Computations on Graphs: Partitioning, Load Balancing, and Algorithms

John Mellor-Crummey

Department of Computer Science
Rice University

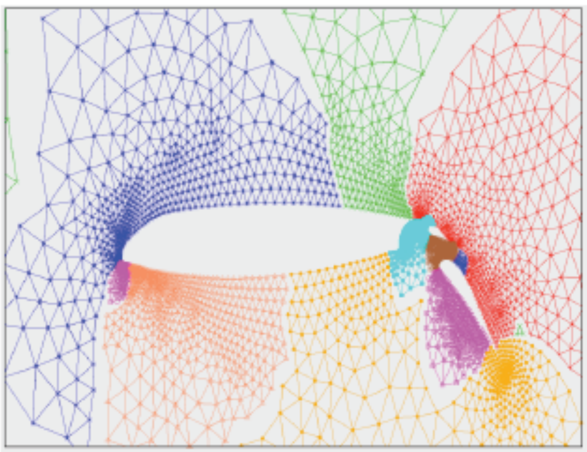
johnmc@rice.edu

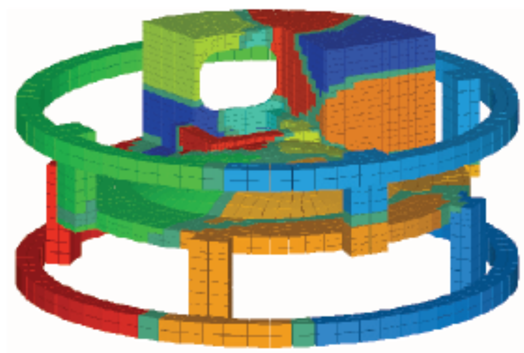
Topics for Today

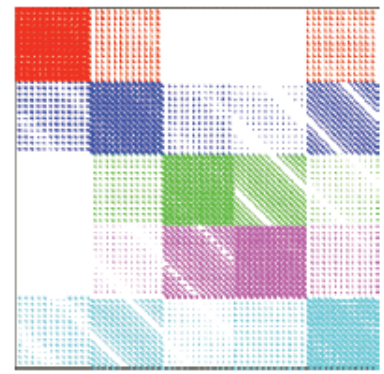
- **Partitioning and load balancing of irregular problems**
  —mostly about graphs, but applies more broadly
- **Computing on graphs**

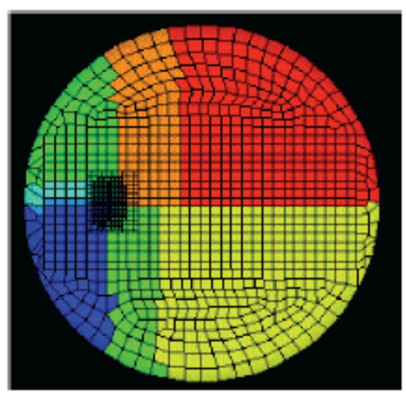
Partitioning and Load Balancing

- **Broad class of problems have irregular structure**
- **Map application data to processors for computing in parallel**
- **Apply to grid points, elements, matrix rows, particles**









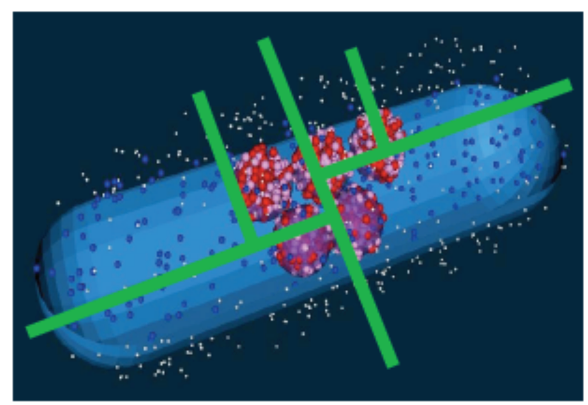


Figure credit: Erik Boman, Cedric Chevalier, Karen Devine. The Zoltan Toolkit – Partitioning, Ordering, and Coloring. Dagstuhl Tutorial. 2009.

Computational Approach

- Distributed memory model
- Data decomposition + “owner computes”
  - —distribute data among the processors
  - —owner performs all computation on its data
  - —data distribution defines work assignment
  - —data dependencies among data items owned by different processors require communication

Kinds of Partitionings

- **Static**
  - —all information available before computation starts

  - —alternatively, could be run as an off-line preprocessing step

- **Dynamic**
  - —information not known until runtime, work changes during computation (e.g., adaptive methods), or locality of objects changes (e.g., particles move)

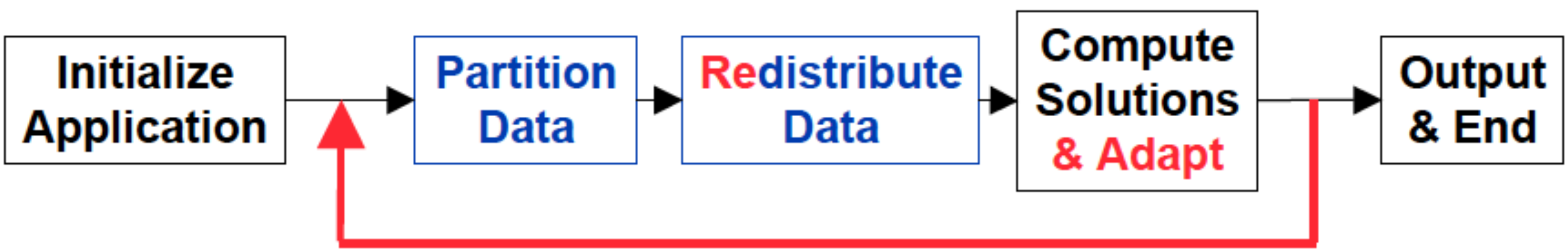


Figure credit: Erik Boman, Cedric Chevalier, Karen Devine. The Zoltan Toolkit – Partitioning, Ordering, and Coloring. Dagstuhl Tutorial. 2009.

Graph Models

- Graphs model the structure of a problem
- Node graph (mesh nodes compute)
  - —vertices = mesh nodes
  - —edges = communication between nodes
- Dual graph (mesh elements compute)
  - —vertices = mesh elements
  - —edges = communication between mesh elements
    - – exchanges take place for every face between adjacent elements

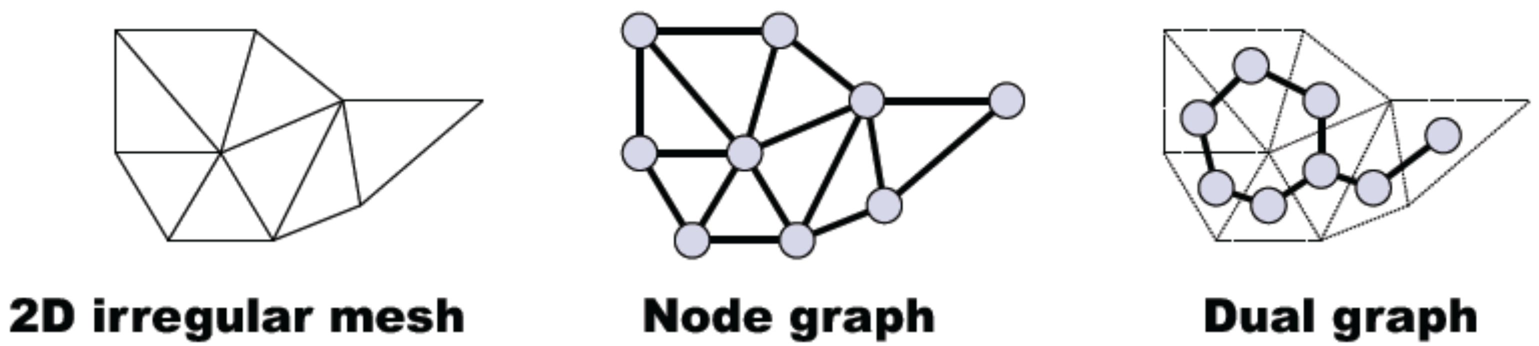
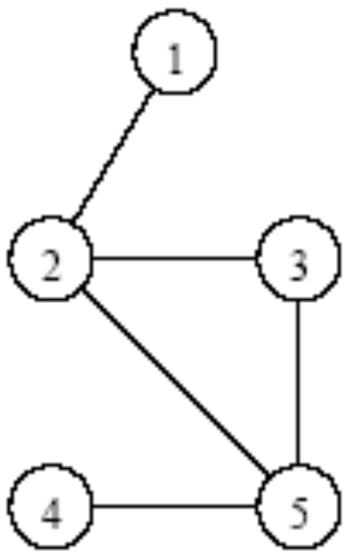


Figure credit: Robert Van Engelen. Graph Partitioning for High Performance Scientific Simulations. Slides. Spring 2009.

Adjacency Matrix for Graph G = (V,E)

- **|V| x |V| matrix**
  - **—matrix element $a_{i,j} = 1$ if nodes *i* and *j* share an edge; 0 otherwise**
  - **—for a weighted graph, $a_{i,j} = w_{i,j}$, the edge weight**
- **Requires $\Theta(|V|^2)$ space**



Undirected graph

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

adjacency matrix is symmetric about the diagonal for undirected graphs

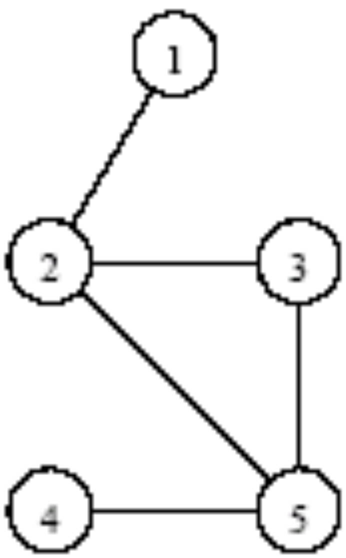
Adjacency matrix representation

Figure Credit: A. Grama et al. Graph Algorithms. Slides accompanying Introduction to Parallel Computing, Second Edition. Addison Wesley, 2003.

Adjacency List for Graph G = (V,E)

- **An array *Adj[1..|V|]* of lists**
  —each list *Adj[v]* is a list of all vertices adjacent to *v*
- **Requires $\Theta(|E|)$ space**

Adjacency list representation



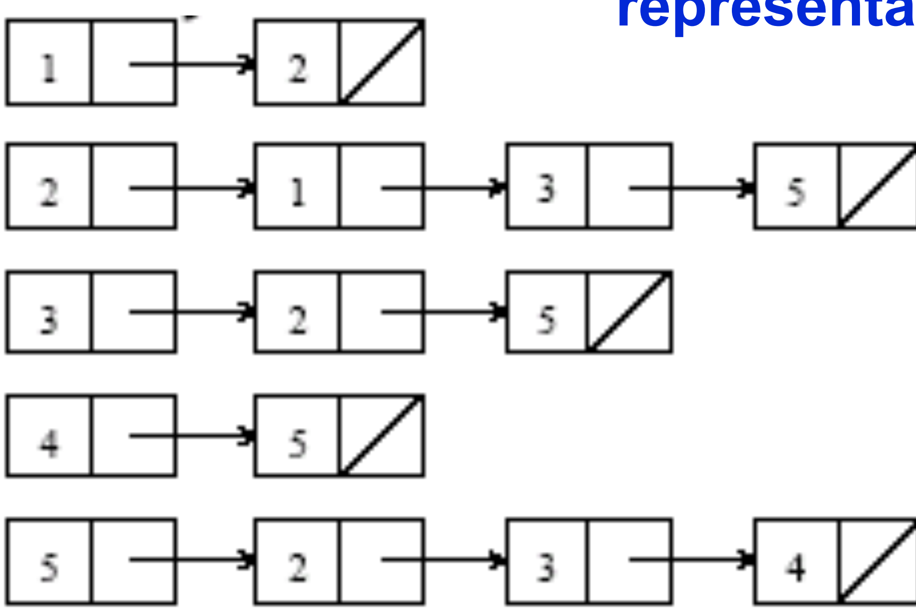
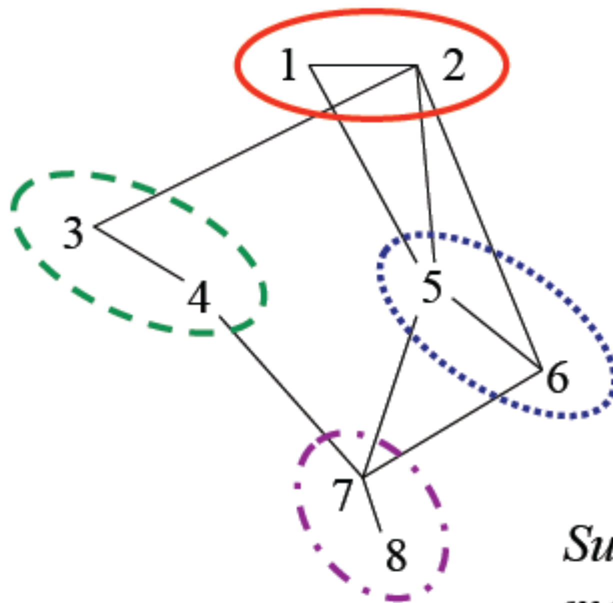


Figure Credit: A. Grama et al. Graph Algorithms. Slides accompanying Introduction to Parallel Computing, Second Edition. Addison Wesley, 2003.

Partitioning using an Adjacency Matrix

	1	2	3	4	5	6	7	8
1	1	1			1			
2	1	1	1		1	1		
3		1	1	1				
4			1	1			1	
5	1	1			1	1	1	
6		1			1	1	1	
7				1	1	1	1	1
8							1	1



Subdomains are mapped to the processors

- **Can reorder rows and columns of the matrix**
  - —non-zeros outside of blocks require communication
- **An optimal partition of the graph for parallel computation has**
  - —equal number of vertices in subdomains
  - —lowest number of edges between subdomains

Figure credit: Robert Van Engelen. Graph Partitioning for High Performance Scientific Simulations. Slides. Spring 2009.

Finite Element Mesh Example

NASA airfoil finite-element mesh, 4253 grid points

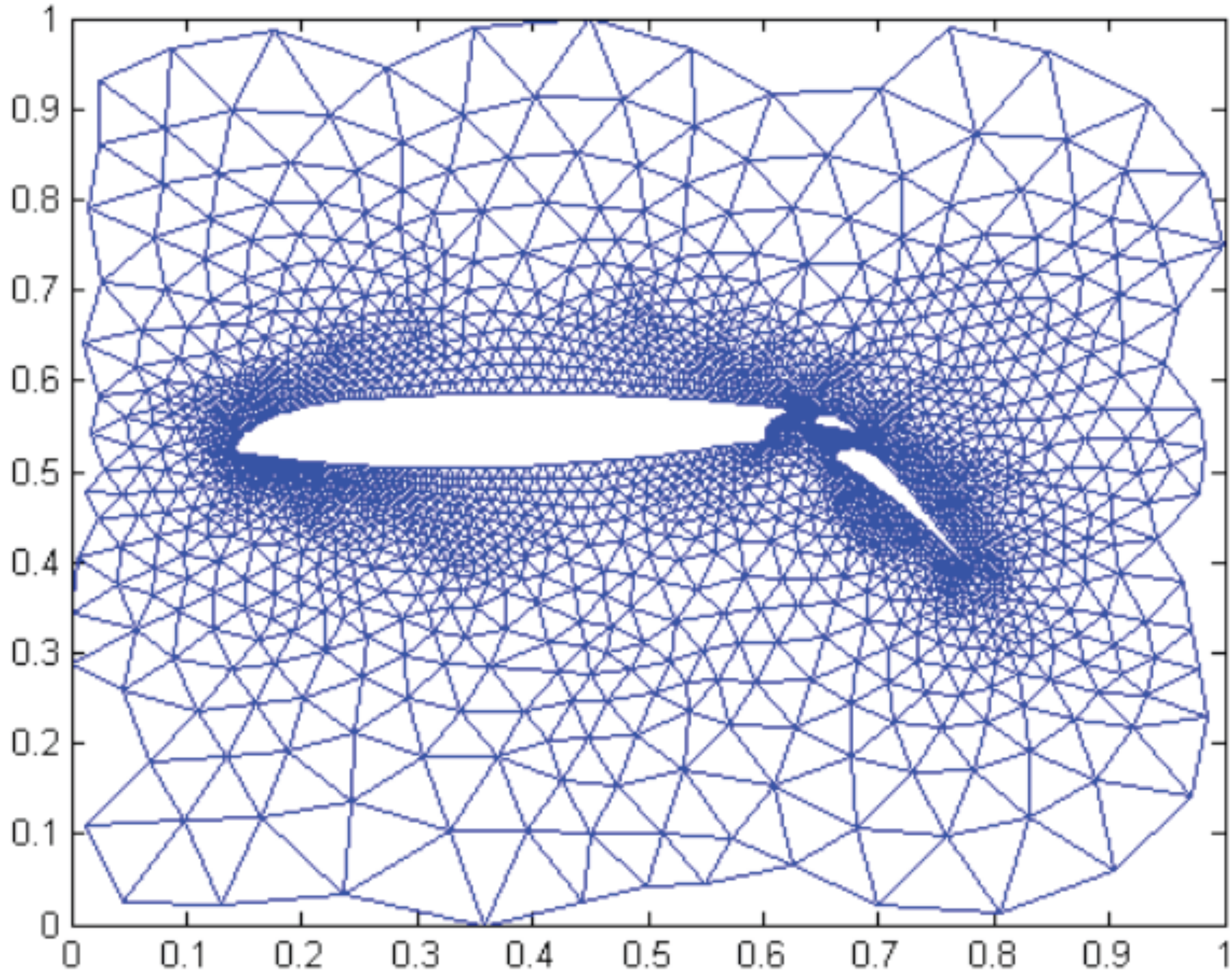


Image source: MATLAB 7.5 NASA airfoil demo

Adjacency Matrix and Reverse Cuthill-McKee Reordering

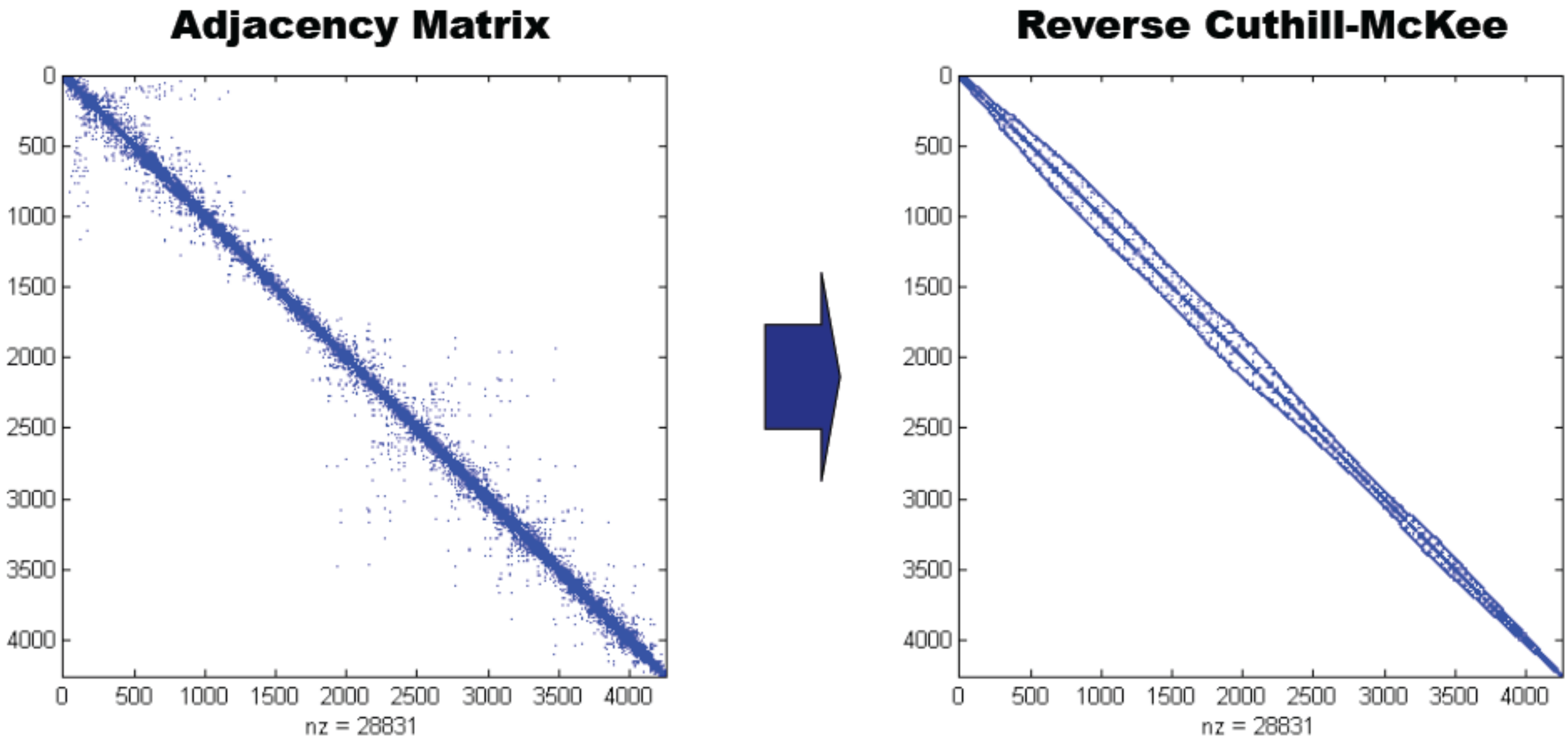


Image source: MATLAB 7.5 NASA airfoil demo

E. Cuthill and J. McKee. *Reducing the bandwidth of sparse symmetric matrices* In Proc. 24th Nat. Conf. ACM, pages 157–172, 1969.

J. A. George and J. W-H. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, 1981

Reverse Cuthill-McKee

- **Cuthill-McKee**
  - —begin with a peripheral vertex
  - —partition vertices into levels until all nodes are exhausted
    - – level set K contains all vertices adjacent to all nodes in level K-1
  - —list nodes in each level in increasing degree
    - – only difference with conventional breadth-first search
- **Reverse order of the above**
  - —reduces fill-in when using GE

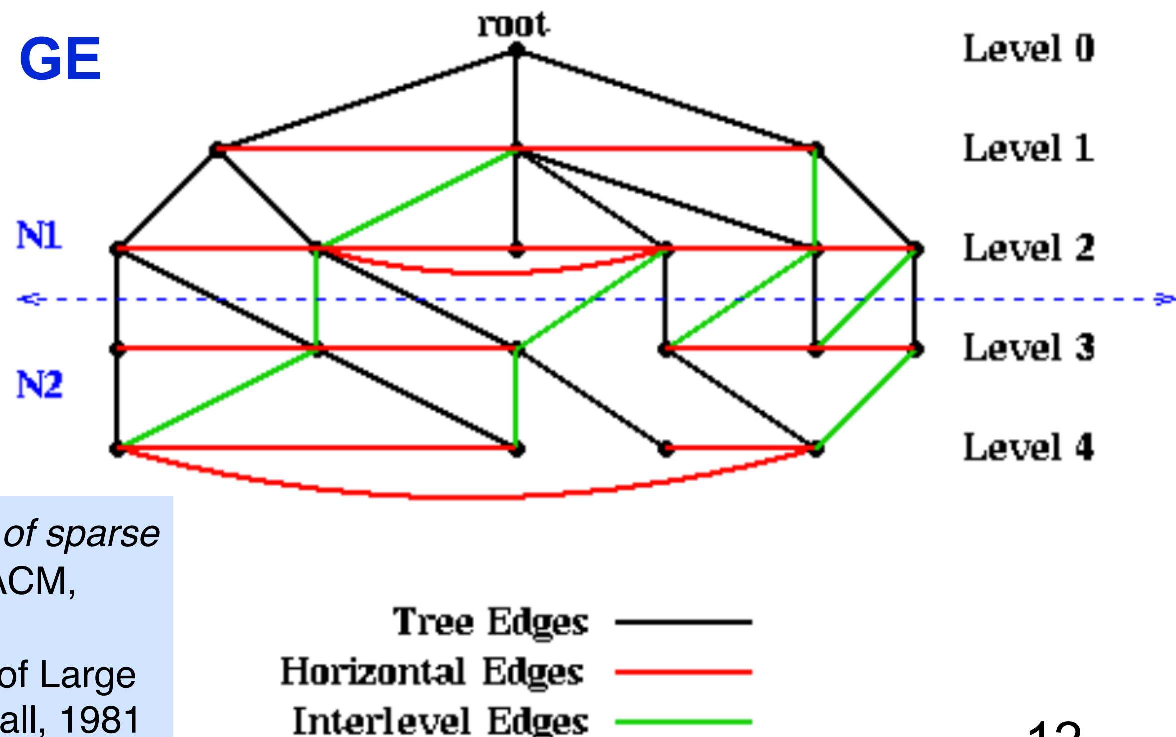


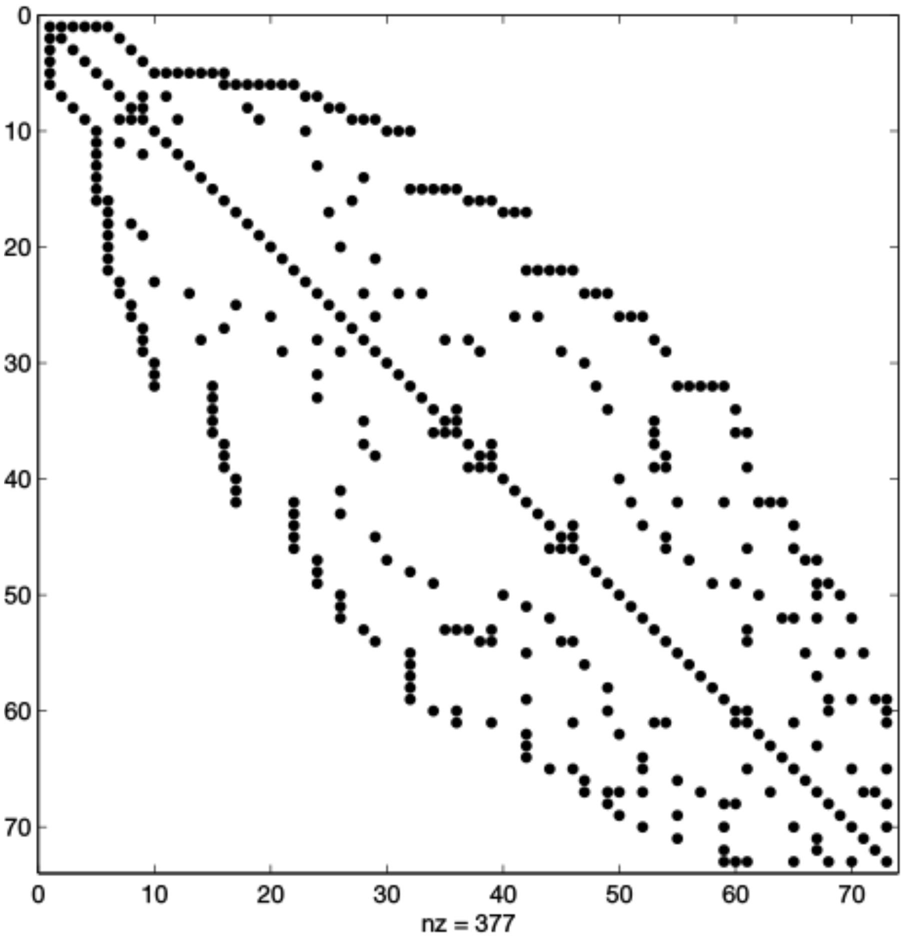
Figure credit: Marsha Berger and Andreas Klöckner. Lecture 12: Load balancing and partitioning. G63.2011.002/G22.2945.001. NYU. November 16, 2010.

E. Cuthill and J. McKee. *Reducing the bandwidth of sparse symmetric matrices* In Proc. 24th Nat. Conf. ACM, pages 157–172, 1969.

J. A. George and J. W-H. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, 1981

Cuthill-McKee vs. Reverse Cuthill-McKee Ordering

Cuthill-McKee



Reverse Cuthill-McKee

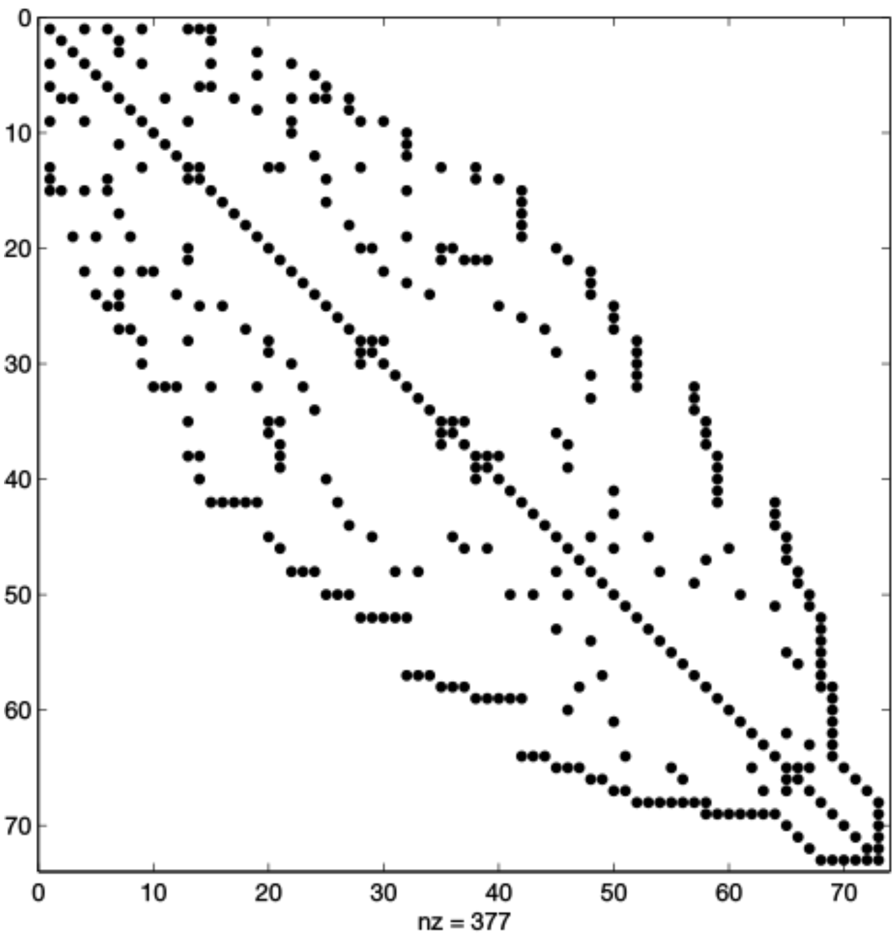
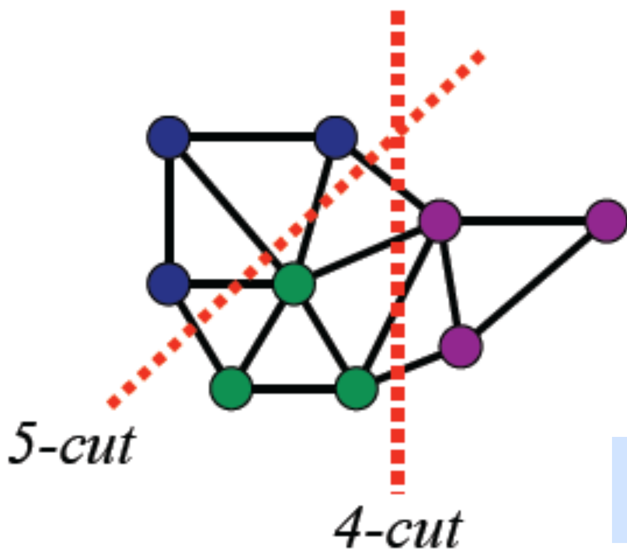


Figure credit: https://en.wikipedia.org/wiki/Cuthill–McKee_algorithm

Goals of Partitioning

- **Balance constraint**
  - —balance computational load such that each processor has the same execution time
  - —balance storage such that each processor has the same storage requirements
- **Minimum edge cut**
  - —minimize communication volume between subdomains, along edges of the mesh
- **Note: communication to computation ratio comes from both partitioning and the algorithm**



Example 3-way partition with edge-cut = 9

Figure credit: Robert Van Engelen. Graph Partitioning for High Performance Scientific Simulations. Slides. Spring 2009.

Graph Partitioning Problem

- Let $G = (V,E)$ be a weighted undirected graph with weight functions $w_V: V \rightarrow R^+$, $w_E: E \rightarrow R^+$
- K-way graph partitioning problem
  - —split V into K disjoint subdomains S_j $j = 1, \ldots, k$ such that
    - balance constraint

      $\sum_{v \in S_j} w_V(v)$ is roughly equal, for all $j = 1, \ldots, k$
    - minimum edge cut

      $\sum_{(u,v) \in E \text{ such that } u \in S_i \text{ and } v \in S_j} w_E((u,v))$ is minimal
- Weight functions are defined that
  - —w_V models computational work
  - —w_E models communication
- Can add subdomain weights to improve mapping to heterogeneous nodes or processors

Static Graph Partitioning

- Geometric techniques
  - —recursive coordinate bisection
  - —recursive inertial bisection
  - —space filling curves
- Combinatorial techniques
  - —Kernighan-Lin
- Multi-level schemes
  - —multilevel recursive bisection
  - —multilevel k-way partitioning

Geometric Partitioning Techniques

- Goal: group together vertices that are nearby in space
- When are these methods applicable
  - —when coordinate system exists or can be constructed
- How:
  - —partition based on coordinate information
    - – may consider vertex weights as well
  - —recursively bisect mesh into increasingly smaller subdomains
- Properties
  - —typically fast
  - —have no concept of edge cut, so no communication optimization
  - —may suffer from disconnected meshes in complex subdomains

Recursive Coordinate Bisection (RCB)

- **Geometric method**
  - —compute centers of mass of mesh elements
  - —project onto axis of longest dimension
  - —bisect the list of centers
  - —repeat recursively
- **Strength: fast, easy to parallelize**
- **Weakness: low quality partitionings**

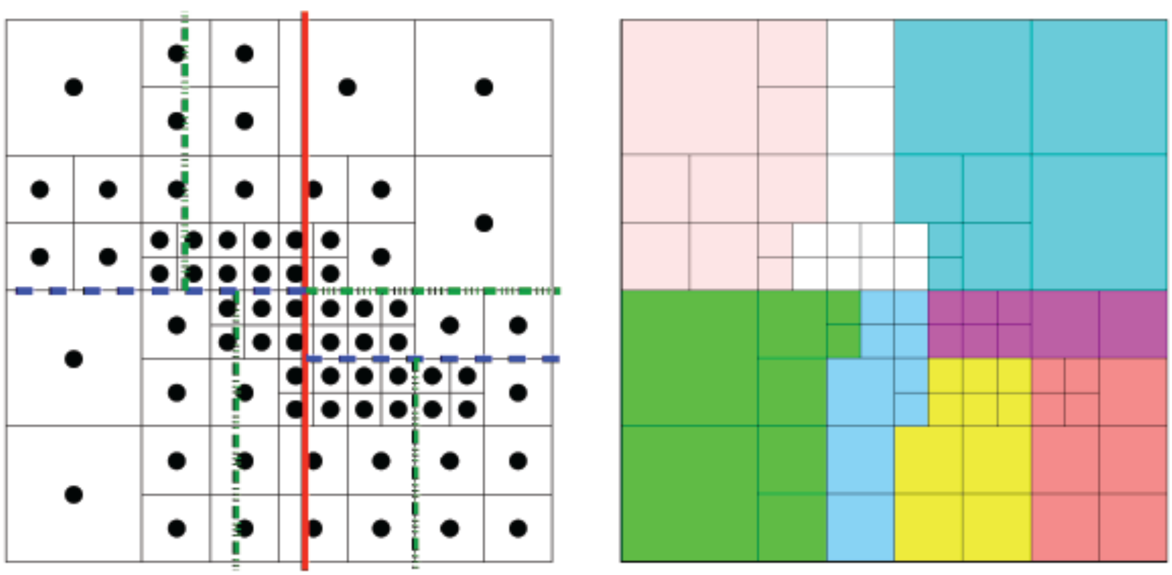
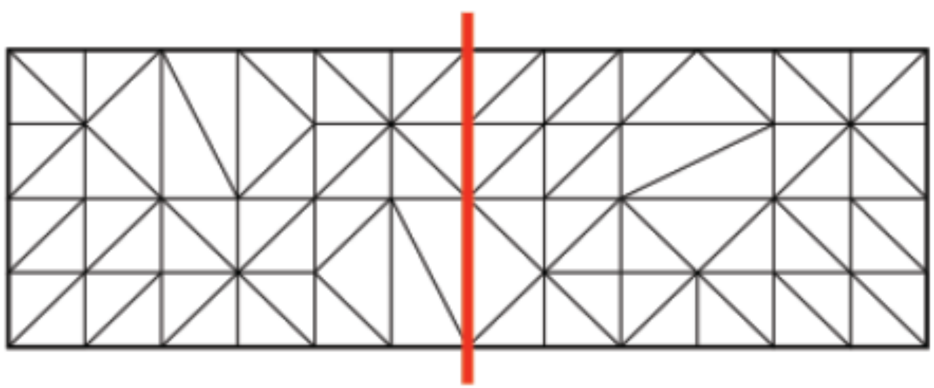


Figure Credit: K. Schloegel, et al. Graph Partitioning for High Performance Scientific Simulations. In CRPC Parallel Computing Handbook. Morgan Kauffman, 2000.

Recursive Coordinate Bisection (RCB)

- **Bisect mesh normal to the longest dimension**
  - —yields smaller subdomain boundaries
  - —typically reduces communication volume

Bisected normal to the x-axis



Bisected normal to the y-axis

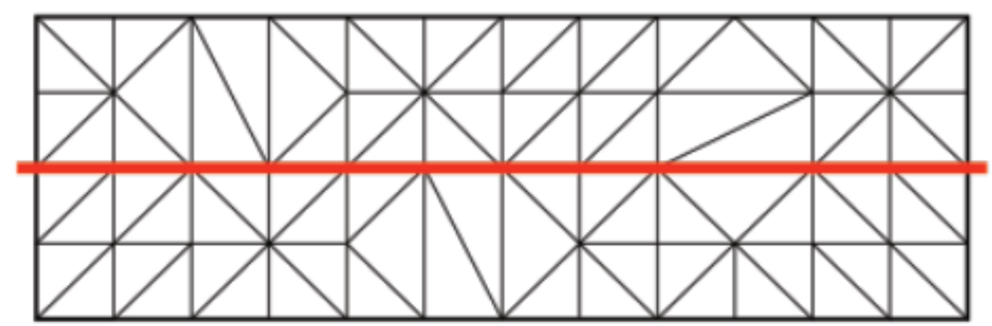
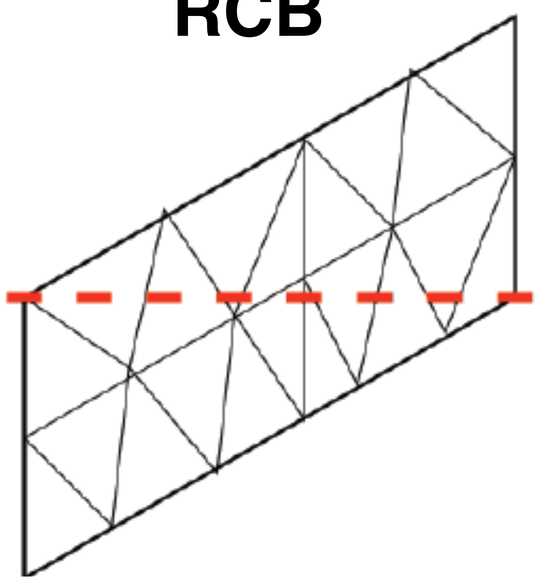


Figure Credit: K. Schloegel, et al. Graph Partitioning for High Performance Scientific Simulations. In CRPC Parallel Computing Handbook. Morgan Kauffman, 2000.

Recursive Intertial Bisection (RIB)

- **Orient bisection to minimize the subdomain boundary**
- **Mesh elements are converted into point masses**
  - **—compute principal inertial axis of the mass distribution**
  - **—bisect orthogonal to the principal inertial axis**
  - **—repeat recursively**
- **Fast and better quality than recursive coordinate bisection**

RCB



RIB

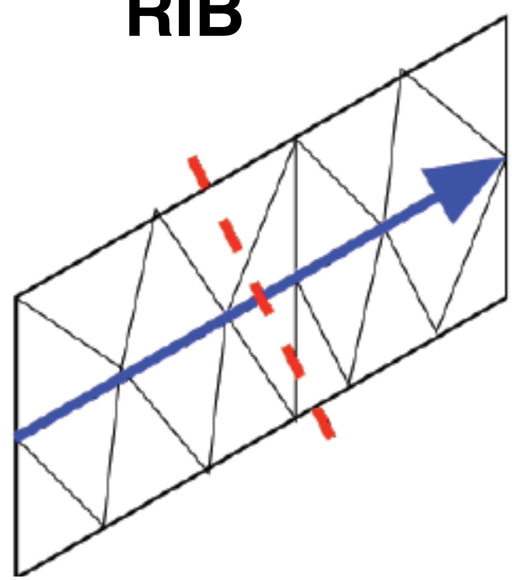
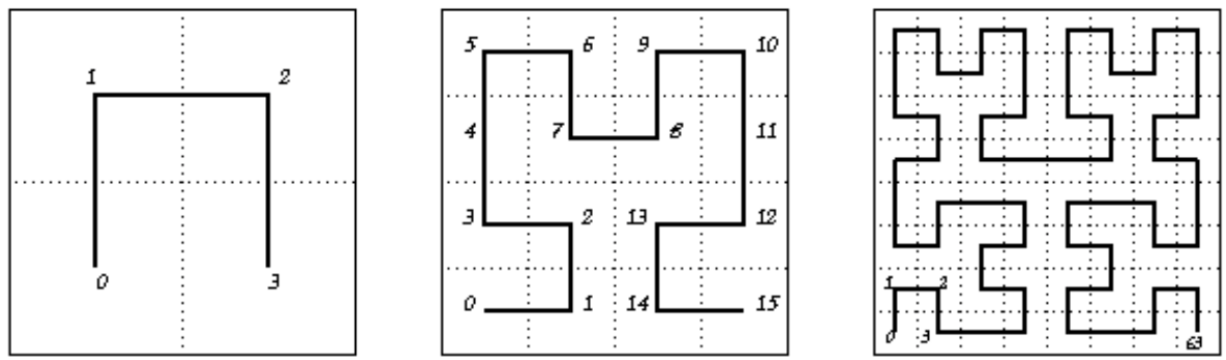


Figure Credit: K. Schloegel, et al. Graph Partitioning for High Performance Scientific Simulations. In CRPC Parallel Computing Handbook. Morgan Kauffman, 2000.

Space-filling Curves

- RCB and RIB only consider a single dimension at a time
- Space-filling curve techniques linearly order a multidimensional mesh (nested hierarchically, preserves locality)

Peano-Hilbert curve



Morton / Z-order curve

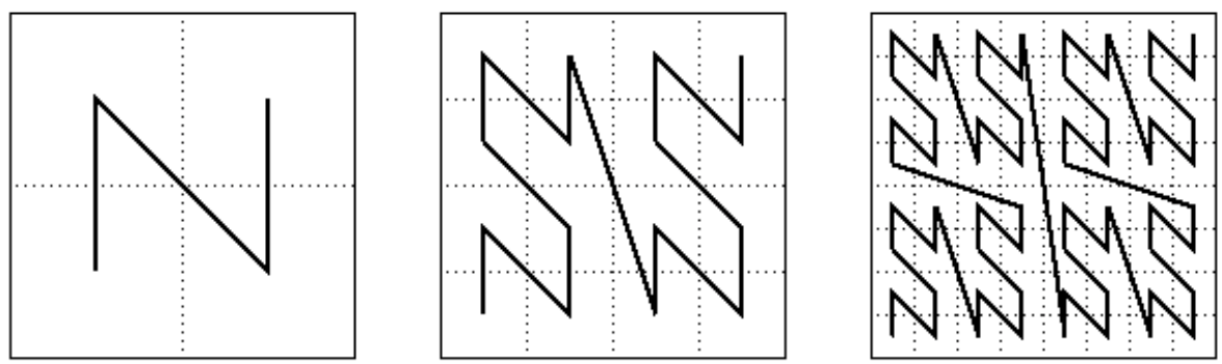
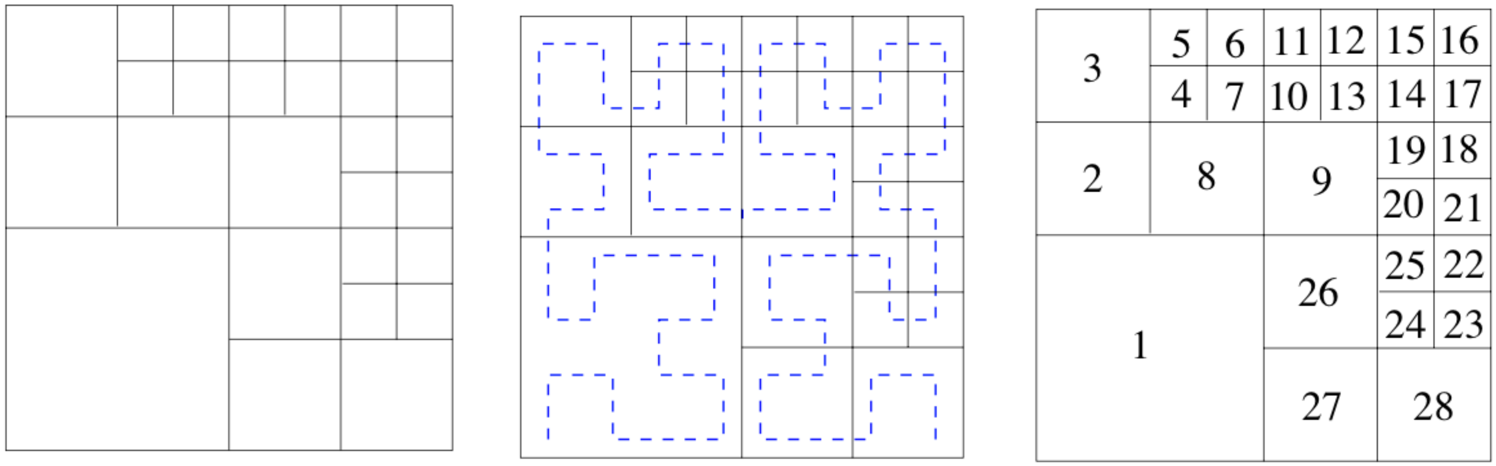


Figure credit: http://www.dcs.bbk.ac.uk/~jkl/BNCOD2000/img1.gif

Space-filling Curves (SFC)

Applies even to adaptively refined meshes and particles



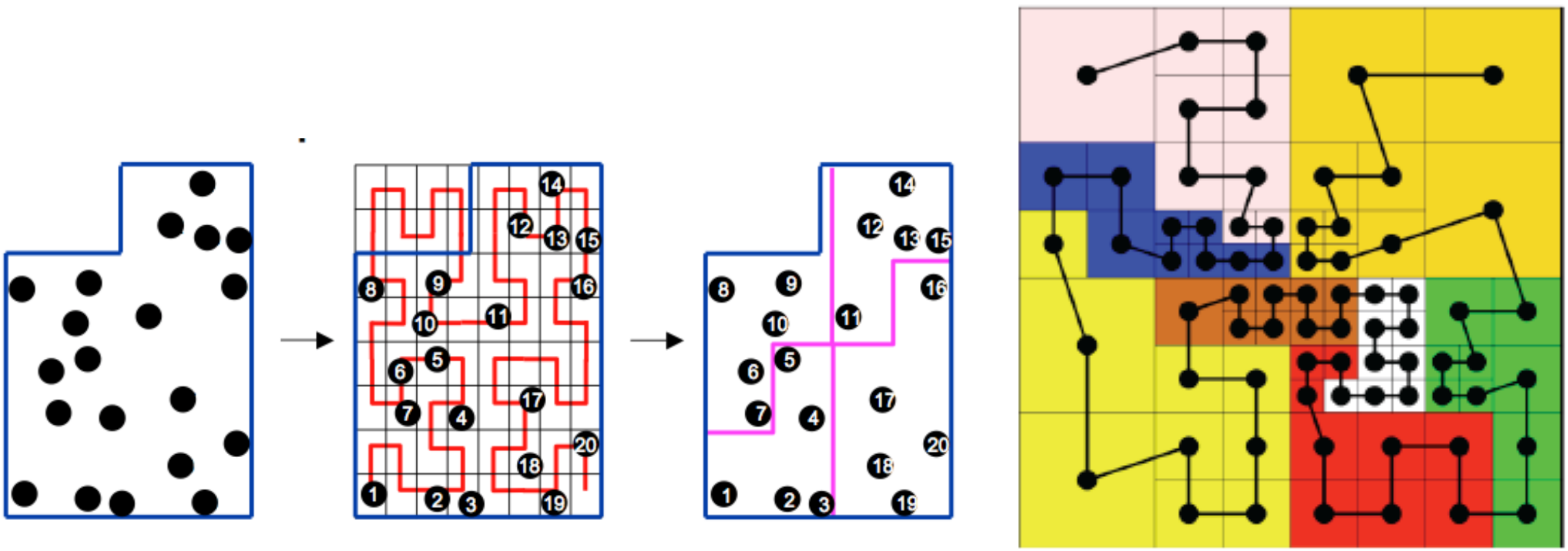


Figure credit: Figure Credit: K. Schloegel, et al. Graph Partitioning for High Performance Scientific Simulations. In CRPC Parallel Computing Handbook. Morgan Kauffman, 2000. Marsha Berger and Andreas Klöckner. Lecture 12: Load balancing and partitioning. G63.2011.002/G22.2945.001. NYU. November 16, 2010.

Space-filling Curve Properties

- **Strengths**
  - —broadly applicable: particles, adaptive mesh refinement, …
  - —generalizes to uneven workloads - incorporate weights
  - —dynamic on-the-fly partitioning for any number of nodes
  - —good for cache performance
- **Weaknesses**
  - —need coordinates
  - —partitions may not be compact

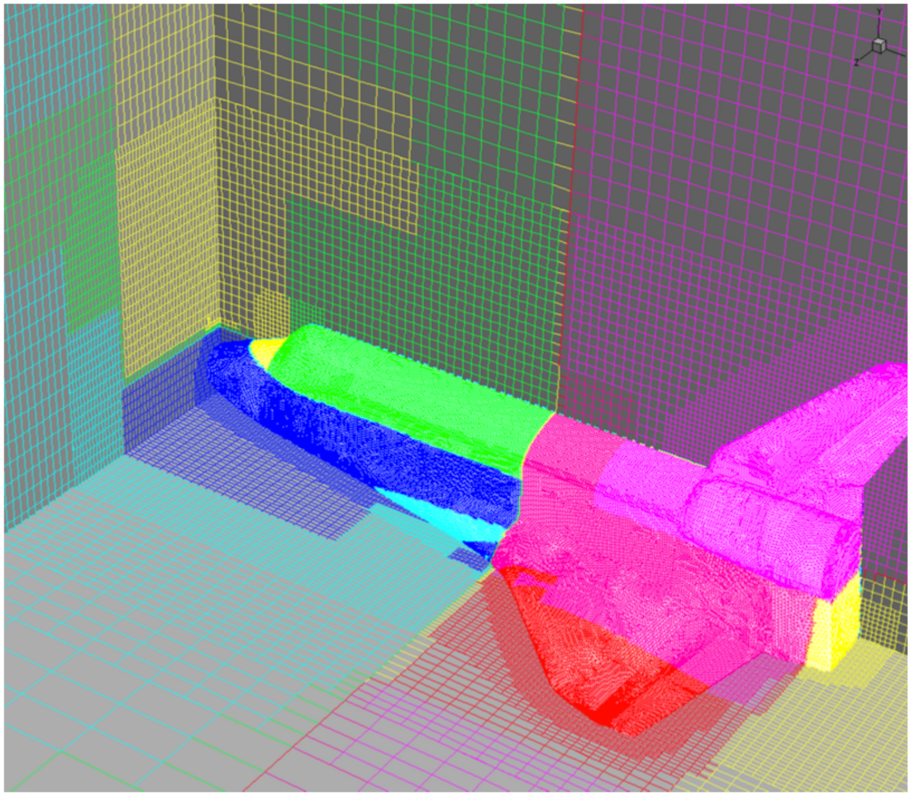


Figure credit: Marsha Berger and Andreas Klöckner. Lecture 12: Load balancing and partitioning. G63.2011.002/G22.2945.001. NYU. November 16, 2010.

Combinatorial Partitioning Techniques

- Geometrical techniques group vertices that are spatially close, whether they are connected or not
- Combinatorial partitioning techniques use adjacency information to group together vertices that are highly connected
- Properties
  - smaller edge cuts
  - reasonably fast
  - harder to parallelize

Levelized Nested Dissection (LND)

- **Select initial vertex v_0, preferably a peripheral vertex**
- **For each vertex, compute the distance to v_0 using a breadth first search starting from v_0**
- **When half of the vertices have been assigned, split the graph into two parts: assigned and unassigned**
- **Can repeat with different vertices as v_0 to improve edge cut**

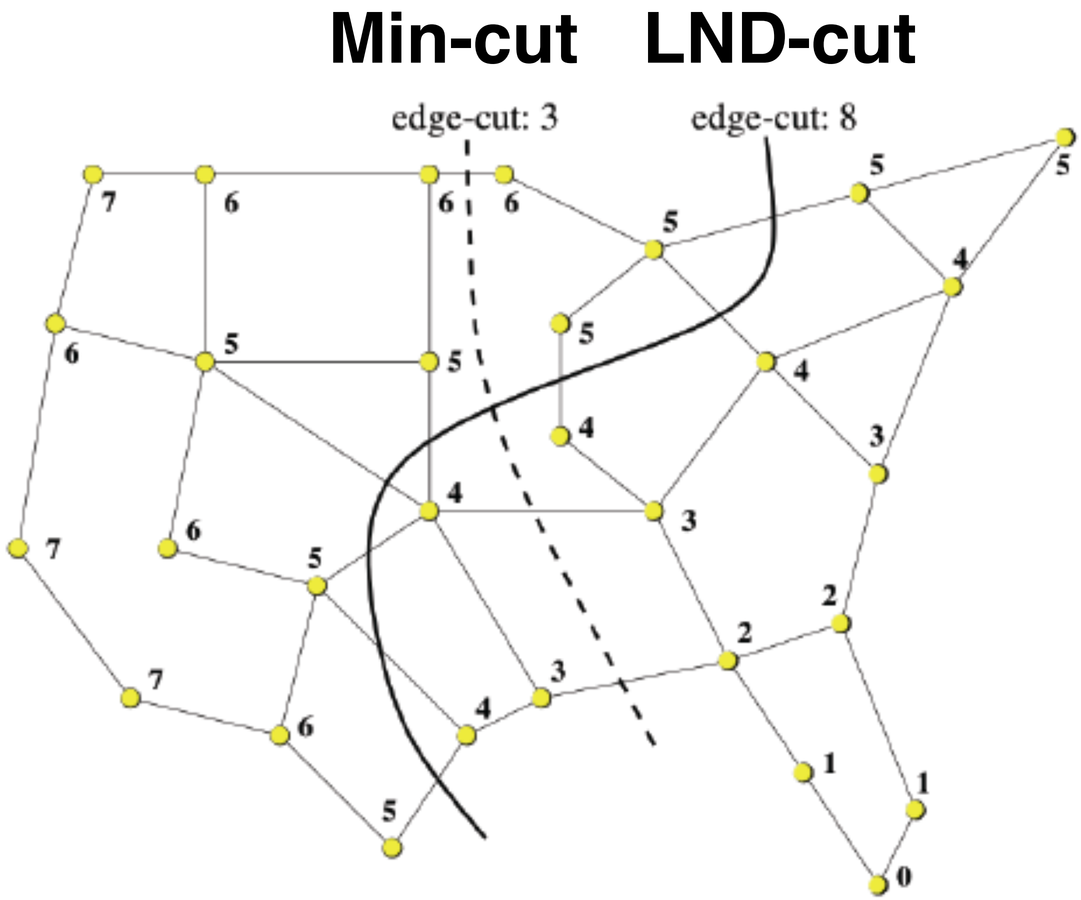


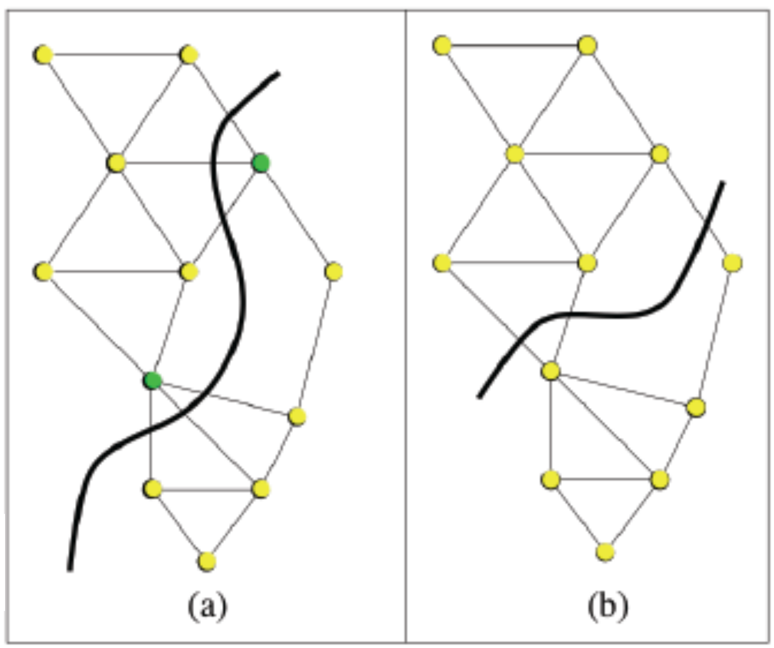
Figure Credit: K. Schloegel, et al. Graph Partitioning for High Performance Scientific Simulations. In CRPC Parallel Computing Handbook. Morgan Kauffman, 2000.

Kernighan-Lin (KL) Partition Refinement

- Given: partition of vertices into two disjoint sets A and B
- Idea: find X⊆A and Y⊆B such that swapping X to B and Y to A yields the greatest reduction in edge cut
- Finding optimal X and Y is intractable
- Kernighan-Lin performs multiple passes over V
- Each pass swaps two vertices, one from A and one from B

Before KL pass

Edge cut 6



Edge cut 3

After KL pass

Figure Credit: K. Schloegel, et al. Graph Partitioning for High Performance Scientific Simulations. In CRPC Parallel Computing Handbook. Morgan Kauffman, 2000.

Multi-level Schemes

Strategy

- —recursively coarsen graph in downward pass
- —partition the coarsest graph
- —refine the partition when unwinding each level of the recursion

If the graph is already partitioned, take that into account

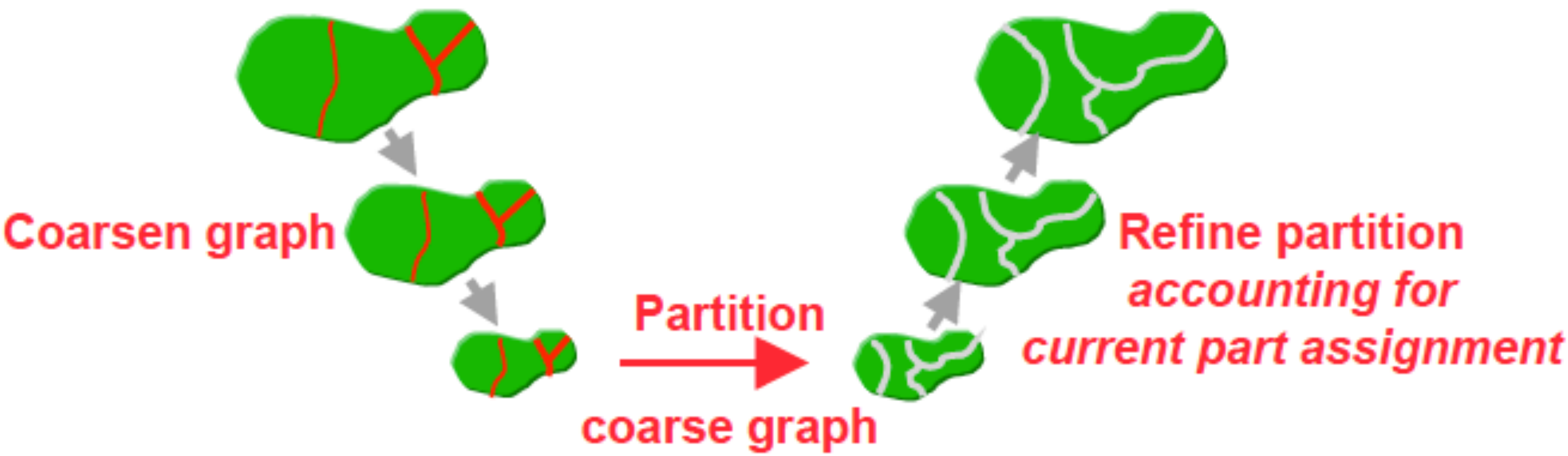


Figure credit: Erik Boman, Cedric Chevalier, Karen Devine. The Zoltan Toolkit – Partitioning, Ordering, and Coloring. Dagstuhl Tutorial. 2009.

Multi-level Schemes

- **Coarsening collapses pairs of vertices**
- **Different coarsening strategies**
  - **—pick random pairs**
  - **—pick heavy edge pairs**
- **Partition coarsest (smallest) graph using recursive bisection**
- **Use refinement, e.g. Kernighan-Lin on uncoarsened graph**

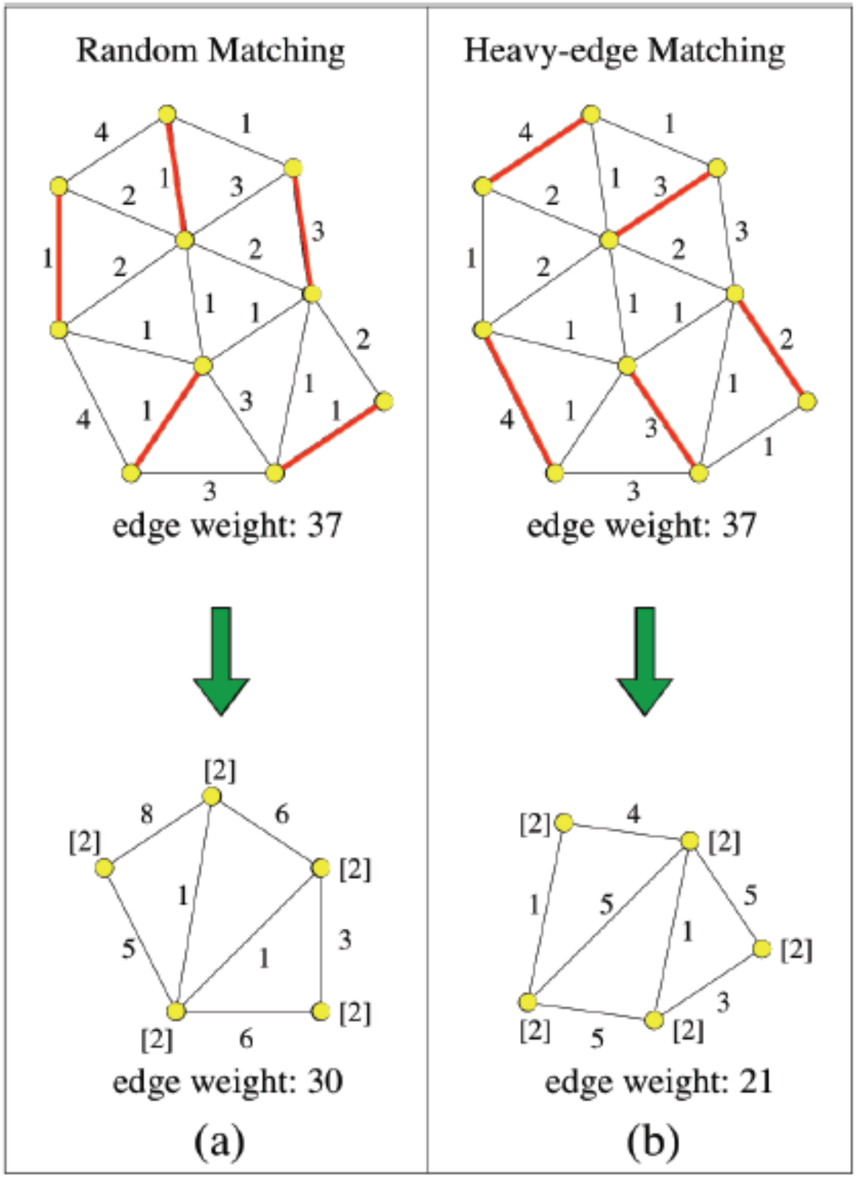


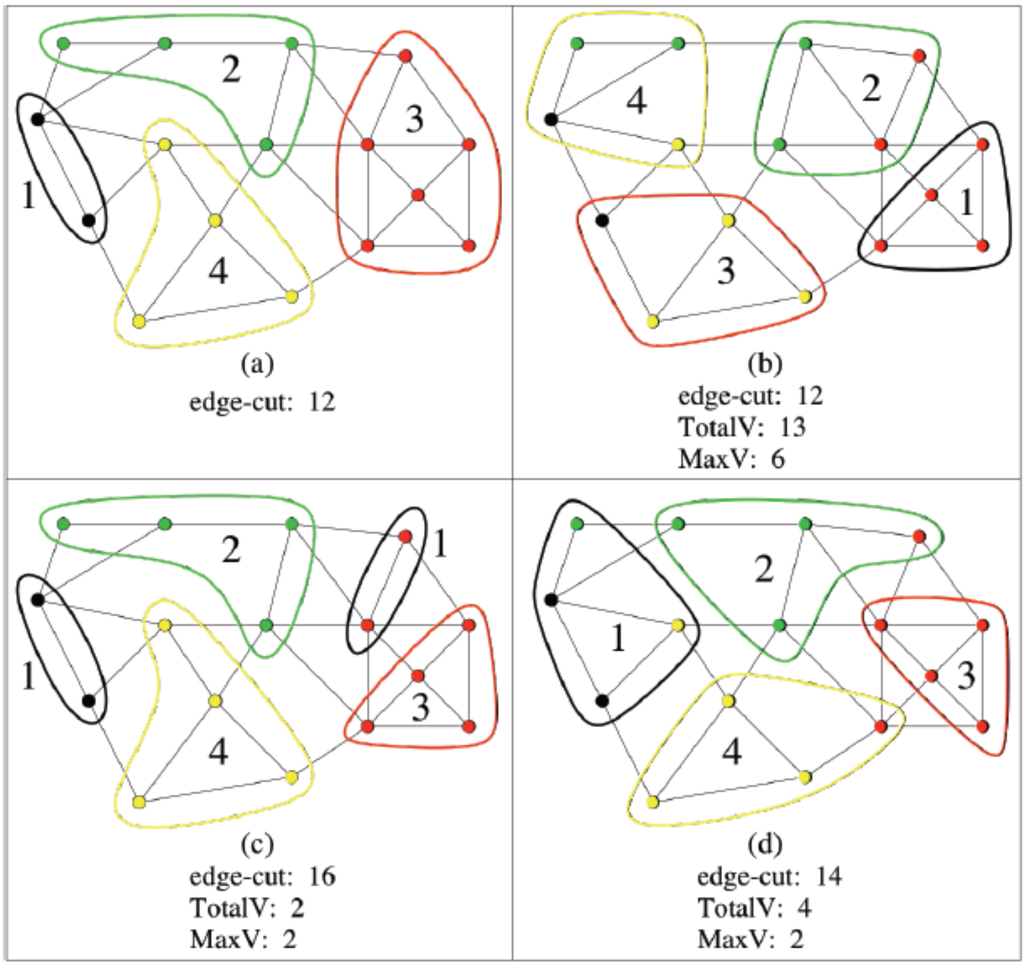
Figure Credit: K. Schloegel, et al. Graph Partitioning for High Performance Scientific Simulations. In CRPC Parallel Computing Handbook. Morgan Kauffman, 2000.

Repartitioning Methods

- Partition from scratch
- Incremental repartition methods
  - —cut and paste
  - —diffusion-based methods

Imbalanced partitioning (vertex weights=1)

Repartitioning from scratch



Cut-and-paste repartitioning

Diffusive repartitioning

Figure credit: Robert Van Engelen. Graph Partitioning for High Performance Scientific Simulations. Slides. Spring 2009.

Comparing Repartitioning Methods

- **Scratch-remap results in higher distribution costs compared to incremental methods that use local perturbations**
- **Incremental partitioning with cut-and-paste**

  —moves fewest vertices between subdomains to restore balance
- **Incremental diffusion-based methods**

Imbalanced partitioning | **Incremental partitioning** | **Scratch-remap partitioning**

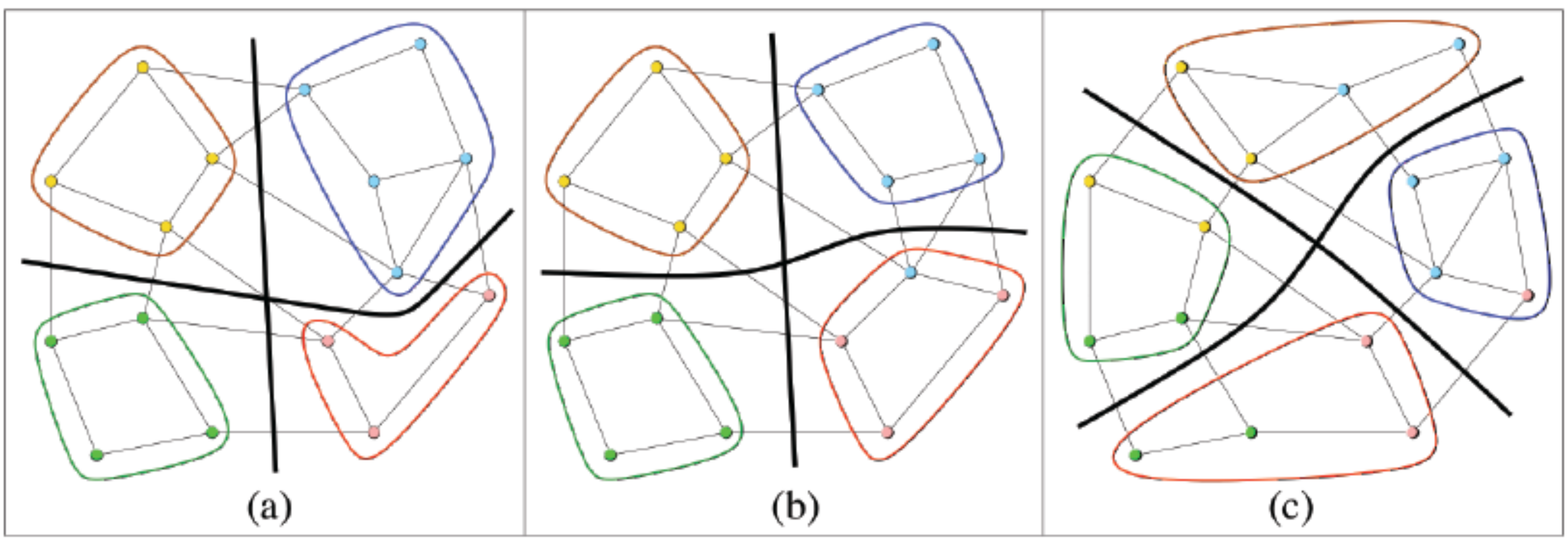
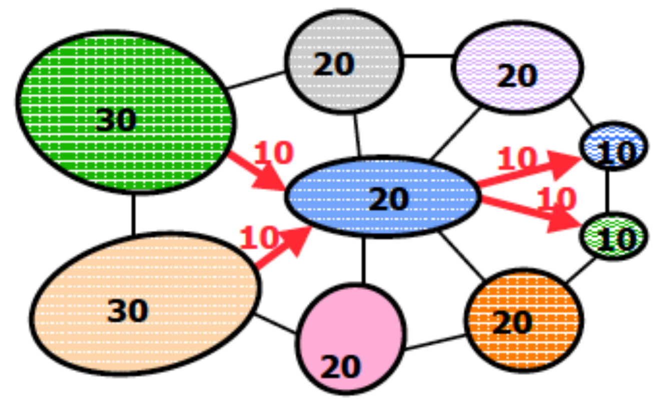
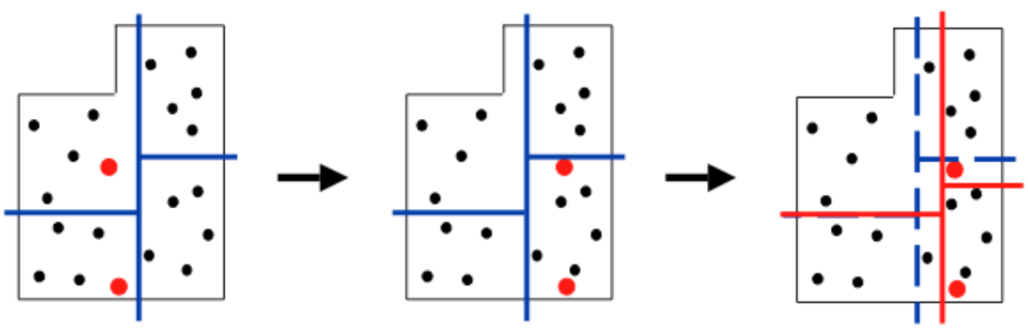


Figure credit: Robert Van Engelen. Graph Partitioning for High Performance Scientific Simulations. Slides. Spring 2009.

Diffusion-based Methods

- **Address two questions**
  - —how much work should be transferred
  - —which tasks should be transferred
- **Attempt to minimize the difference between original and final repartitioning by making incremental changes**
- **Global diffusion schemes consider entire graph**
  - —recursive bisection diffusion partitioners
  - —adaptive space-filling curve partitioners





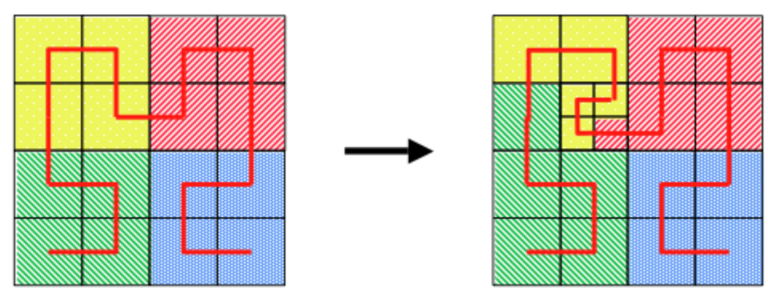


Figure credit: M Berger et al. Lecture 12: Load balancing and partitioning. G63.2011.002/G22.2945.001. NYU. November 16, 2010. E. Boman et al. The Zoltan Toolkit – Partitioning, Ordering, and Coloring. Dagstuhl Tutorial. 2009.

Computing on Graphs

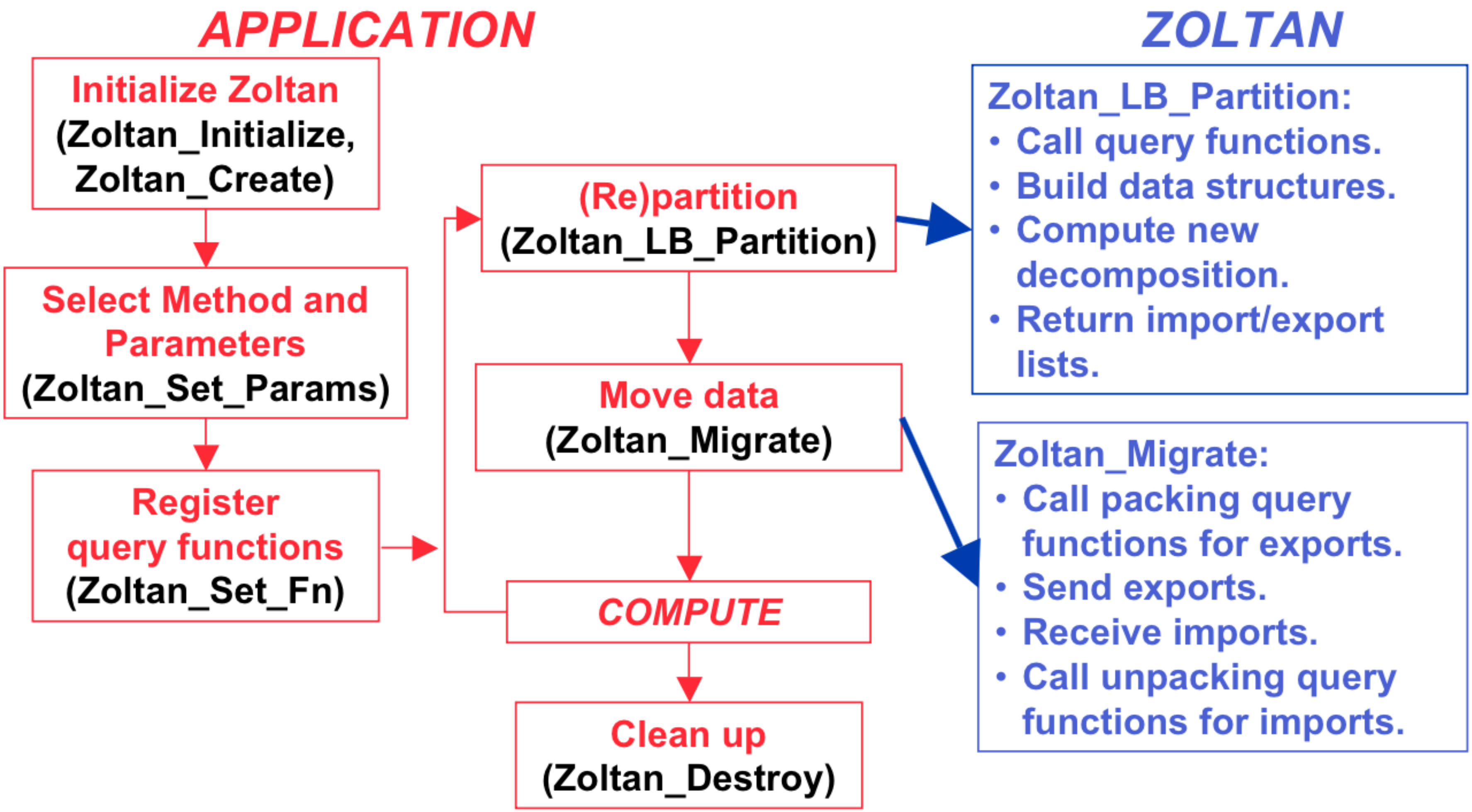
- Precompute your communication schedule
  - —what has to be communicated to whom
- Locally partition data into separate parts
  - —elements that can be computed with local information only
  - —elements that require non-local elements to compute
- Begin communicating your data to neighbors
  - —post non-blocking receives
  - —issue non-blocking sends
- Perform your local computation
- Wait for communication from neighbors
- Perform computation on non-local data

Sandia's Zoltan Library

- Dynamic load-balancing and parallel partitioning tools that distribute data over sets of processors
- Data migration tools for redistribution
- Parallel graph/matrix ordering algorithms
- Parallel graph coloring algorithms
- Distributed data directories that efficiently locate off-processor data
- An unstructured communication package that greatly simplifies interprocessor communication
- A dynamic memory debugging package for use on parallel systems

http://www.cs.sandia.gov/zoltan/

Sandia's Zoltan Library



http://www.cs.sandia.gov/zoltan/

References

- **Kirk Schloegel, George Karypis, and Vipin Kumar. Graph Partitioning for High Performance Scientific Simulations. CRPC Parallel Computing Handbook. Dongarra, Foster, Kennedy, Torczon, White, editors. Morgan Kauffman, 2000.**
- **Zoltan: Parallel Partitioning, Load Balancing and Data-Management Services. http://www.cs.sandia.gov/zoltan**
- **Erik Boman, Cedric Chevalier, Karen Devine. The Zoltan Toolkit – Partitioning, Ordering, and Coloring. http://www.cs.sandia.gov/~kddevin/papers/zoltan_tutorial_dagstuhl09.pdf**
- **Robert Van Engelen. Graph Partitioning for High Performance Scientific Simulations. Slides. Spring 2009. https://www.cs.fsu.edu/~engelen/courses/HPC-adv/GraphPartitioning.pdf**
- **Marsha Berger and Andreas Klöckner. Lecture 12: Load balancing and partitioning. G63.2011.002/G22.2945.001. NYU. November 16, 2010. http://www.cs.nyu.edu/courses/fall10/G22.2945-001/slides/lect12.pdf**
- **E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In Proc. 24th Nat. Conf. ACM, pages 157–172, 1969.**
- **J. A. George and J. W-H. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, 1981**
- **A. Grama et al. Graph Algorithms. Slides accompanying Introduction to Parallel Computing, Second Edition. Addison Wesley, 2003.**