Phase 1 Project Selection Status Report

Name: Albert Kutsyy

College: Trinity

User Identifier: ak2149

Director of Studies: Prof. Frank Stajano, Dr. Sean Holden, and Dr. Neel Krishnaswami

1 Introduction and Description of Work

Betweenness centrality is a very important metric for analyzing graphs, and algorithms to efficiently compute it have become very relevant within the past decade. In addition to the variety of algorithms that exactly compute betweenness centrality, a number of algorithms have been developed which attempt to estimate the betweenness centrality of a node through probabilistic methods. Further, recent studies have noted that the performance of individual betweeness centrality algorithms depends a great deal on the characteristics of the network they are run on.

I will attempt to use graph statistics to determine the situations where each of a number of algorithms should be used. Specifically, I will look at the Brandes [1] algorithm, the Brandes++ [2] algorithm, an algorithm based on random walks [4], and an algorithm based on random sampling [5]. I will implement each of these algorithms in Java, adding instrumentation such that I can accurately measure a variety of performance metrics.

I will verify my implementation of each of these algorithms by comparing their performance to the fastest known implementation. This will allow me to verify the correctness, as well as confirm that my performance is within an order of magnitude of the most efficient known implementation. The comparison is important to verify that I'm not missing any major optimization tricks. Since I will be implementing all four algorithms (and using the same language/toolkit) and verifying that the performance is as expected, I'll be able to confirm that their relative performance is actually due to the algorithms rather than the implementation.

Next, I will select several network datasets from the Stanford SNAP [3] dataset collection. I will find a variety of network statistics for each dataset, and then run each of my instrumented algorithms on the dataset. Using these results, I will attempt to determine a set of metrics by which one can predict the most efficient algorithm to run on a given dataset. I will also determine a set of metrics by which one can predict the accuracy of the two approximation algorithms. As a possible extension, I may create tools to modify the datasets to change their properties (such as adding or deleting nodes to change connectivity) in order to measure the effects on the algorithms.

2 Starting Point

Of the four algorithms, Brandes [1] is the best known and de-facto "standard" algorithm. Brandes++ [2] can exhibit significant speedups over Brandes, especially for graphs that have small k-cuts. It doesn't provide much benefit for graphs that have large k-cuts. Kornaropoulos and Riondato claim that their algorithm outperforms Brandes on a number of real-world datasets, but makes no comparison to Brandes++ or other approximation algorithms [5]. Newman makes no such claims about their random walk algorithm [4]. I have not been able to find any other comparisons between these algorithms.

3 Substance and Structure of the Project

I will implement the algorithms and graph statistic calculations in Java. The program will output the time taken to run each algorithm on each dataset as well as performance metrics such as number of reads.

4 Success Criteria

My success criteria will be that I have correctly and reasonably efficiently implemented the four algorithms, run them on at least 5 datasets with varied network statistics, and determined to what extent it is possible to predict the relative performance of the algorithms, as well as the accuracy of the approximate algorithms.

I will evaluate my project by comparing the efficiency and outputs of my implementations with a known implementation. I will also verify that my instrumentation gives repeatable results for each algorithm, and that this correlates strongly to the real amount of time spent.

5 Plan of Work

5.1 Timetable

The following timetable breaks the project into 10 2-week work packages.

1. Verify that all necessary resources are ready. This includes familiarizing myself with running projects on the compute server.

   Install all software and packages needed to run the project.

   Research how to instrument code.

2. Use a pre-existing package to run a betweenness centrality algorithm on several Stanford SNAP datasets to determine size constraints.

   Research relevant graph statistics.

3. Implement and instrument the Brandes algorithm. Test on a small dataset. Compare performance to best known implementation.

4. Implement and instrument the Brandes++ algorithm. Test on a small dataset. Compare performance to best known implementation.

5. Implement and instrument the random samples and random walk algorithms. Test on a small dataset. Compare performance to best known implementations.

6. Run each algorithm on at least 5 datasets.

   Write the progress report

7. Determine the graph statistics of each of these datsets.

   Analyze the results of these experiments.

8. Begin writing dissertation.

9. Continue writing dissertation.

10. Finish writing dissertation.

5.2 Milestones

- By the end of the 6th week of work, I should be able to run Brandes [1] on a small dataset and confirm that performance is comparable to the best known implementation.
- By the end of the 8th week of work, I should be able to run Brandes++ [2] on a small dataset and confirm that performance is comparable to the best known implementation.
- By the end of the 10th week of work, I should be able to run the random samples [5] and random walk [4] algorithms on a small dataset and confirm that performance is comparable to the best known implementations.
- By the end of the 10th week of work, I should have run all of these algorithms on my selected 5 datasets. **This is the first part of my success criteria**
- By the end of the 12th week of work, I should have run all relevant graph statistics on the 5 datasets. **This is the second part of my success criteria**

6 Resource Declaration

- My own laptop
  - I will use this for small experiments and for ease of use.
  - All code will be checked into GitHub daily, and all setup and non-code items will be mirrored onto the MCS. In case of failure, I will use the MCS backup.
  - *I accept full responsibility for this machine and I have made contingency plans to protect myself against hardware and/or software failure.*
- Graph Analysis Package
  - I will use a graph analysis package to do my initial tests with.
  - This is widely available. I will use the `org.jgrapht.alg.scoring` library.
- Stanford SNAP datasets
  - I will use 5 of these datasets to run experiments on.
  - It is freely available and I will download at least 5 datasets as soon as the project is approved.

- Access to a high performance computer.
  - I would benefit from a high-performance computing server to speed up the experiments.
  - My supervisor has already arranged access to a high performance computer through the computer lab.
  - In case that doesn't work, I can use my laptop or the MCS and allocate more time for the experiments.

References

[1] Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[2] Dora Erdos, Vatche Ishakian, Azer Bestavros, and Evimaria Terzi. A divide-and-conquer algorithm for betweenness centrality, 2015.

[3] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[4] M.E. J. Newman. A measure of betweenness centrality based on random walks. *Social Networks*, 27(1):39 – 54, 2005.

[5] Matteo Riondato and Evgenios Kornaropoulos. Fast approximation of betweenness centrality through sampling. volume 30, pages 413–422, 02 2014.