**Overview**

The project investigates the relationships between research papers hosted on Google Scholar. Relationships are determined by the citations of one research paper by another. The first part of the project will attempt to match papers through common citations. The second part of the project will attempt to match papers by leveraging more advanced methods of grouping graph nodes into communities.

**Data**

I will be using the *Google Scholar citation relations* data set, found in the Awesome Public Datasets GitHub repository (<https://github.com/caesar0301/awesome-public-datasets#social-networks>).

This dataset contains research papers (titles only) hosted on Google Scholar, and the citations in this paper to other papers in the dataset. I am unsure of the origins of the dataset, and therefore do not know how this dataset might have already been filtered from the entire Google Scholar project.

The direct link to the dataset is here: <http://www3.cs.stonybrook.edu/~leman/data/gscholar.db>

The dataset is stored in a SQLite 3 database, organized neatly as a graph defined such that:

Node: Research paper

Edge (Directed): Citation of a research paper by another research paper

Number of nodes (papers) = 82,937

Number of edges (citations) = 148,116

**Questions**

**Easier:** Given a paper, can we find similar papers by identifying those with common citations?

**Harder:** Given a paper, can we find similar papers by segregating the entire dataset into communities? How will this compare to our method from the easier question?

**Algorithms and Data Structures**

Main Data Structure: The network has been modeled as a graph through an adjacency list. Each node in the graph is a research paper, and a directed edge between nodes represents one paper’s citation of the other paper.

**Easier question:** To answer this question, we can begin by identifying the out-neighbors (cited papers) for the given node (paper). We will then search the rest of the data set, looking for nodes with common out-neighbors. The nodes with the most common out-neighbors could be the most similar papers.

**Harder question:** To answer this question, we are going to leverage a similar algorithm to the one Mia introduced. Then we’ll compare the papers within a community to those ‘similar’ papers from our easier implementation.

**Algorithm Analysis, Limitations and Risks**

**Easier Question:**

Identify out-neighbors for one node O(1). // Worst case is all other nodes are out-neighbors

For every other node: O(V)

Identify out-neighbors O(1) // Worst case is all other nodes are out-neighbors

Return the count of matches between this result and the first set O(V) // Everything matches

Reverse sort nodes by number of matches – O(Vlog(V))

O(1)+O(V2)+O(Vlog(V)) ~ O(V2)

One thing which should help improve the worst-case is the fact that papers can only cite papers that have already been written. Further, papers that have already been written cannot be modified to cite a paper written after the first paper. So technically I don’t think it would be O(V2) but more like O(Vlog(V)).

**Harder Question:**

I’m going to implement the algorithm that Mia introduced in the videos, which computes in O(V\*(V+E) time in the general case.

However, the real-world rule mentioned earlier allows us to take advantage of the fact that our dataset can be defined as a *Directed Acyclic Graph (DAG)*. A DAG is a directed graph that has a topological ordering, or a sequence of the nodes such that every edge is directed from earlier to later in the sequence. One advantage to a DAG is that the problems of finding shortest paths can be solved in *linear* time. It’s possible that I’ll be able to find a quicker algorithm for detecting communities.

**Risks:**

It does not seem valid for a node and its neighbor to cite each other, but I found instances of this in the data set. So, I cannot leverage any DAG optimizations unless I scrub the data.

The risk with this is that:

1. The dataset could be too large to run through and detect communities on my machine with the original generalized algorithm.
2. Identifying the invalid nodes/edges is too difficult / time-consuming.
3. Removing the invalid nodes/edges to turn this into a DAG compromises the integrity of the dataset and my findings.

Also, I did some preliminary research and found a paper describing one method for detecting communities in DAGs: *Detecting Communities in DAGs -* <https://arxiv.org/pdf/1503.05641.pdf>

The problem is that this method uses an Adjacency Matrix graph implementation, which doesn’t make sense for my data set.