Evolving Multi-level Graph Partitioning Algorithms

Aaron Pope [1 2] Daniel Tauritz [1] Alexander Kent [2]

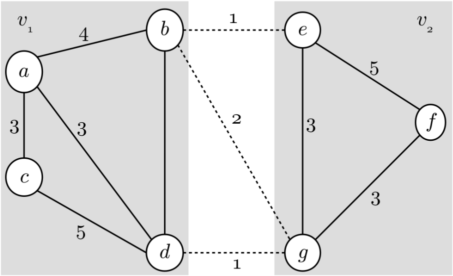
[1]Department of Computer Science,
Missouri University of Science and Technology,
Rolla, Missouri 65409

[2]Los Alamos National Laboratory
Los Alamos, New Mexico 87545

December 8, 2016

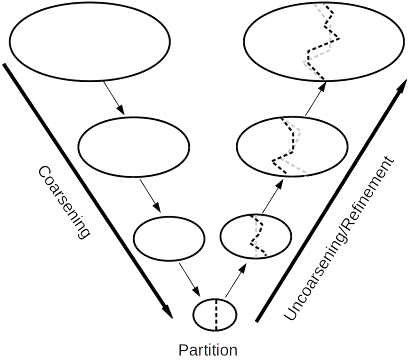
Graph Partitioning

- Graph partitioning can be applied to several problems
  - Parallel computation: minimize inter-process communication
  - Network security: placing network traffic monitors at critical junctures
- In general, optimal graph partitioning is NP-hard
- Real-time applications require fast approximation heuristics

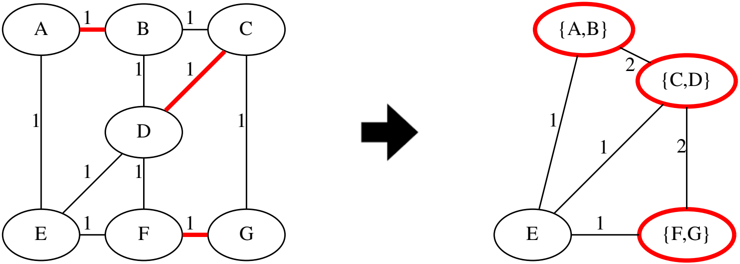


Multi-level Graph Partitioning

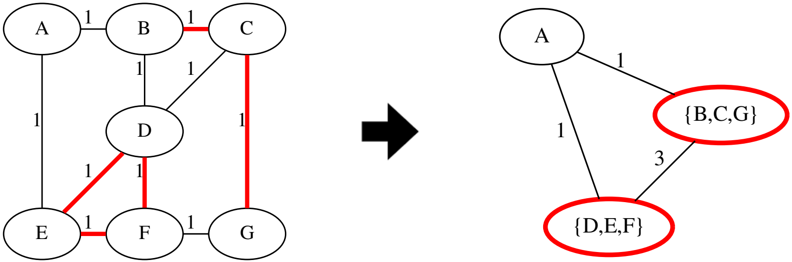
1. Approximate graph with sequence of smaller graphs
2. Quickly partition smallest graph
3. Use partition of smallest graph as an initial partition of the next smallest
4. Perform greedy search to improve the quality of the partition
5. Repeat steps 3 and 4 until partition is found for the original graph



Coarsening: Matching Contraction



Coarsening: Subgraph Contraction



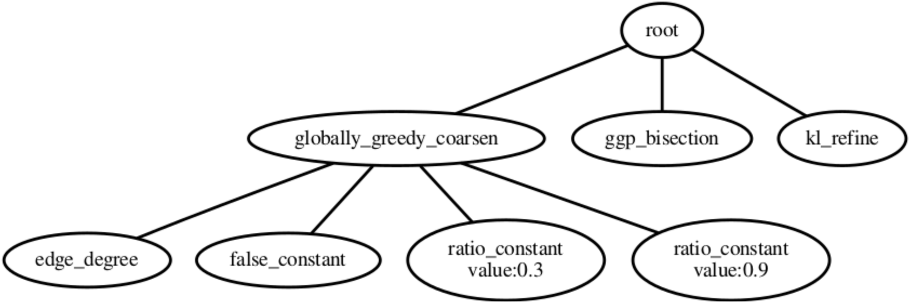
Optimizing Multi-level Partitioning

- Multi-level partitioning of power-law graphs can be improved with specialized coarsening heuristics*
- Demonstrates the potential to customize an algorithm for graphs with particular characteristics
- Manually developing new heuristics can be expensive
- Process can be automated using hyper-heuristics

*A. Abou-Rjeili and G. Karypis, "Multilevel Algorithms for Partitioning Power-law Graphs," in Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS'06), IEEE, Rhodes Island, Greece, 2006.

Hyper-heuristic Approach

- Extract functionality from existing graph partitioning techniques
- Use Genetic Programming (GP) to construct new algorithms



Methodology

- Evolve a population of graph partition algorithms
- Strongly typed parse tree representation
- Evolution targeted at specific type of graph (e.g., random graph model, computer network)
- Cost of partitions produced determines solution fitness (averaged over many graphs)

$$Fitness = \frac{1}{|P|} \sum_{p \in P} \left[\sum_{(u,v)|\ p[u] \neq p[v]} w_e \left[(u,v) \right] \right]$$

Primitive Operations

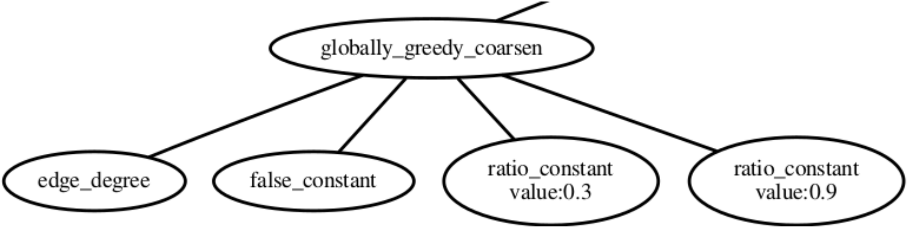
Root node

- child node for each of: coarsening, partitioning, uncoarsening

Coarsening

- Traditional matching schemes: random, heavy edge, light edge
- Globally greedy: considers all edges in graph sorted by some metric
- Locally greedy: randomly visits vertices, considers incident edges
- Edge metrics: weight, degree, or core number of incident vertices, or edge weight (can be combined using math operators)

Globally and Locally Greedy Coarsening



- Edge metric
- Reverse ordering (maximize)
- Maximum vertex weight ratio
- Maximum contraction ratio

Primitive Operations (cont.)

Partitioning

- random, growth, greedy growth, spectral, Kernighan-Lin

Uncoarsening

- KL refinement, greedy refinement

Miscellaneous

- Constants: integers, ratios, booleans
- Random conditional: randomly true or false according to child probability value

Experiment

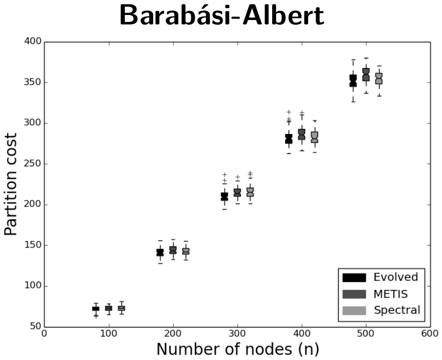
Partition algorithms are evolved targeting:

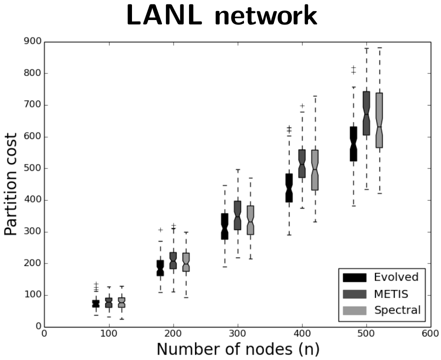
- Erdös-Rényi random graphs
- Barabási-Albert random graphs
- Los Alamos National Laboratory (LANL) authentication graphs

Performance is compared against METIS partitioning software as well as spectral partitioning

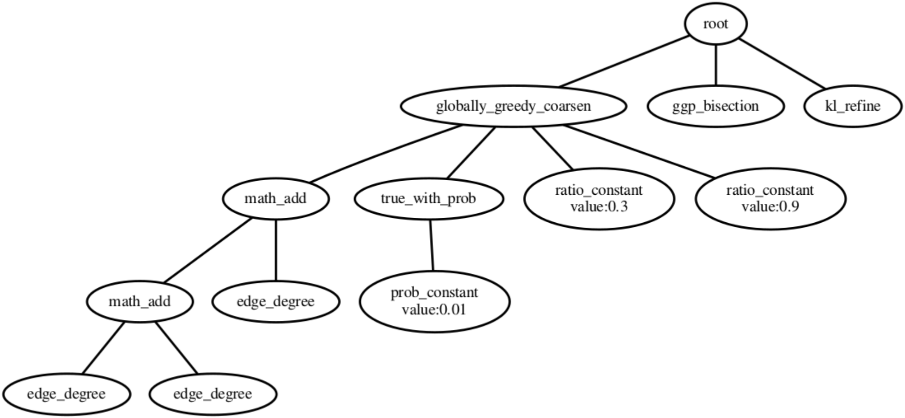
Results

Method	E_{ER}	E_{BA}	E_{LANL}	METIS	SP
E_{ER}	0.0	−0.06	−0.38	−0.06	−4.55
E_{BA}	−0.53	0.0	−0.12	−0.72	−0.92
E_{LANL}	−0.48	−0.10	0.0	−2.97	−3.90





Representative High Fitness Final Evolved Solution ($\mathbf{E_{BA}}$)



Conclusion

- Multi-level partitioning algorithms can be tailored to specific graph types, including real-world networks
- These customized heuristics improve upon general purpose graph partitioning methods
- The design process can be automated using hyper-heuristics

Future Work

- Evaluate execution time as an additional objective
- Alternatives to balanced bisection
- Community detection applications
- Relax the *coarsening* $\rightarrow$ *partition* $\rightarrow$ *uncoarsening* format restriction to allow for more flexibility in algorithm structure

Questions?