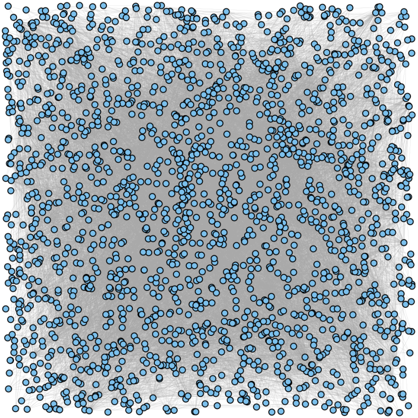
The Spectral Analysis of Graphs for Efficient Algorithms using MPI

TCD HPC MSc

Martin O'Connor

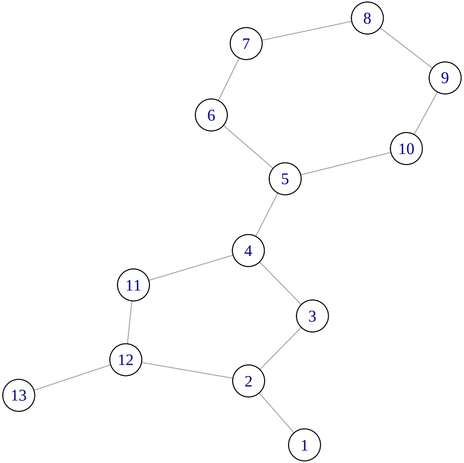
16 September 2004

Example Unstructured Graph



- Graph G(V,E)
- V = 2,032
- E = 13,131
- Unweighted, undirected, connected

13 Vertex Graph



13 Vertex Graph Adjacency Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13
1		1											
2	1		1									1	
3		1		1									
4			1		1						1		
5				1		1				1			
6					1		1						
7						1		1					
8							1		1				
9								1		1			
10					1				1				
11				1								1	
12		1									1		1
13												1	

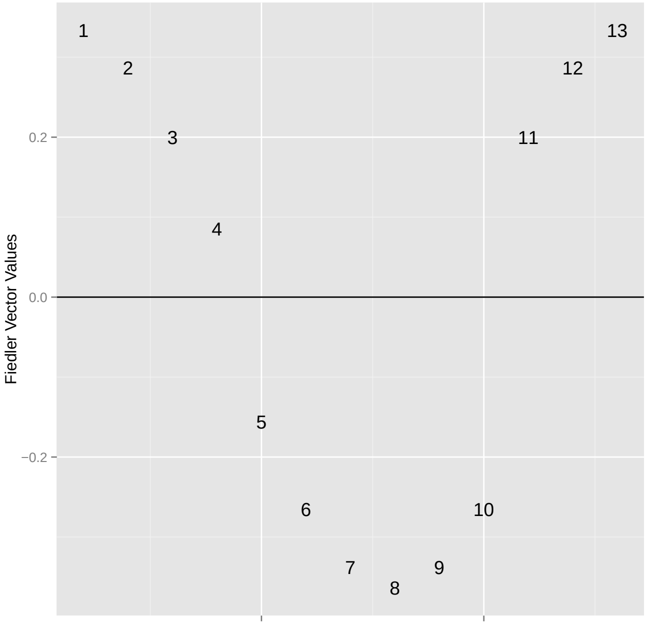
13 Vertex Graph Degree Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1												
2		3											
3			2										
4				3									
5					3								
6						2							
7							2						
8								2					
9									2				
10										2			
11											2		
12												3	
13													1

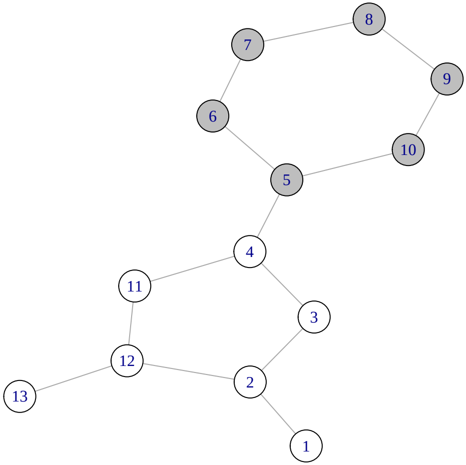
Laplacian Matrix = Degree Matrix – Adjacency Matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	-1											
2	-1	3	-1									-1	
3		-1	2	-1									
4			-1	3	-1						-1		
5				-1	3	-1				-1			
6					-1	2	-1						
7						-1	2	-1					
8							-1	2	-1				
9								-1	2	-1			
10					-1				-1	2			
11				-1							2	-1	
12		-1									-1	3	-1
13												-1	1

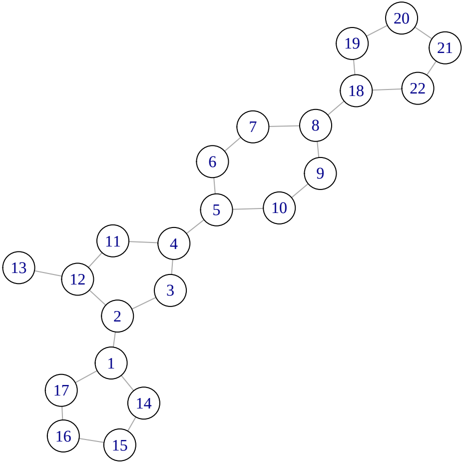
Values of the Fiedler Vector of 13 Vertex Graph



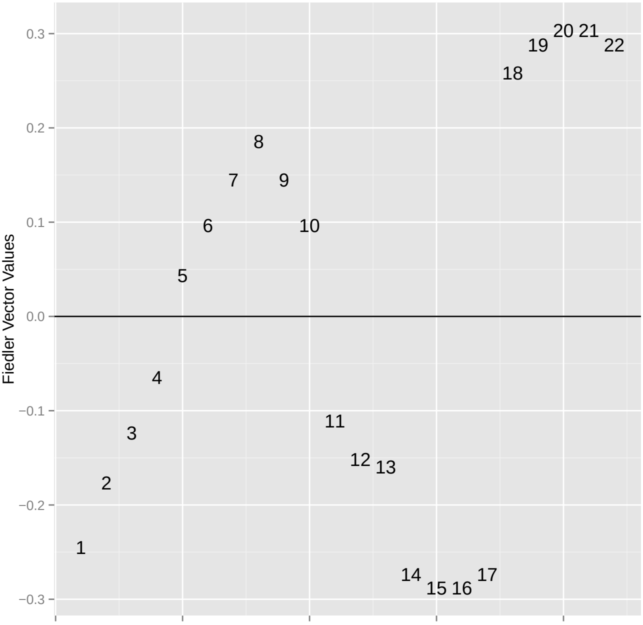
Shading Corresponds to Sign of Fiedler Vector Values



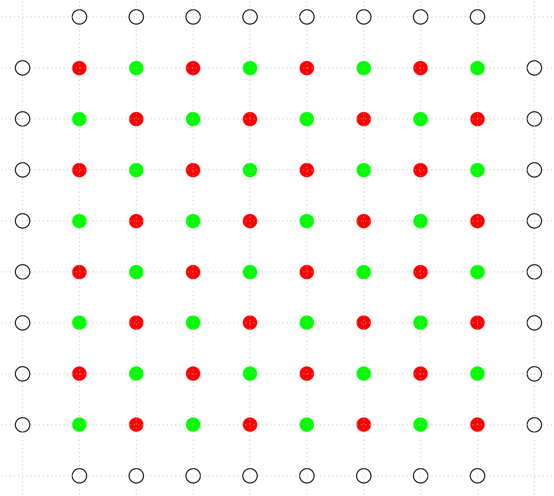
22 Vertex Graph



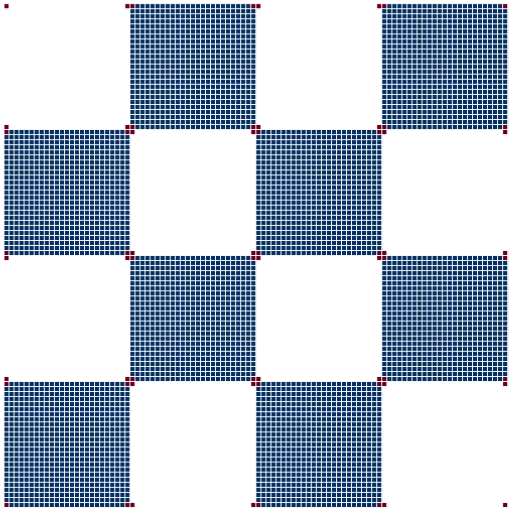
Fielder Vector Values of 22 Vertex Graph



Representing a 2-d Plane as a Graph for an algorithm such as the Ising Model



Checkerboard 2-d Plane as a Graph, Up-Down and Left-Right communicate. Graph is Connected by the extra Corner Cells and PBC



Obtaining an Eigenvector of a Large Matrix

Matrix is Sparse, Symmetric, Real.

Idea: First find a *similar* matrix that is tridiagonal. ($T = Q^{-1}AQ$)
Tridiagonal matrix is much faster to deal with and a 'bag of tricks' can be used to find the Fiedler Eigenvalue and Vector.

1. Apply Lanczos Algorithm to A, constructing the projection of A onto a Krylov subspace with basis matrix Q which we construct (and keep). We obtain the tridiagonal matrix T with $T = Q^T AQ$ so A and T share the same eigenvalues.
2. Use Gerschgorin's Theorem to find global bounds for the eigenvalues of T.
3. Use sign changes on Sturm Sequences of T to find brackets around the second eigenvalue.

Obtaining an Eigenvector of a Large Matrix

5. Apply the Bisection Method to search the Characteristic Equation (last term in the Sturm Sequence), within the brackets from Step 4, the eigenvalue λ_2 of T is found.
6. With the eigenvalue λ_2, use the Inverse Power Method to find the corresponding eigenvector $x_{2(T)}$. The Inverse Power Method needs to solve a system of equations at each step, LU decomposition for tridiagonal matrices was used.
7. Use the saved Q (from the Lanczos method) to calculate the eigenvector $x_{2(A)}$ of A from the eigenvector of T. This is calculated by $x_{2(A)} = Qx_{2(T)}$.

Comparison of External Edge Cuts by Random Assignment and by Spectral Partitioning

		Random Assignment		Spectral Partitioning	
	Edges	External Edges	External Edges (%)	External Edges	External Edges (%)
Regular Lattice w PBC	5,000	4,731	95%	459	9%
Checkerboard Lattice w PBC	9664	8,459	88%	354	4%
Random Graph	13,131	12,330	94%	320	2%

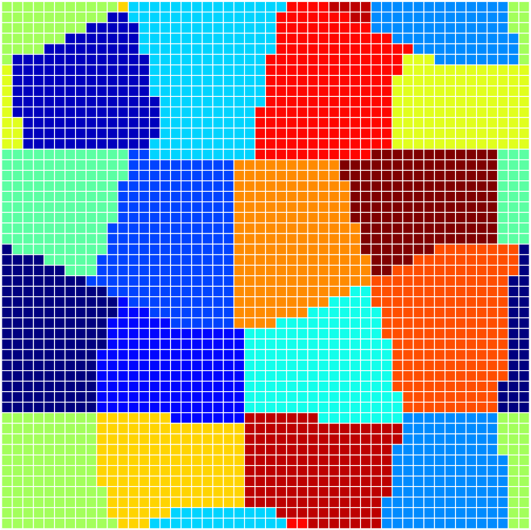
'As is' Partitioning

- Just partition the vertices as they sit in the adjacency matrix, first n go to first partition and so on
- One usual way to generate a lattice is line by line, so this partitioning would generally put lines of vertices together
- A graph probably has more connected things sitting together, a road network recorded town by town etc.
- Would generally not be random, but probably not optimal

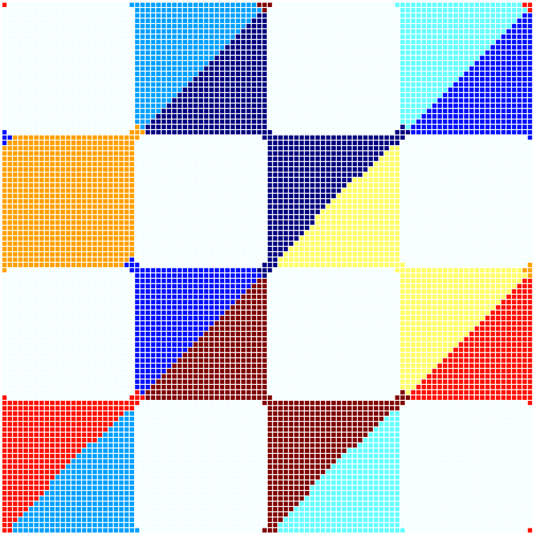
Comparison of External Edge Cuts by ‘As is’ and Spectral Partitioning

	‘As is’			Spectral Partitioning	
	Edges	External Edges	External Edges (%)	External Edges	External Edges (%)
Regular Lattice w PBC	5,000	830	17%	459	9%
Checkerboard Lattice w PBC	9664	232	2%	354	4%
Random Graph	13131	3831	29%	320	2%

Spectral Partitioning (16) of Regular Lattice with PBC



Spectral Partitioning (8) of Checkerboard Lattice with PBC



Partitioning of Random Graph

How the random graph was created:

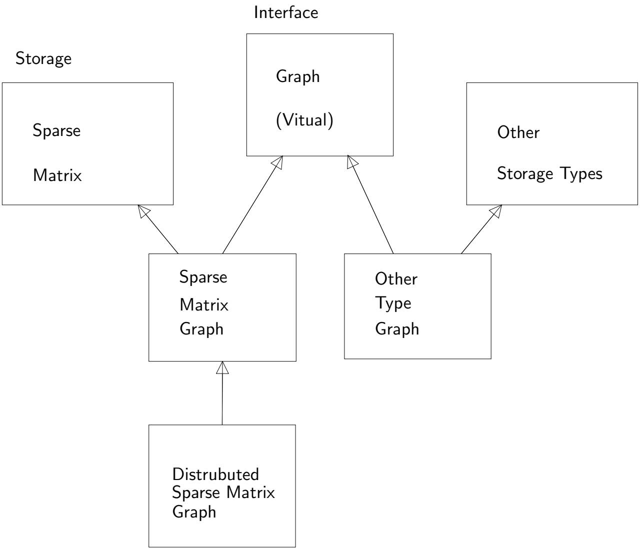
- Graph had 16 centres of size between 200 and 300
- Internal Degree was between 4 and 10
- Each centre had between 6 and 10 edges to other centres

The Spectral Partitioning put 90% of the vertices in groups which consisted mainly of their original centres

Test Algorithm: The Ising Model

- The algorithm is local, each vertex is updated based on its neighbours' state. **communication across edges**
- The algorithm is global due to the connectedness of the graph
- All vertices are updated once in each iteration. Colour Iterations. Many iterations.
- MPI was used in two algorithms
  - Checking Bivariate-ness and Colouring
  - Ising Model Markov Chain
- The MPI function `MPI_Alltoallv` is the main communication function
- Communicate state of vertices with external edges: All vertices or those of a single colour

Representing a Graph in C++



General Programming Approach

- Partitioning is done separate to the model (in serial)
- Ising Model works on a Graph, Distributed Ising Model works on a Distributed Graph
- Distributed Ising Model reads in a partition, or else distributes on an 'as is' basis
- Distributed versions of Graph and Ising Model Classes inherit from serial versions, using as much as possible the inherited data-structures and functions. Distributed::isBiparitite function relies on Serial::isBipartite function when applied to a Distributed::Graph by operating on component found by Serial::connectedComponents function of Serial::Graph
- Output log-files Class and RNG wrapper Class work in serial or parallel

Ising Model on Regular Lattice with PBC

CPUs	Samples	Partitioning	Vertices	Edges	External Edges	Total Time	Send/ Recv	Pack/ Unpack
16	5000	As is	2500	5000	17%	7358	7265	2.53
16	5000	Rand			95%	7258	7163	7.12
16	5000	Spectral (16)			9%	7360	7280	2.91

Ising Model on Checker Lattice with PBC

CPUs	Samples	Partitioning	Vertices	Edges	External Edges	Total Time	Send/ Recv	Pack/ Unpack
8	5000	As is	5032	9664	2%	441	91	3.54
8	5000	Rand			88%	506	138	17.04
8	5000	Spectral (8)			4%	461	114	2.40

Ising Model on Random Graph

CPUs	Samples	Partitioning	Vertices	Edges	External Edges	Total Time	Send/ Recv	Pack/ Unpack
16	5000	As is	2032	13131	29%	7229	7148	2.94
16	5000	Rand			94%	7189	7095	9.30
16	5000	Spectral (8)			2%	7270	7173	2.49

One Possible Reason why 'More Communication = Faster'

- Ising model runs in lock-step
- Running `MPI_Alltoallv` 'as is' can have an advantage because it tends to have less *external ranks*, if not less *external edges*
- For example, with regular lattice graph, typically 'as is' has 2 *external ranks* against 5 for Spectral partitioning
- A simulation of a 16 core MPI run passing 64, 48 or 32 ints to 2, 8 or 15 other ranks, iterating 250,000 times with a random sleep $\leq 50\mu$ per rank per iteration.

Ranks	Data size	Time
2	64	372.81
8	48	381.8
16	32	393.82

Conclusions

- Spectral Partitioning works quite well in reducing external edges partitioning a graph, at least versus a random partition
- Partitioning in itself did not greatly improve performance in the case of my particular tests
  - In tests where communication time was dominant there was little difference between 94% and 2% external edges
  - In tests where compute time dominated also partitioning brought no major benefit
  - There is some evidence that having less *external ranks* as opposed to less *external edges*, makes some difference