Exercises 9:
Graph Partitioning

Recap of Lecture

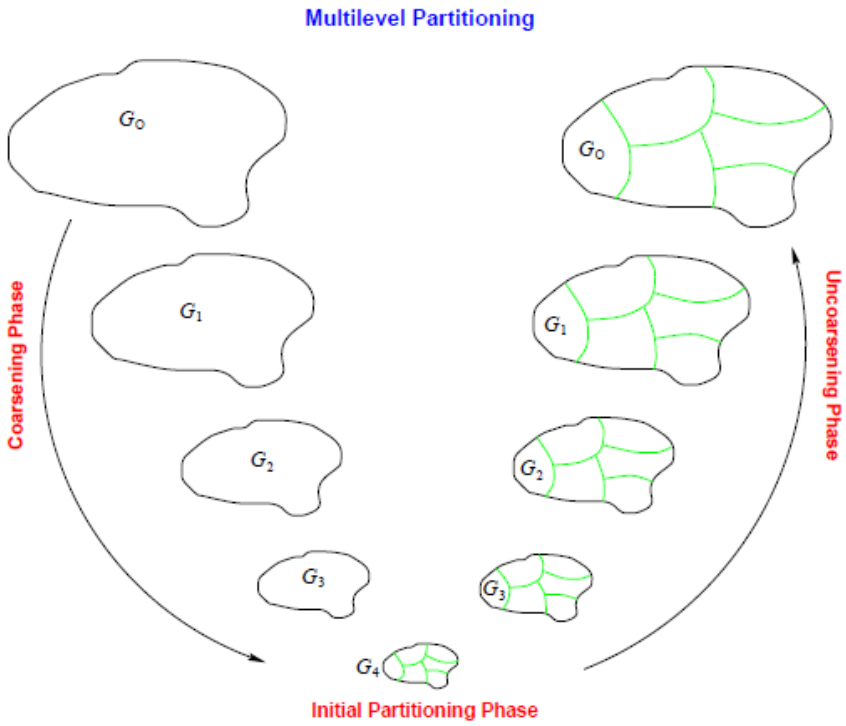
- Graph Partitioning problem
- Overview of heuristics
- Partitioning with Nodal Coordinates
  - Ex: In finite element models, node at point in (x,y) or (x,y,z) space
- Partitioning without Nodal Coordinates
  - Ex: In model of WWW, nodes are web pages
- Multilevel Acceleration
  - BIG IDEA, appears often in scientific computing
- Available Implementations
- Beyond Graph Partitioning: Hypergraphs
- Graph Algorithms for Sparse Direct methods

Software: Metis Overview

Metis

- Both standalone command-line programs and API for C/C++ and Fortran

- Partitions arbitrary graph into k parts using multilevel approaches
  - Multilevel recursive bisection or
  - Multilevel k-way partitioning*



*G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed Computing, 48(1):96–129, 1998.

Partitioning Objectives

- objective of the traditional graph partitioning problem is to compute a k-way partitioning such that the number of edges (or in the case of weighted graphs the sum of their weights) that straddle different partitions is minimized.
  - commonly referred to as the edge-cut.
- Recall: when partitioning is used to distribute a graph or a mesh among the processors of a parallel computer, the objective of minimizing the edge-cut is *only an approximation of the true communication cost* resulting from the partitioning
- communication cost resulting from a k-way partitioning generally depends on the following factors:
  - (i) the total communication volume,
  - (ii) the maximum amount of data that any particular processor needs to send and receive; and
  - (iii) the number of messages a processor needs to send and receive.
- Metis can be used to minimize both (i) and (iii) and (indirectly) (ii)

Load Balancing

- Many important types of multi-phase and multiphysics computations require that multiple quantities be load balanced simultaneously.
- To handle this, Metis allows not only a single weight per vertex, but a vector of m weights per vertex
  - the objective of the partitioning routines is to minimize the edge-cut subject to the constraints that each one of the m weights is equally distributed among the domains
- Ex: first weight corresponds to the amount of computation, second weight corresponds to amount of storage
  - partitioning computed will balance both the computation performed in each domain as well as the amount of memory that it requires

Fill-reducing orderings

- Metis provides the `ndmetis` program and its associated `METIS_NodeND` API routine for computing fill-reducing orderings of sparse matrices based on the multilevel nested dissection paradigm*

*G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing, 20(1):359–392, 1999.

Options Array

- Options are set in metis via an "options array", which is passed as a parameter to most API routines

- **METIS_OPTION_PTYPE:** Specifies the partitioning method.
  - METIS_PTYPE_RB Multilevel recursive bisectioning.
  - METIS_PTYPE_KWAY Multilevel k-way partitioning
- **METIS_OPTION_OBJTYPE** Specifies objective type
  - METIS_OBJTYPE_CUT Edge-cut minimization.
  - METIS_OBJTYPE_VOL Total communication volume minimization
- **METIS_OPTION_CTYPE** Specifies the matching scheme to be used during coarsening
- **METIS_OPTION_IPTYPE** Determines the algorithm used during initial partitioning
  - e.g., METIS_IPTYPE_RANDOM, Computes a bisection at random followed by a refinement

Options Array Continued

• METIS_OPTION_RTYPE	Determines the algorithm used for refinement
• e.g., METIS_RTYPE_FM,	FM-based refinement
• METIS_OPTION_NCUTS	Specifies the number of different partitionings that it will compute. The final partitioning is the one that achieves the best edgecut or communication volume.
• METIS_OPTION_NSEPS	Specifies the number of different separators that it will compute at each level of nested dissection. The final separator that is used is the smallest one.
• METIS_OPTION_NITER	Specifies the number of iterations for the refinement algorithms at each stage of the uncoarsening process.
• METIS_OPTION_UFACTOR	Specifies the maximum allowed load imbalance among the partitions. A value of x indicates that the allowed load imbalance is (1 + x)=1000

Many other graph and hypergraph partitioners

- Zoltan (/Zoltan2)
  - http://www.cs.sandia.gov/Zoltan/Zoltan.html
  - Parallel routines for graph and hypergraph partitioning, graph coloring, reordering
  - Integrated into Trilinos library

Matlab Demos

MeshPart Matlab Routines

- MATLAB MeshPart routines by Gilbert et al.
- Available in ex9.tar on Moodle

- Open MATLAB in directory with files and run

```
meshpart_startup
```

to add needed paths to working directory

Folder structure

- src
  - grid
  - util
  - vis
  - various partitioning methods

- test
  - demos

Basic partitioning routines

- We will experiment with three partitioning routines that we saw in lecture:

- Inertial partitioning
- Geometric partitioning ("random spheres")
- Spectral partitioning

- on a variety of meshes

Task 1: Basic Grid

Type the following commands:

//Generate 40^2 x 40^2 SPD matrix A corresponding to 5-point stencil on a 40x40 grid; xy is coordinates of gridpoints

```
[A,xy] = grid5(40);
gplotg(A,xy);          //plot the graph

[part1,part2] = inertpart(A,xy); //inertial partitioning
gplotpart(A,xy,part1);           //view partition

[part1,part2] = geopart(A,xy);   //geometric partitioning
gplotpart(A,xy,part1);           //view partition

[part1,part2] = specpart(A);     //spectral partitioning
gplotpart(A,xy,part1);           //view partition
```

Task 2: 3D Tetrahedral Mesh

```
% GRID3DT Generate 3-dimensional tetrahedral finite element mesh.
%   [A,xyz] = GRID3DT(k) returns a k^3-by-k^3 symmetric positive definite
%   matrix A of the k-by-k-by-k grid, with cells divided into tetrahedra,
%   and an array xyz of coordinates for the grid points.
```

```
[A, xyz] = grid3dt(10)
```

Which partitioning method does the best?

Is there a big difference? Try them.

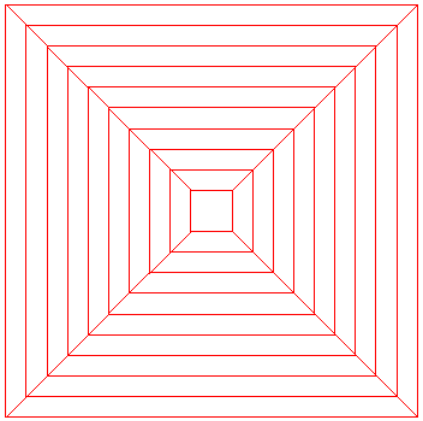
(Hint: rotate the figure to see the 3D mesh)

```
[part1,part2] = inertpart(A,xyz);
gplotpart(A,xyz,part1);
[part1,part2] = geopart(A,xyz);
gplotpart(A,xyz,part1);
[part1,part2] = specpart(A);
gplotpart(A,xyz,part1);
```

Task 3: "Badmesh"

```
[A,xy]=badmesh(100,0);
gplotg(A,xy);
```

- First ask yourself: what is the minimum sized separator?
- Which partitioning method will do the best? worst?



badmesh(10,0);

```
[part1,part2] = inertpart(A,xy);
gplotpart(A,xy,part1);

[part1,part2] = geopart(A,xy);
gplotpart(A,xy,part1);

[part1,part2] = specpart(A);
gplotpart(A,xy,part1);
```

"Badmesh"

```
% BADMESH Mesh that can't be partitioned well with a straight line.
%
%   [A,xy] = badmesh(k,alpha) Generate a k-level mesh (with 4*k points) with
%   no straight-line separator, with alpha<1 the ratio between shell sizes.
%   (alpha defaults to 4/5). alpha=0 means linearly spaced shells.
```

Task 4: Partitioning "Cockroach" graph

```
[A,xy] = cockroach(10);
gplotg(A,xy);
```

- First ask yourself: What is the minimum separator for this graph?
- Which partitioning method will do the best?

```
[part1,part2] = inertpart(A,xy);
gplotpart(A,xy,part1);
[part1,part2] = geopart(A,xy);
gplotpart(A,xy,part1);
[part1,part2] = specpart(A);
gplotpart(A,xy,part1);
```

Guattery and Miller, "On the performance of spectral graph partitioning methods," SODA 1995.

"Cockroach graph"

```
% COCKROACH Planar graph for which spectral partitioning works poorly.
%
%   [A,xy] = cockroach(k) Generate a mesh (with 6*k points) whose best edge
%   separator has size 2, but for which the spectral algorithm gives a
%   separator of size O(k). (From Guattery and Miller, "On the performance
%   of spectral graph partitioning methods," SODA 1995.) Outputs: A is the
%   Laplacian; xy is coordinates for a planar drawing.
```

Task 5: Nested Dissection Demo

Run:

```
meshdemo(4)
```

Read the text and follow along through the steps of the demo

Task 6: Nested Dissection for random sparse matrix

Read and understand what the following commands are doing. Then try them. (Type, e.g., "help specnd" to learn about the specnd function).

```
A = sprand(100,100,.01)+diag(ones(100,1));
A=A'*A;
figure(); spy(A,3)
figure(); spy(chol(A),3)
nd = specnd(A);
figure(); spy(A(nd,nd),3);
figure(); spy(chol(A(nd,nd)),3);
```

What do you observe? (pay attention to "nz", the number of nonzeros, given in the figures).

Some reading...

Guattery and Miller, "On the performance of spectral graph partitioning methods," SODA 1995.

(canonically bad examples)

Gilbert, John R., Gary L. Miller, and Shang-Hua Teng. "Geometric mesh partitioning: Implementation and experiments." SIAM Journal on Scientific Computing 19.6 (1998): 2091-2110.

(random spheres idea)

Pothen, Alex. "Graph partitioning algorithms with applications to scientific computing." *Parallel Numerical Algorithms*. Springer, Dordrecht, 1997. 323-368.

(great overview/survey paper)