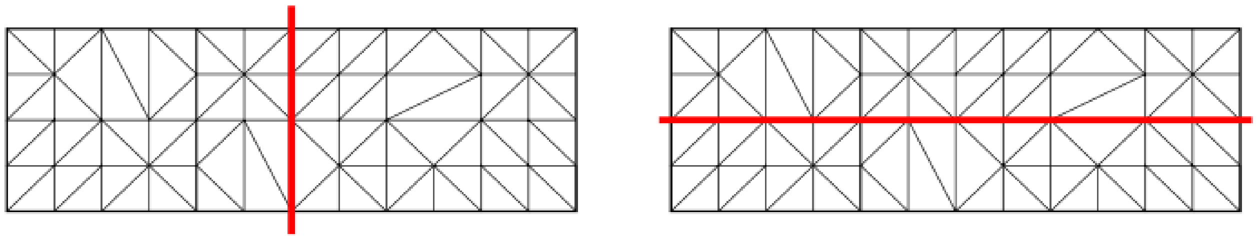
$$\text{cut } size = \sum_{n_j \in Ext} (\lambda_j - 1)$$

Partitioning

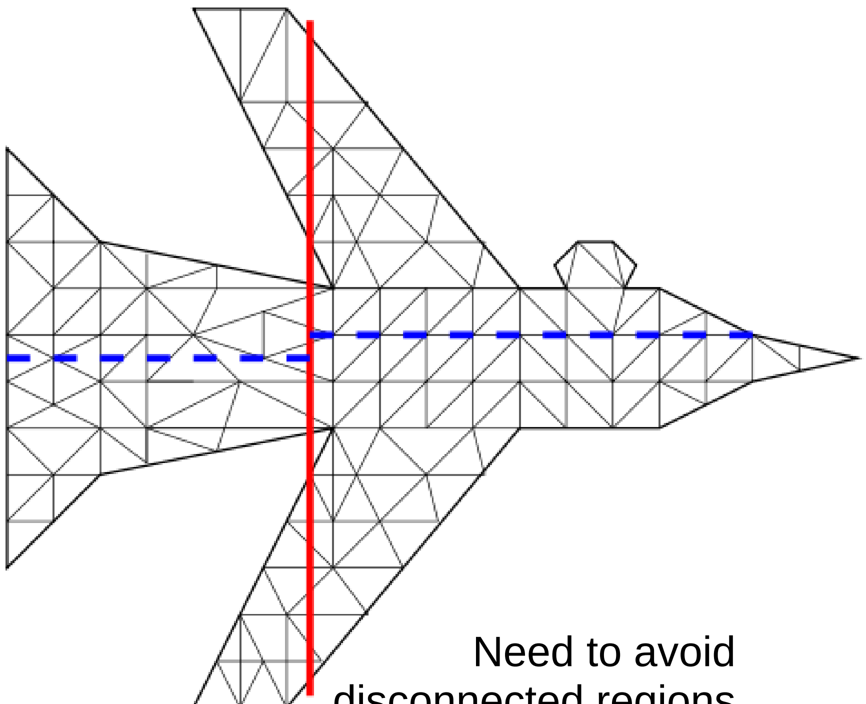
- Geometric techniques
- Combinatorial techniques
- Spectral graph partitioning
- Multilevel techniques

Geometric Techniques

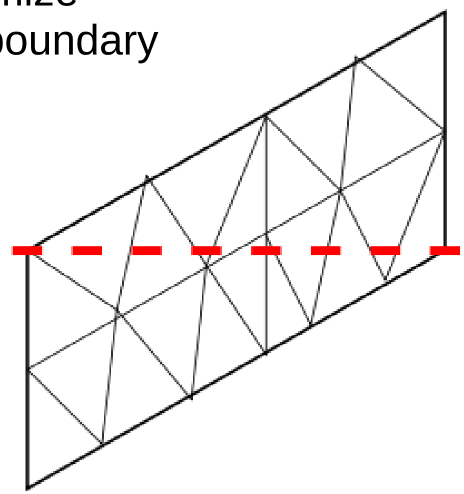
Use only coordinate information

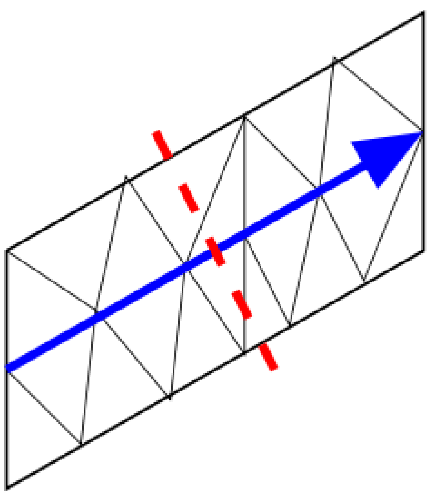


Attempt to minimize
the size of the boundary



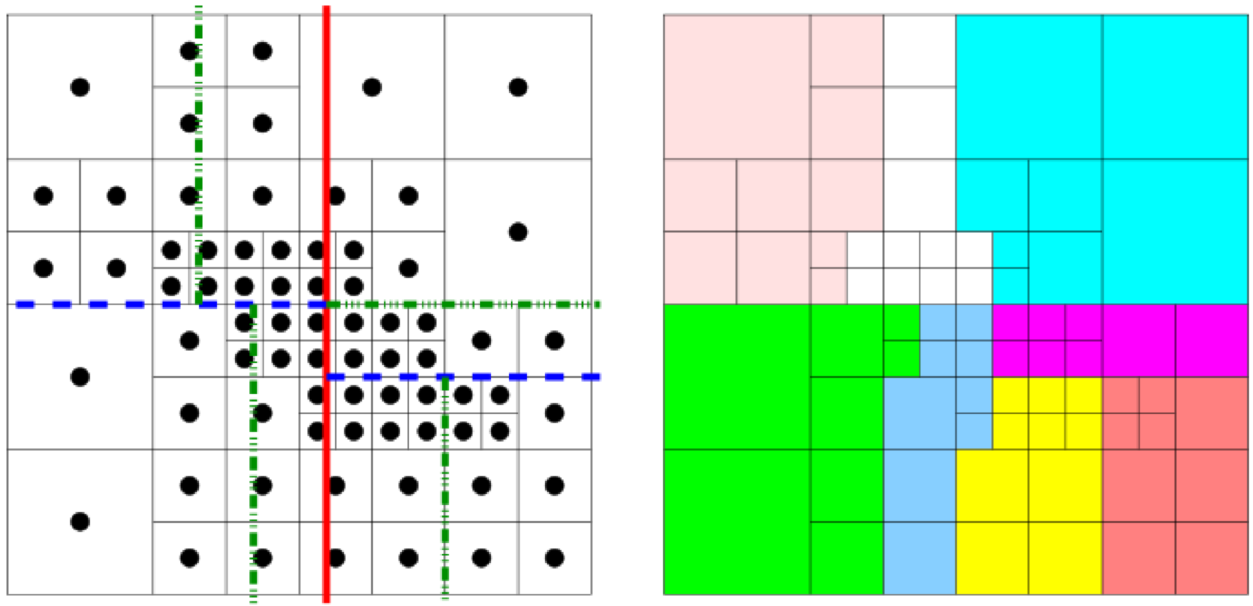
Need to avoid
disconnected regions





Geometric techniques
are generally fast
but low quality

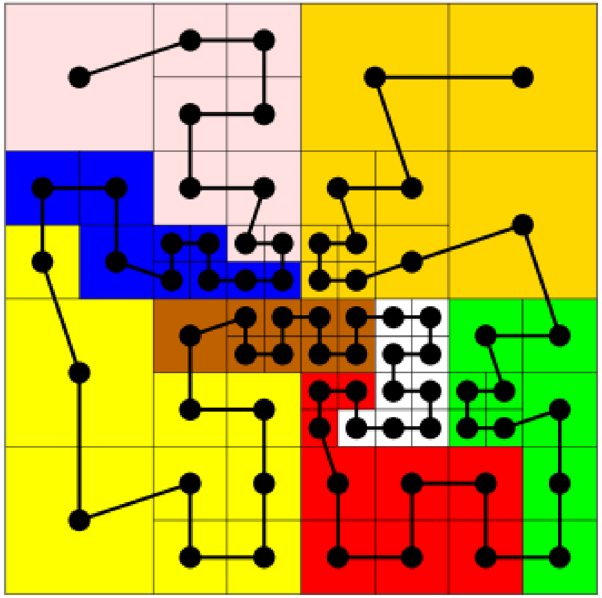
Geometric Techniques



Coordinate nested dissection:

Project centers of mass onto
the longest coordinate axis;
then bisect these centers

Geometric/Combinatorial

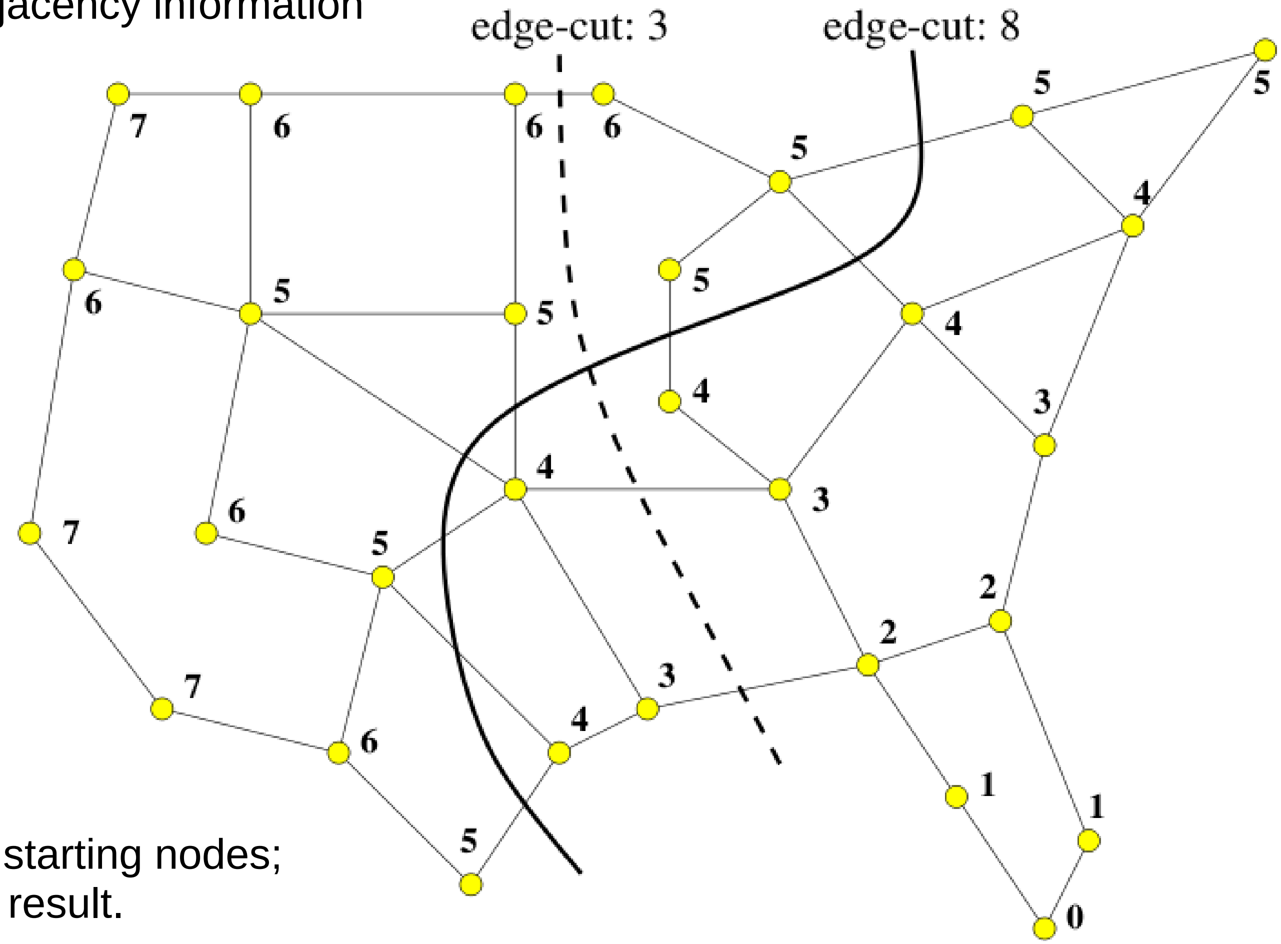


Space-filling curves:

Curve fills space in a locality-preserving fashion;
The resulting ordering is partitioned into k parts.

Combinatorial Techniques

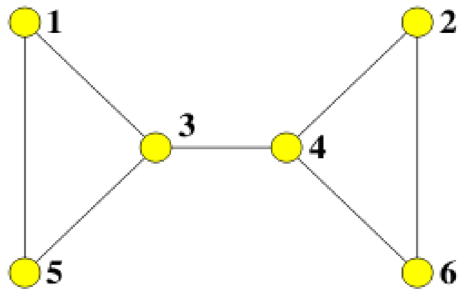
Use only adjacency information



Use multiple starting nodes;
Choose best result.

Levelized Nested Dissection

Spectral Partitioning



+.46
-.46
+.26
-.26
+.46
-.46

Use second largest eigenvalue eigenvector of the graph Laplacian (Fieldler vector) for partitioning

	1	2	3	4	5	6
1			1		1	
2				1		1
3	1			1	1	
4		1	1			1
5	1		1			
6		1		1		

A

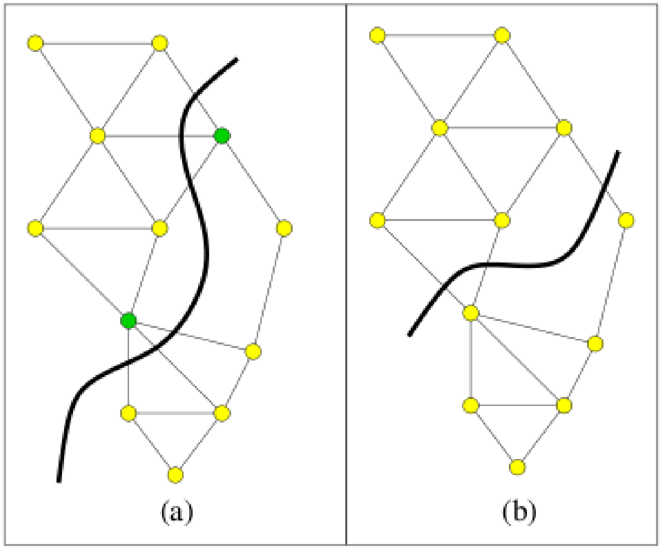
	1	2	3	4	5	6
1	2					
2		2				
3			3			
4				3		
5					2	
6						2

D

	1	2	3	4	5	6
1	-2		1		1	
2		-2		1		1
3	1		-3	1	1	
4		1	1	-3		1
5	1		1		-2	
6		1		1		-2

$L_G = A - D$

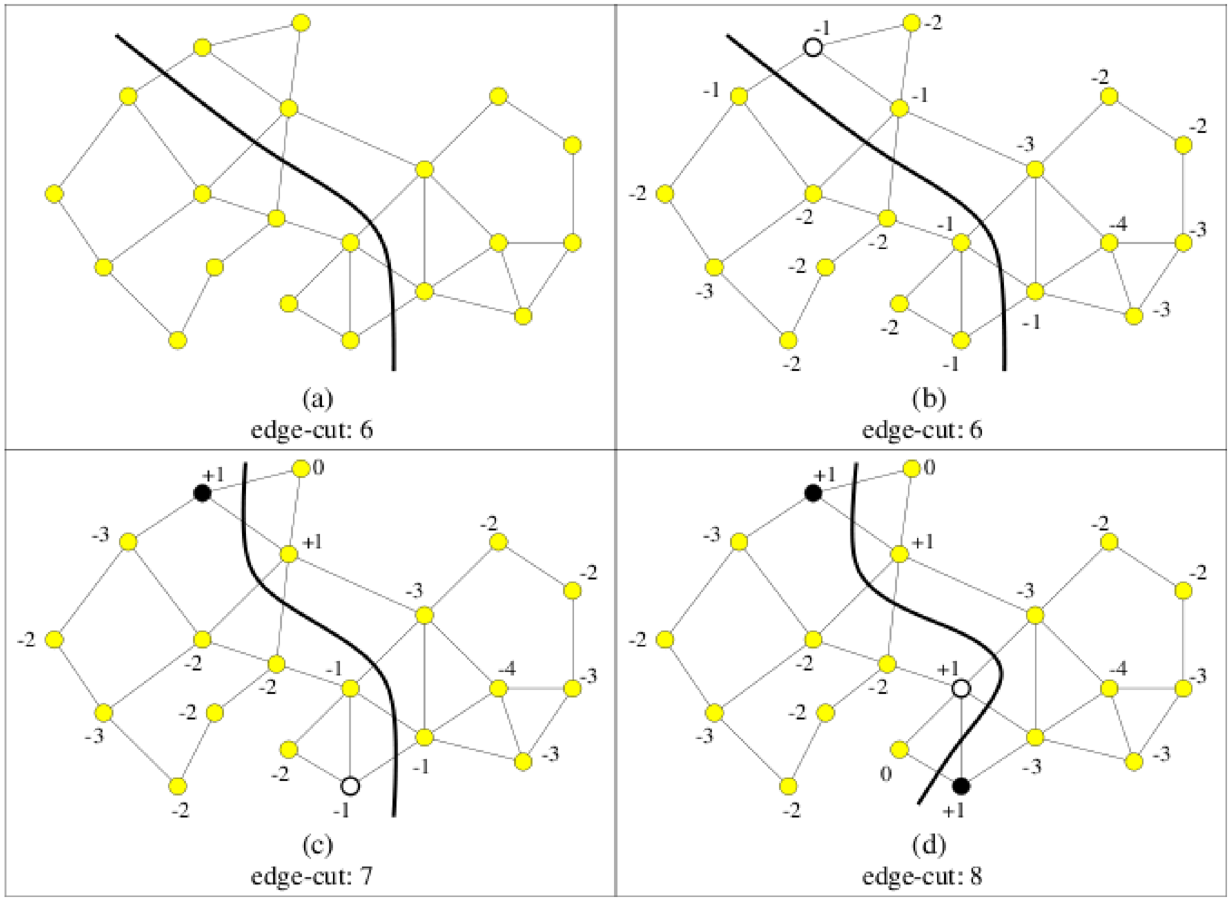
Partition Refinement



Kernighan-Lin Algorithm

Ideas:
Swap best pair of vertices;
Allow the cut size to get worse
to move out of local minima

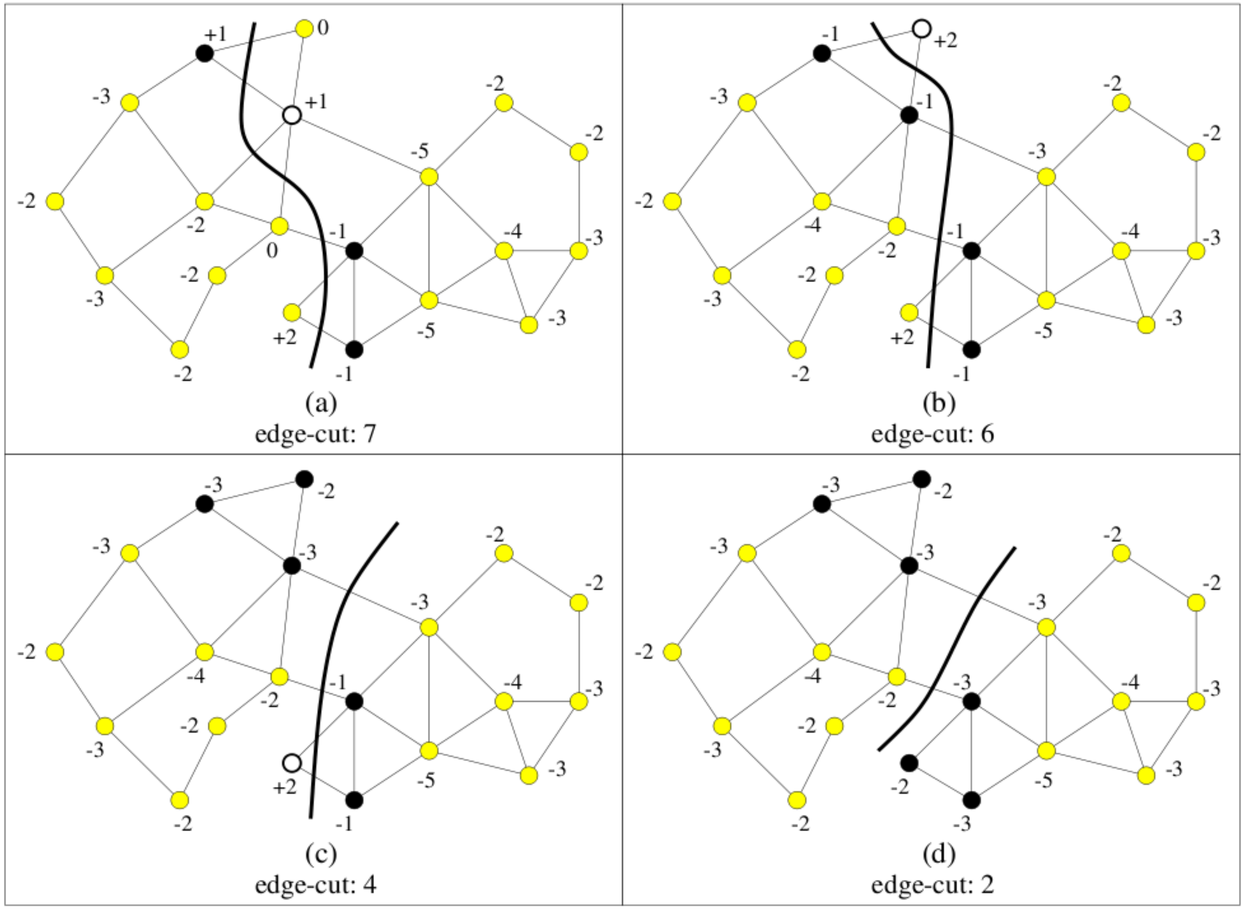
Partition Refinement



Fiduccia-Mattheyses Algorithm

Like KL, but move single vertex (faster)

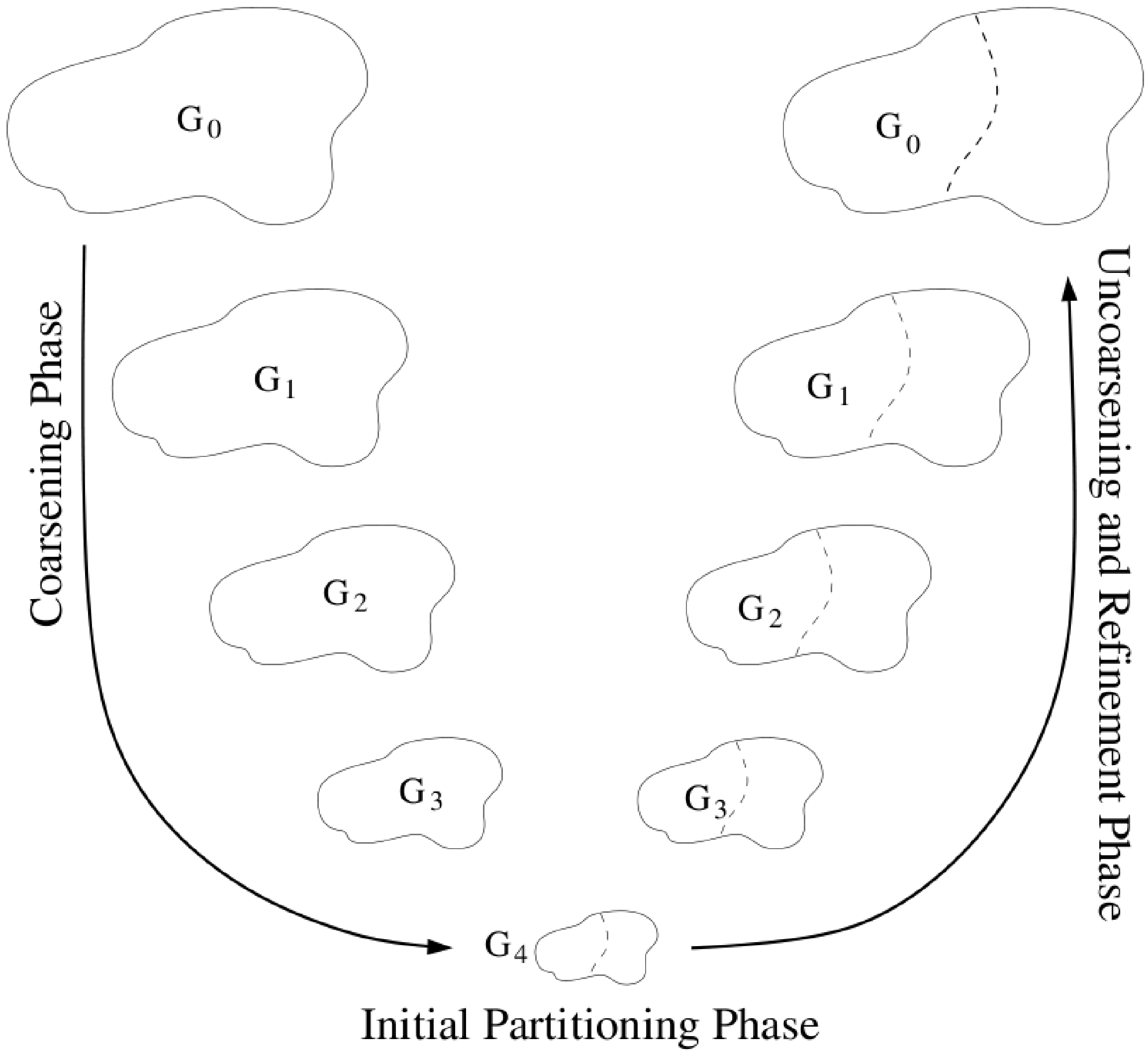
Partition Refinement



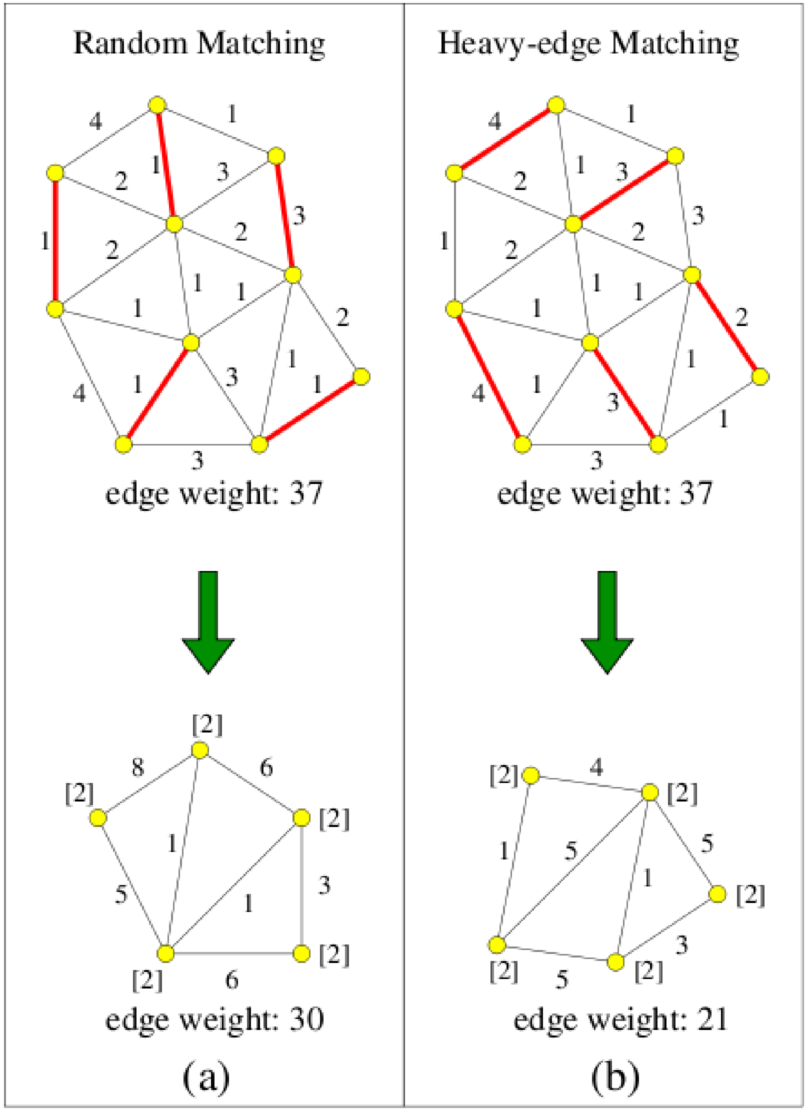
Fiduccia-Mattheyses Algorithm

Like KL, but move single vertex (faster)

Multilevel Techniques



Graph Coarsening via Matching

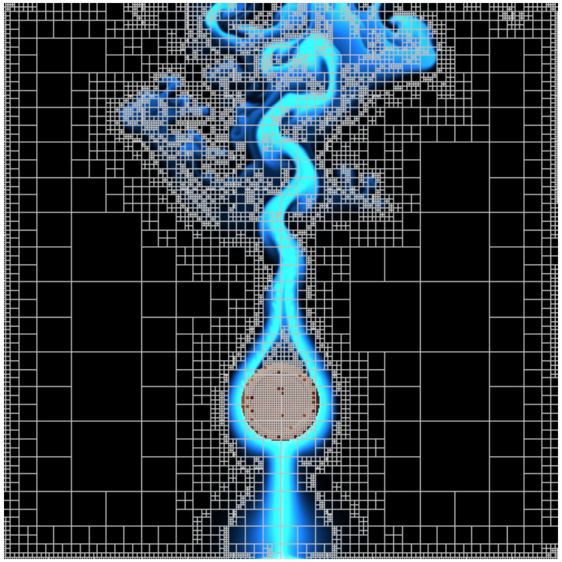


Reduce the exposed edge weight

Partition refinement is faster on coarsened graphs

Load Rebalancing/Repartitioning

- Additional objective:
  minimize the total amount of data that needs to be moved
- (But really want to minimize the maximum amount moved to/from any processor)



Repartitioning Methods

- Repartition from scratch
- Cut-and-paste (bad method)
  - Does not consider edge cut
  - may give disconnected subdomains
- Scratch-remap
  - Repartition from scratch; then somehow map to processors to minimize resulting data movement
- Diffusive schemes
  - Recursive bisection diffusion
  - Adaptive space-filling-curves
  - Formulate and solve an optimization problem

Graph partitioners

- Input is a graph, output is a partitioning
- Many codes: METIS, Chaco, SCOTCH, etc.
- Input graph data structure: edge list for each vertex

What if…

- Work per task cannot be estimated accurately
- Tasks are generated by other tasks

Dynamic scheduling

- Partition problem into large number of tasks, which are put into a queue
- Each node takes a task from the queue when it is idle
- Load is balanced if tasks are small enough
- Does not consider communication
- If the queue is centralized, then it may become a bottleneck (queue can only be accessed by one node at a time)

Work stealing

- Try to reduce the synchronization cost of dynamic scheduling (decentralized dynamic scheduling)
- Each process has its own task queue
- When process runs out of work, it “steals” work from other processes
- Possible to have a “work donating” paradigm, which may be better if one node is overloaded compared to the others; estimate of work can be done on earlier iterations