1 Introduction

My name is Patrick J. McSweeney and I am a fourth year PhD candidate in computer science at Syracuse University. This document outlines a proposal for a Google Summer of Code (GSOC) 2009 project to implement several network statistics into the open source application Gephi. The rest of this article is organized as follows: section 2 outlines my extended proposal, section 3 discusses a proposed timeline and development issues section 4 discusses lessons learned from participating in last year’s GSOC and section 5 discusses my interests and strengths for this project. I am very interested in this project as I am coming up on my doctoral defense (hopefully by May 2010) and I wish to learn about the many different network metrics that I have not been exposed to.

2 Proposal Details

For my proposal I have chosen to implement idea # 2 Network Algorithm and Statistics. My introductory understanding of Gephi, is that its primary feature is to provide useful and efficient network visualization and exploration techniques. Some network features may not be readily visible through standard visual perception and thus the network statistics for Gephi will act to extend the users perception into the mathematical domains of the networks. In addition to the algorithms listed I have suggested three other network metrics, which are marked with stars.

Network Metrics:

1. **Watt-Strogatz Clustering Coefficient**: This metric measures how close the neighborhood of a specific node $u$ is to a complete subgraph, where the neighborhood of a node is defined as the set of nodes that are immediately adjacent to node $u$. The clustering coefficient for a node $u$ is then:

\[
(\text{undirected}) \quad C(u) = \frac{2 \cdot |\{(v, w) \in E | (u, v) \in E \wedge (u, w) \in E\}|}{k_u \cdot (k_u - 1)},
\]

where $k_u$ is the degree of node $u$. This formulation accounts for the number of edges in the neighborhood of $u$, normalized by the maximum possible number of edges in a complete subgraph of size $k_u$. The clustering coefficient ranges from 0 to 1, with 1 indicating a complete subgraph and 0 indicating no clustering.

Gephi Network Statistics
Google Summer of Code 2009 Project Proposal

Patrick J. McSweeney
pjmcswee@syr.edu
(directed) \( C(u) = \frac{|\{(v, w) \in E | (u, v) \in E \land (u, w) \in E\}|}{k_u \cdot (k_u - 1)} \),

where \( E \) is the set of edges, and \( k_u \) is the total degree (in and out edges) of node \( u \). We can refer to the clustering coefficient of a graph \( G \) as the average clustering coefficient of all nodes in the network:

\[
C(G) = \frac{1}{N} \sum_{i=0}^{N} C(i)
\]

2. **Pagerank**: An iterative algorithm that measures the importance of each node within the network. The metric assigns each node a probability that is the probability of being at that page after many clicks. The page rank values are the values in the eigenvector that has the highest corresponding eigenvalue of a normalized adjacency matrix \( A' \). The standard adjacency matrix is normalized so that the columns of the matrix sum to 1. I know the least about this algorithm, so I will need to do some research into efficient ways of computing these values.

3. **HITS**: Actually computes two different scores: hubs and authority. The authority score indicates the value of the page(node) itself and hubs estimates the value of the links outgoing from the page (node). Hits is an iterative algorithm at each iteration:

(a) Update the authority value of each node to be the sum of the hub values for every node it has a link into.

(b) Update the hub values for each node to be the sum of the authority values that it has a link into.

(c) Normalize the hub and authority scores for all nodes by normalizing each value by the system sum for each value.

(d) Repeat these steps (assumingly until the values no longer fluctuate).

4. **Graph Distance Metrics**

The following four metrics I propose to do at the same time (within the same function) as doing them together can save time. I will most likely use a modified FloydWarshall algorithm to compute these values.

(a) **Average Shortest Path**: The graph distance, \( \gamma \), between two nodes \( u \) and \( v \) is defined as the minimum number of edge-hops required to traverse the network starting from node \( u \) and end at node \( v \) (in an undirected network these
two values are identical, but not so in a directed network). The average shortest path between all nodes in a network is seen as a measure of the small world effect.

(b) **Network Diameter:** The largest $\gamma$ to be found in a network. This metric will be computed at the same time as the average shortest path.

(c) **Node Betweenness Centrality:** This metric indicates how often a node is found on a shortest path between two nodes in the network.

$$CB(v) = \sum_{u,w \in N, u \neq v \neq w} \frac{\sigma_{u,w}(v)}{\sigma_{u,w}},$$

where $\sigma_{u,w}(v)$ is the number of shortest paths from $s$ to $t$ that pass through node $v$ and $\sigma_{u,w}$ is the total number of shortest paths from node $s$ to node $t$.

(d) **Node Closeness Centrality**: This metric indicates how long it will take for information from a node $u$ will take to reach other nodes in the network.

$$CC(v) = \sum_{u \in N, u \neq v} \frac{\gamma(u,v)}{N},$$

which is the average shortest path from node $u$ to all other nodes $v$. In the case where not all nodes are reachable from node $u$, we take the average over only non-infinite paths.

5. **Modularity**: Real world networks have been shown to separate into logical clusters in which nodes are tightly connected to each other but only loosely connected to nodes outside of their module. Newman’s modularity is currently the most widely used metric to measure how modular a network is. Given a partition $P$ and a network $G$ modularity is defined as:

$$Q(P, G = < N, E >) = \sum_{C_i \in P} \frac{l(C_i)}{|E|} - \left(\frac{d(C_i)}{2 \cdot |E|}\right)^2,$$

where $C_i$ is a module within the partition, $l(C_i)$ is the number of edges that connect nodes within $C_i$ and $d(C_i)$ is the degree of $C_i$. The subtractive term is the square of the percentage of edge-endpoints that occur within $C_i$. Thus the metric as a whole computes the variation between the fraction of internal edges discovered within $C_i$ and the at-random expected frequency of internal edges. There are several existing algorithms that have been developed to compute these values, I am considering using
extremal optimization. If time allows I may implement one of my own algorithms (publication pending).

6. **Degree-Distribution (Power-law)**: It has been shown that many networks are “scale-free”, in the sense that their node degree distributions follow a power-law that is not affected by the size of the network. This power law states that the frequency $F$ of nodes that have degree $k$ decreases according to:

$$F(k) \approx k^{-a},$$

where $a$ is a real typically between 2 and 3.

### 3 Time Line & Development Issues

According to Google's website we have 4 weeks (April 20th - May 23rd) of prep-time and 13 weeks (May 23rd - August 17) to implement this project. I have broken down this time into five phases, described below.

1. **Phase 1: “Learning Curve” April 20th - May 23rd** During this time I will be learning the source code for Gephi, and start laying down an outline for the (GUI). Also during this time I will spend some time examining some of the suggested metrics that I have not spent any time on yet, specifically HITS and PageRank. While this little code is actually written, I do feel in many ways this is the most important phase for the project. Learning the Gephi source code is an essential step for successfully completing this project on time. I should point out that for the first 2 weeks of this phase I will be involved with responsibilities from the regular semester, however, once these are wrapped up the project will have more of my attention.

   **Tasks:**

   - Begin examining the code base for Gephi.
   - Thoroughly research of algorithms for HITS and PageRank.
   - Outline potential GUI layouts for consideration.

2. **Phase 2: “Planning” May 23rd - June 8th** This 2 week phase will have me outlining the algorithms at a high-level. I view this phase as necessary in order to produce quality code. Planning should reduce the complexity of the code along with avoiding problems later on.

   **Tasks:**

   - Construct class Diagrams.
• Construct software architecture.
• High level program structure design.
• Get general approval for GUI interface.

3. Phase 3: “Algorithm Implementations” June 8th - July 13th Phase 3 will consist of 6 weeks in which I will implement the algorithms discussed in section 2 based on the outlines and preliminary models created in Phase 2. This phase is where the majority of the coding work shall be done. Here I will be doing the actual Java coding and debugging for the project.

Tasks:
• Implement code outlined from Phase III
• Fill in details from Phase III
• Check program correctness (debugging)
• Work on front end for user input

4. Phase 4: “Testing & Final touches” July 13th - August 3rd By this time I plan on being finished with the main programming effort. In these three weeks I will be putting final touches on the project. Including debugging, test cases (perhaps using JUnit), and ensuring that the code is fully commented with javadoc tags. Also catching up in any areas where we may have hit some pitfalls from the previous sections.

Tasks:
• Conduct Testing
• Final touches

5. Phase 5: “Documentation & Tutorials” August 3rd - August 17th During this phase I will construct tutorials (both video and html) to instruct users how to use the network statistics module. I will also produce documentation that will fully explain the different statistics that were created. It has been my experience at large that the code in and of itself is never the fully finished product.

Tasks:
• Create tutorials
• Create documentation
3.1 Milestones

To ensure timely completion of the project I have outlined several milestone deadlines to help move progress along. You will notice that there are relatively fewer milestones in the first few weeks, as this is time I will be spending getting to know the source code and understanding the network metrics. Below are 8 milestones and the date (month-day) when I expect to be at the point.

1. 6-8: High level design with class structures and main algorithms outlined.
2. 6-15: A working algorithm for the 4 graph-distance algorithms outlined in section 2.
3. 6-22: Clustering-coefficient & power-law distribution completed.
4. 7-6: HITS algorithm & PageRank completed.
5. 7-13: Modularity completed.
6. 8-3: Formalized test cases.
7. 8-3: Beta Version completed.
8. 8-17: Final Version completed with documentation.

3.2 Development Hurdles

I predict that the most significant hurdle will be learning the existing code in the initial phase of the project. For this reason I have devoted the first several weeks to exploring the Gephi source code. Implementing the algorithms described above will only be possible once I am able to understand the data structures and interfaces (API) that deal with networks.

4 Lessons Learned GSOC 2008

Last summer I participated in Google’s Summer of Code implementing a plugin for the open source application Cytoscape. The plugin’s main tasks were to create random networks, randomize existing networks and compare random networks to existing networks based on four network metrics: clustering coefficient, average shortest path, degree-distribution, and average degree. To date the plugin had been downloaded 500 times since September 2008.

While I do believe that the RandomNetworks plugin was a successful GSOC project, I learned a great deal from this experience; knowledge that I will be able to apply to this summer. First an intuitive graphical interface is worth a thousand efficiency improvements. It wont matter if I have implemented beautifully elegant algorithms that crunch numbers with blazing speed and accuracy if no one can understand how to use the interface for these tools. I plan to make the graphical interface an important aspect of this project.
Last summer each week I provided a progress report that stated what I had completed and any concerns that were raised during the week; I plan on continuing that policy this summer. Additionally I also plan on keeping a weekly blog of the experience that I think is helpful from an administrative point of view for keeping track of progress.

Lastly spending time in Phase 2 to outline the code at a high-level I think is very important. Last summer we had to refactor the code more than once to make the framework more reusable. For that reason I plan on implementing this project within a more general framework that can be easily extended to include other network metrics, and to keep the code as modular as possible so that each metric is a plug-able entity into the framework.

5 My Qualifications

The GSOC ’08 project I think was great preparation for this years project as they do have some overlapping algorithms, both were in Java. The plugin’s website is at http://sites.google.com/site/randomnetworkplugin. Furthermore, I have been involved with several class projects and internships that have exposed me to a great deal of programming theory and methodology. My dissertation research has me programming and running experiments almost every day.

**Academics:** As I have stated earlier, I am in my fourth year of PhD studies at Syracuse University majoring in Computer Science. I have completed my course work with a GPA of 3.97. I received my masters before entering the PhD program in 2004 from Syracuse University. For more details my CV can be found at http://web.ecs.syr.edu/~pjmcswee/cv.pdf.

**Languages:** My preferred language is java for two reasons. First, I work on several different machines I like the ability to move my code around as java is platform independent (although java versions do sometimes become bothersome). Secondly I also appreciate Java standardized data structures.

**Work Experience:** I have been a teaching assistant at Syracuse University for the last five years. This semester I am instructing CIS 586 Design of Operating Systems which is an introductory course into the design of operating systems, predominately composed of graduate students. My experience both as a teaching assistant and as an instructor have enforced the importance of organization and people skills. I have interned for four different companies: Coherent Networks, Osmose Inc, Telephonics and Lockheed Martin. (For more info see my CV). These experiences have emphasized the necessity for clear requirements and disciplined coding.

**Research:** My own research is geared towards community detection within complex
networks. The last two years have been spent immersing myself in graph theory and publications in the field of complex networks. Currently my advisors and I are working on a manuscript draft involving force-directed layouts and community detection. In the course of my research I have already implemented several of the algorithms listed in section 2.

Motivation: Part of my research currently calls for an investigation into the underlying nature of communities in complex networks and what kind of information is carried with modularized networks. While I have spent many hours parsing through papers on the subject, I have had little time to do a wider inquiry to gain the mastery of other common complex network analysis. I feel that this is something I will have to do before my dissertation defense, this summer would be a great time to learn about these other network metrics.