1 Conclusions and Future Work

In conclusion I have utilized maximum locally-connected subgraph partitioning from Chung-Lu to partition a graph into a planar subgraph and a teleportation subgraph. I then combine optimal multigrid solves on the planar subgraph with linear algebra and a direct solve on the teleportation subgraph. I have now solved the original laplacian linear system. I give a complexity model for the partitioning algorithm, $O(num\_iter. \times num\_edges \times E[d]^3)$, and a suboptimal complexity for the linear system solve, $O(num\_nodes^3)$. Using the Woodbury matrix identity [**?**] I solve the laplacian linear systems for biological networks of *C. Elegans*, for Facebook friend circles, and show how my method does not work for the power grid of the Western US. The individual operations align with their theoretical computational cost. However, there is much to continue working on.

1.1 Graph Partitioning

My graph partitioning algorithm is written in python mainly because of ease-of-use and because of the NetworkX library that provides simple graph computations. However I can rewrite much of the code in C to efficiently create graph structs. I can then test local-connectivity for edges in parallel, drastically decreasing time to partition. I would also like to submit my graph partitioning algorithm to the open source NetworkX library as an alternative to the k,l_connected_subgraph function for small k [**?**].

1.1.1 Theoretical Bounds

In addition to working on the code for the algorithm, I would like to work with graph theorists to really classify graphs with large locally-connected components. There might be bounds on the size of the teleportation subgraph which can lead to bounds on the rank of the teleportation laplacian matrix. This could help establish a class of graphs for which my method is appropriate.

1.2 Linear System Solve

Most of my code is built on PETSc which is written in C and optimized for sparse matrices [**?**] however the initial singular value decomposition is computed using numpy and does not utilize the low-rank (r) nature of the teleportation laplacian matrix. This results in $O(n^3)$ operations instead of $O(r^2 n)$ operations. Also, PETSc does not have a vectorized multiple right-hand-side solve technique that will drastically decrease computations for the $Q_1 = VQ$ step in the linear system solve that is causing a bottleneck. If I implement this, I can speed up the system solve.

1.3 Larger Graphs and Finding Meaningful Solutions

I only tested my method on graphs of small to medium size due to memory constraints on my laptop. I would like to test these and much larger graphs on a

cluster to determine how my method scales. Because I am especially interested in biological systems, I would like to test my method on more complex systems than *C. Elegans*.

Finally, and most importantly, I did not have enough time to truly evaluate solutions to the laplacian linear systems for my example graphs. I need to do more work to find appropriate right hand sides to solve against. Currently I am solving against a random right hand side. What are the most important regions of the worm's nervous system? Are there any hidden protein-phenotype expressions that could be important in unraveling the worm's genome? Who are the origins of influence in the Facebook friend networks? These are the real questions scientists are asking, I have proposed a method to answer them, and would like to see the final results.