A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs

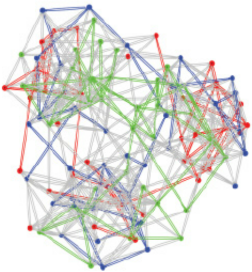
George Karypis and Vipin Kumar

Jared Di Carlo

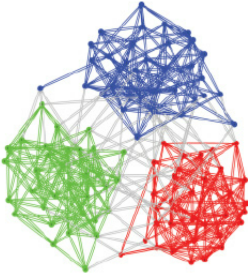
April 23, 2019

Graph Partitioning

- Divide vertices into p parts of roughly equal size (or sum of vertex weights)
- Minimize edges across parts
- NP-Complete



(a) A poor edge-cut partitioning. Vertices are assigned to partitions at random, thus, there are many inter-partition links.



(b) A good edge-cut of the same graph, where vertices that are highly connected are assigned to the same partition.

Applications of k-way Graph Partitioning

- Scheduling work on k processors
  - Edges represent sharing of data between tasks
  - Sparse matrix vector product
- Sparse matrix factorization
  - Reorder matrix to make the factorization sparse too
- Power Law Graphs?

Algorithms for Graph Partitioning

- Spectral Partitioning: Slow

Algorithms for Graph Partitioning

- Spectral Partitioning: Slow
- Geometric Partitioning : Requires vertices to have coordinates

Algorithms for Graph Partitioning

- Spectral Partitioning: Slow
- Geometric Partitioning : Requires vertices to have coordinates
- Multilevel: Fast, but low quality

Multilevel Partitioning

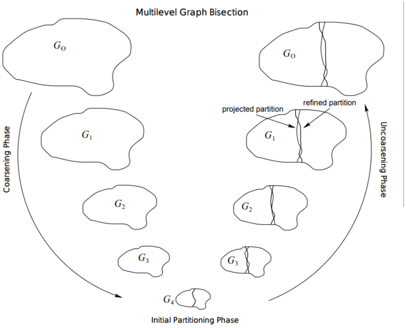
- Three phases
  - Coarsening (collapse vertices)

Multilevel Partitioning

- Three phases
  - Coarsening (collapse vertices)
  - Partition

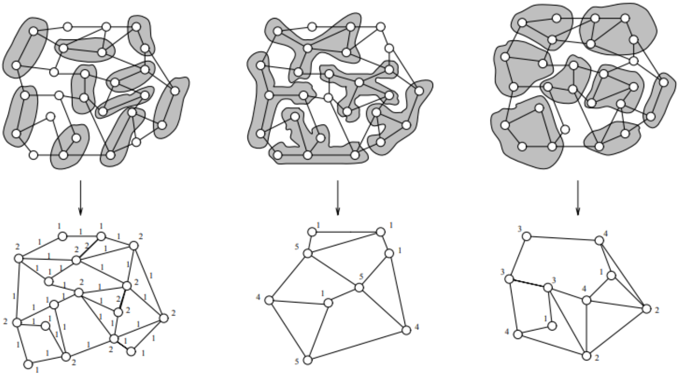
Multilevel Partitioning

- Three phases
  - Coarsening (collapse vertices)
  - Partition
  - Uncoarsening (possibly refining the partitions)



Coarsening

- Each coarsening iteration collapses a *maximal matching*
  - Matching: set of edges which hit each vertex at most once
  - Coarsen: Collapse edges in matching
  - Maximal Matching: Every non-matched edge touches a vertex which has a matched edge



Coarsening Strategies

Random

- Visit random vertices, add random edges. $O(|E|)$
- Terminate when there are no more vertices that can have edges added
- Works well on "engineering" graphs - meshes

Coarsening Strategies

Heavy Edges

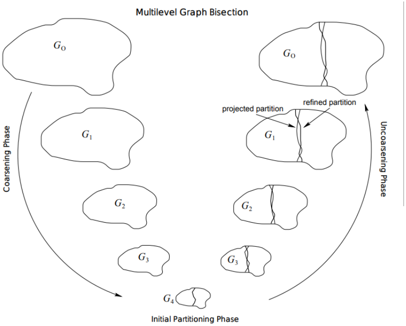
- Greedy
- Visit random vertices, pick heaviest edge to remove: $O(|E|)$
- Can reduce the edge-cut because heavy edges are removed
- Does well for VLSI graphs

Heavy Clique

- Collapse nodes which are unlikely to be split by the bisection
- Randomly visit vertices, pick edges leading to highest edge-density vertex
- Edge Density: $2\frac{E_U}{U(U-1)} = 2\frac{CE(u)+CE(v)+EW(u,v)}{(VW(u)+VW(v))(VW(u)+VW(v)-1)}$

Partitioning Phase

- This step is fast - coarse graph should have around 100 vertices
- Spectral Bisection, KL, GGP



Spectral Bisection

- Consider $Q = D - A$ where D is diagonal degree matrix, A is adjacency matrix.
- Eigenvectors: $Qx = \lambda_i x$
- Let x represent a partition with $x_i \in \{-1, 1\}$
- The product Qx is proportional to the number of cut edges

KL Algorithm

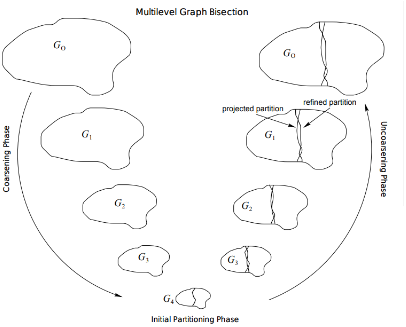
- Iterative - greedily swaps vertices to make things better
- Can get stuck in local minima
- Run the algorithm a few times (5 - 10)
- Can be improved by prioritizing vertices with a large effect

Graph Growing Partition (GGP)

- BFS from a random vertex until half the vertices are added
- Sensitive to initial choice

Uncoarsening

- As vertices are expanded, move ones on the edge to improve edge-cut
- KL, KL(1), KL Boundary



KL, KL(1)

- Same algorithm as KL previously
- Terminates very quickly, as the partition is already good
- Dominated by insertion into data structure
- KL(1) runs a single iteration, allowing simpler data structures
- Boundary KL - only consider vertices that are on the edge
- Use BKL(1) on large graphs, BKL on smaller graphs

Experiments - Graph Partition

- SGI Challenge, 200 MHz MIPS R4400, 1.2 GB RAM
- Vertices: 4960 to 448695
- Edges: 9462 to 3358036
- 2D/3D meshes, stiffness matrix, “Chemical Engineering”, Highway, Stiffness matrix, Circuits (adder, memory, sequential)
- Coarsening: Heavy Edge has the lowest edge cut, and good runtime
- Partitioning: GGP or Spectral, depending on graph
- Uncoarsening: BKL or BKL dynamic - dynamic is faster, but BKL is 2% better

Experiments - Sparse Matrix Factorization

TABLE 6

The number of operations required to factor various matrices when ordered with multiple minimum degree (MMD), spectral nested dissection (SND), and our multilevel nested dissection (MLND).

Matrix	MMD	SND	MLND
144	2.4417e+11	7.6580e+10	**6.4756e+10**
4ELT	1.8720e+07	2.6381e+07	**1.6089e+07**
598A	6.4065e+10	2.5067e+10	**2.2659e+10**
AUTO	2.8393e+12	7.8352e+11	**6.0211e+11**
BCSSTK30	**9.1665e+08**	1.8659e+09	1.3822e+09
BCSSTK31	2.5785e+09	2.6090e+09	**1.8021e+09**
BCSSTK32	**1.1673e+09**	3.9429e+09	1.9685e+09
BRACK2	3.3423e+09	3.1463e+09	**2.4973e+09**
CANT	4.1719e+10	2.9719e+10	**2.2032e+10**
COPTER2	1.2004e+10	8.6755e+09	**7.0724e+09**
CYLINDER93	6.3504e+09	5.4035e+09	**5.1318e+09**
FINAN512	5.9340e+09	1.1329e+09	**1.7301e+08**
FLAP	1.4246e+09	9.8081e+08	**8.0528e+08**
INPRO1	**1.2653e+09**	2.1875e+09	1.7999e+09
M14B	2.0437e+11	9.3665e+10	**7.6535e+10**
PWT	1.3819e+08	1.3919e+08	**1.3633e+08**
ROTOR	3.1091e+10	1.8711e+10	**1.1311e+10**
SHELL93	1.5844e+10	1.3844e+10	**8.0177e+09**
TORSO	7.4538e+11	3.1842e+11	**1.8538e+11**
TROLL	1.6844e+11	1.2844e+11	**8.6914e+10**
WAVE	4.2290e+11	1.5351e+11	**1.2602e+11**

- Parallelization - better than MMD (greedy)
- 56x speedup on 128-CPU Cray T3D

Comparison

TABLE 9
Characteristics of various graph partitioning algorithms.

	Number of Trials	Needs Coordinates	Quality	Local View	Global View	Run Time	Degree of Parallelism
Spectral Bisection	1	no	●●●●	○	●●●●	■■■■	▲▲
Multilevel Spectral Bisection	1	no	●●●●	○	●●●●	■■■	▲▲
Mulitlevel Spectral Bisection-KL	1	no	●●●●●●	●●	●●●●	■■■	▲▲
Multilevel Partitioning	1	no	●●●●●●	●●	●●●●	■■	▲▲
Levelized Nested Dissection	1	no	●●	○	●●	■■	▲▲
	1	no	●●	●●	○	■■	▲
Kernighan-Lin	10	no	●●●◐	●●	●◐	■■■	▲▲
	50	no	●●●●	●●	●●	■■■◧	▲▲