Social and Collaborative Programming

COMP3121 Project Report

Co-authorship Network Analysis

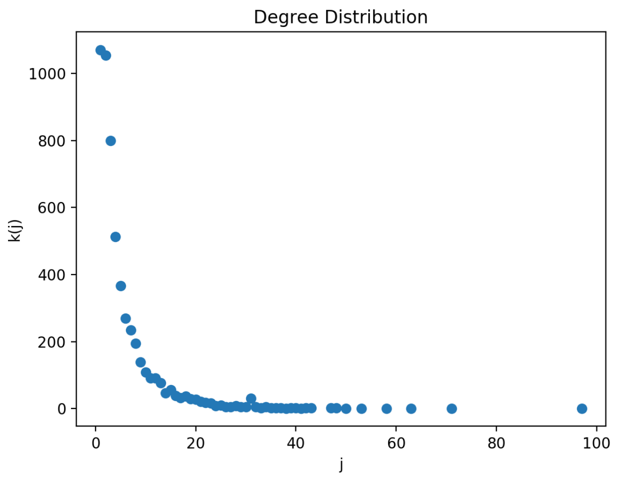
Network analysis techniques on a network dataset

ANGARA Jacky 15101317D
HO Tin Yau 15070597D
KHOSUMA Willy 15100967D

Contents

1 Network Degree Distribution

Our approach for degree distribution is fairly simple. We get the degree of each node and sort them out from the node with the highest degree to the node with the lowest degree. We then plot the graph based on the data. The result is shown below as a graph and we found out that it follows the power law distribution.



It means that it states the probability p_k of having a node with k neighbours is

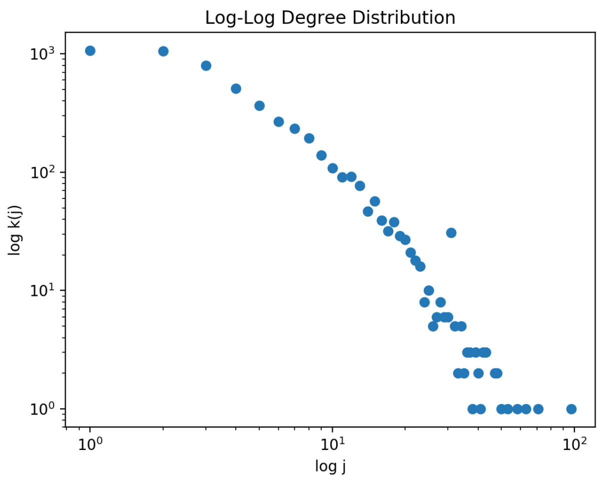
$$p_k = Ck^{-\alpha}$$

where C is a normalization constant and α is known as the exponent of the power law. Values in the range $2 \leq \alpha \leq 3$ are typical for degree distributions networks. Notice that, taking the logarithm of both members of the power law definition, we have:

$$\log p_k = -\alpha \log k + logC$$

Hence, if p_k follows a power law and we plot p_k as a function of k on log-log scales (both axes are logarithmic), then we should notice a straight line of slope $-\alpha$.

And below is our graph plot for the log-log scale of the graph:



This indicates that in the co-authorship network, there exists a lot of users with few connections or authorship while there also exists a few users with lots of connections of authorship. Another way to see that this is a power-law distribution is to plot the graph in a log-log scale. If we get a a linear relationship, then the graph represents a power-law distribution. In this case, if we observe the graph above we can see a straight line formed by the data set. We can conclude that this graph is a scale-free network.

We also found out that the graph has an average of 5.4 degree per node. This number will be used in the latter part of this report.

2 Clustering Co-efficiency

Average Clustering Co-efficiency = 0.46404472236602806

For Clustering Coefficient, our approach is to use the NetworkX Clustering function [1] to get the value of average clustering coefficient of the graph. The formula used for the function is shown below:

$$C = \frac{1}{n} \sum_{v \in G} c_v$$

The result is displayed as above. As we know, Clustering Coefficient measures the transitivity in a graph or the number of triangles in a graph. When applied to a single node, it measures how complete the neighborhood of a node is. We then apply this to the entire network to get the average clustering coefficient over all of the nodes in the network.

As we can observe from the result, the value of the average clustering coefficient is not that high at 0.46. This depicts that there isn't a lot of triangles / cliques. In real life it means that there are not a lot of groups that actually knows each other.

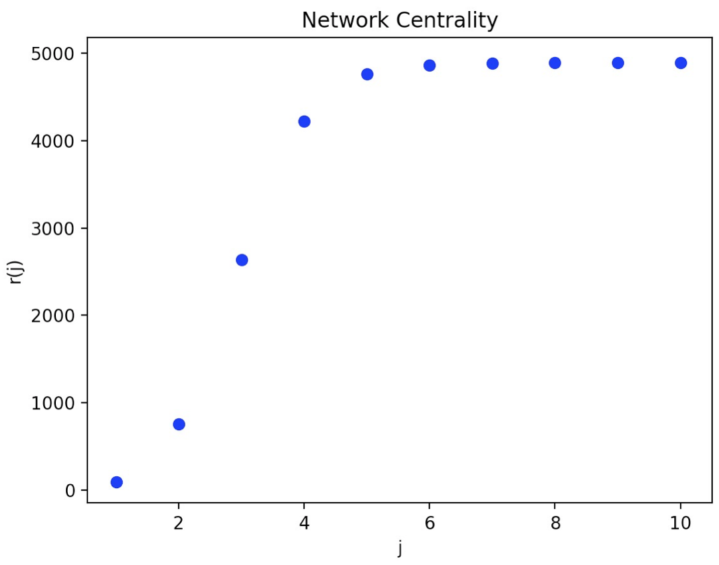
3 Shortest Path Length

Average shortest path length = 4.975054620953341

We first use breadth first search (BFS) to derive the shortest paths of each node to all other nodes. After traversing the whole graph, we take the average of those shortest paths to get the graph average shortest path length. Based on the result that we get, the average shortest path length is roughly equal to 5. When we compare it to the famous Milgram's small world experiment, the findings are actually close to one another since Milgram's experiment has the average shortest path of 6. It means that the average distance between one author to another author is typically around 5 nodes/authors.

According to Watts and Strogatz [2], the average path length between two nodes in a random network is equal to lnN/lnK, where N = total nodes and K = acquaintances per node. The total number of nodes in the graph is 5457 and the number of acquaintances per node is the average degree distribution we acquire in section 1 which is 5.4. Inserting the numbers to the formula will return 5.1 as the average path length of the graph. Our calculated average shortest path length is very near to the formulated average path length on a random network.

4 Network Centrality



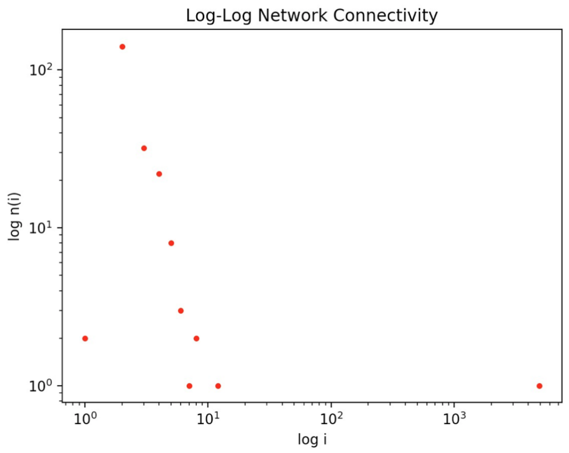
Centrality is a a measure that tell us how influential or significant a node is within the overall network. Our approach is to calculate the closeness centrality [3] in the graph and pick the node with the highest centrality as the centric node. The formula to calculate the closeness centrality is shown below:

$$C_i = \frac{1}{l_i} = \frac{n}{\sum_j d_{i,j}}$$

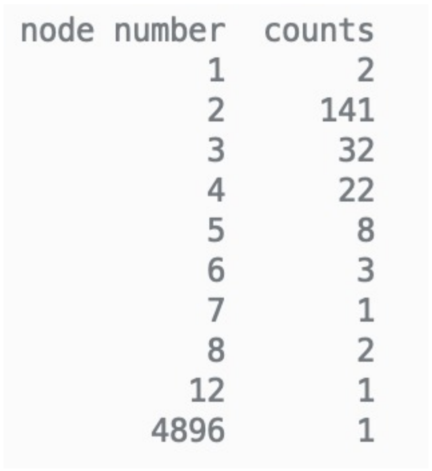
In this co-authorship network, Lee is the centre node of the whole network. After we define Lee as the centric node, we run the BFS algorithm [4] from him to derive his neighbors by levels. Meaning that at first we will find Lee's neighbors, and then finding the neighbors of his neighbors and so on.

The graph depicts clearly the way the graph connections grow. Because at first we only have Lee as the node, the projected data will be low. But as we explore Lee's neighbors and their neighbors the plot starts to grow exponentially as shown in the graph. This happens because as we explore different edges, the number of connections start growing bigger as there are more nodes to explore. However, after exploring the 4th level of neighbors from Lee, we can see a more stagnant growth because it almost reaches the population limit. Thus creating the graph displayed above.

5 Network Connectivity



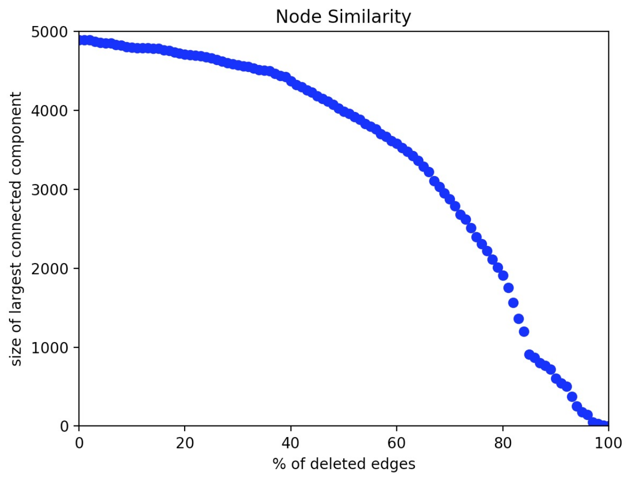
In order to gain more insight about the network connectivity, we plot the largest component of the ordered pair $(log\,i, log\,n_i)$ where n_i is the number of connected components of size i. After we plot the graph, we get the results above. To complement with the graph that we created, we will also provide an additional information regarding the number represented on the graph below:



node number	counts
1	2
2	141
3	32
4	22
5	8
6	3
7	1
8	2
12	1
4896	1

As we can observe based on the data above, we found out that there exist 2 singleton in the graph (meaning it is a node without any edges or connections), followed by a high spike of 141 numbers of nodes with one connection, and the number goes on until we found one large component consisting of 4896 nodes. These numbers explains the graph earlier as we can detect the singleton by the red dot on the bottom left, followed by a spike of 141 numbers of 2 nodes and it keeps going down, boiling down to the far bottom right we can find the large component appearing as an outlier.

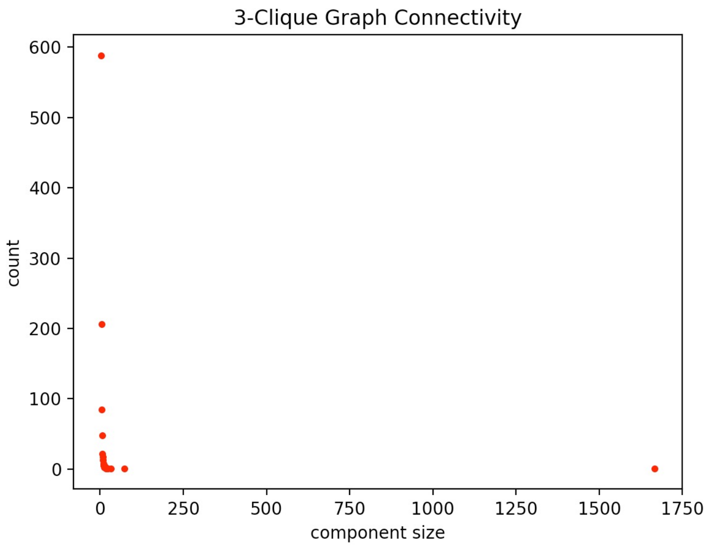
6 Node Similarity



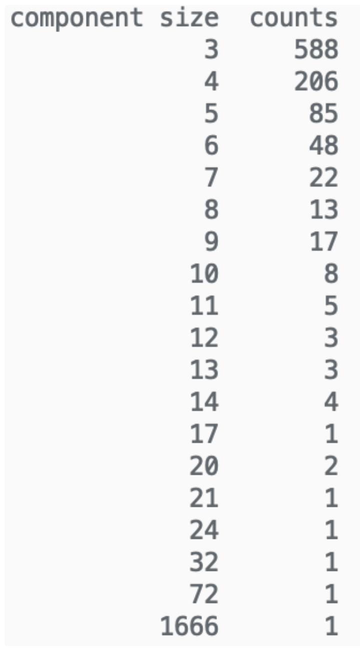
Similarity measures the number of similar nodes that are grouped together. We are interested in plotting the largest components in the graph and after that removing some percentages of the edges and see how does that affects the largest component of the graph. Our approach is to get the shared neighbors of node i and j. We are using the NetworkX common_neighbor function [5] to get the shared neighbors. We then plot the largest components we get from these connections and gradually decreasing the number of edges and plot the components again to compare. The results we have depicts that as we delete some percentages of the edges inside the graph, the largest components gradually goes down to. This actually makes sense because as we delete edges, it means that every nodes will have less connections, thus making the component of connected nodes even smaller.

7 Community Discovery

7.1 Community Detection Through 3-Clique



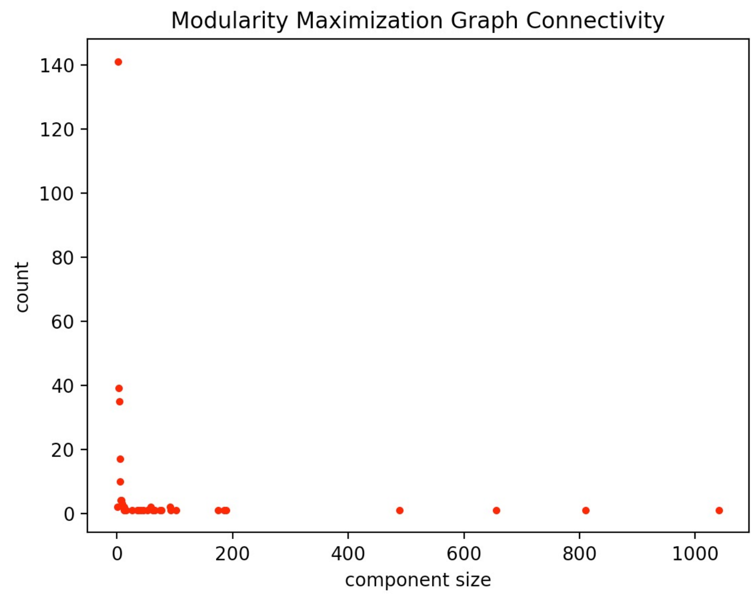
For the first community detection, we use the k-clique technique [6] to identify the community. We pick k=3 because it has better representations over other k. The result is plotted on the graph above. As we can observe, most of the 3-cliques communities are found on the left side of the graph. Below is the statistics representation of the graph above for a better understanding:



component size	counts
3	588
4	206
5	85
6	48
7	22
8	13
9	17
10	8
11	5
12	3
13	3
14	4
17	1
20	2
21	1
24	1
32	1
72	1
1666	1

As seen above, the 3-clique of size 3 accounts for the highest number as depicted on the graph with the dot on the top left. The number goes down drastically to 286 for clique of size 4, and it gradually goes down until clique of size 72. The interesting part is after clique size of 72, it directly jumps to clique size of 1666 as depicted on the bottom right dot on the graph. This shows that the graph contains only a handful of components in which the largest shortest path distance between any node is $<= k$.

7.2 Modularity Maximization



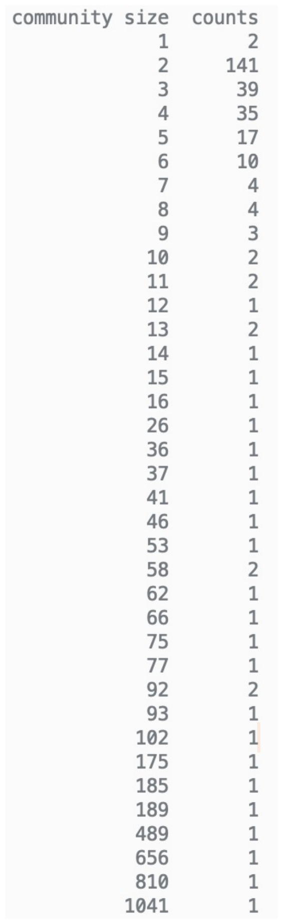
For our Modularity Maximization, we use Clauset-Newmann-Moore greedy algorithm [7]. The formula is depicted as below:

$$Q = \frac{1}{2m} \sum_{i,j} \left(A_{i,j} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) = \frac{1}{2m} \sum_{i,j} B_{i,j} \delta(c_i, c_j)$$

where:

$$B_{i,j} = A_{i,j} - \frac{k_i k_j}{2m}$$

and B is called the modularity matrix. The rseult of the program is shown with the graph above. With the assumption that real-world network is far from random, the more distant a graph is from a random generated network, the more structured they are. Hence, Modularity measures this distance while Modularity Maximization is maximizing this distance. Above is the result of the graph we are able to project. Based on the plot above, we can see that in this network there are a lot of segmentation skewed to the right, meaning that most communities are based on <= 200 components. We provide a statistical representation of the graph below to further understand it:



```
community size  counts
             1       2
             2     141
             3      39
             4      35
             5      17
             6      10
             7       4
             8       4
             9       3
            10       2
            11       2
            12       1
            13       2
            14       1
            15       1
            16       1
            26       1
            36       1
            37       1
            41       1
            46       1
            53       1
            58       2
            62       1
            66       1
            75       1
            77       1
            92       2
            93       1
           102       1
           175       1
           185       1
           189       1
           489       1
           656       1
           810       1
          1041       1
```

As we can observe from the statistics above, most communities are formed at the size of 2-6, while others are more spread out at a bigger number. This is expected because if we reduce the number of components inside the graph, the more likely it is to form a community since the scope is smaller. If we increase the number of components, then finding mutual connections between each other will be harder, thus creating a more generalized community in the end.

8 Group Members Contributions

Name	Coding	Report	Research	Discussion
ANGARA Jacky	62%	40%	40%	35%
HO Tin Yau	10%	10%	25%	25%
KHOSUMA Willy	28%	50%	35%	40%
TOTAL	100%	100%	100%	100%

ANGARA Jacky **(Coding)**
Jacky analyzes the network and visualize the result. The data will then be sent to Willy for reporting. Jacky does research on several NetworkX built-in functions and apply them for network analytics. Jacky provides useful information for decision making in making both the program and the report.

KHOSUMA Willy **(Reporting)**
Willy analyzes the data given by Jacky and put them together into one cohesive report. This includes reformatting the report into LaTeX, conduct research on various network analysis methods - formula - and results reporting.

HO Tin Yau **(Support Research & Discussion)**
Ho Tin You provides support by giving some research data about our statistical findings.

References

[1] NetworkX. (2018). Clustering, [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/clustering.html.

[2] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, p. 440, 1998.

[3] NetworkX. (2018). Closeness centrality, [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.closeness_centrality.html#networkx.algorithms.centrality.closeness_centrality.

[4] ——, (2018). Breadth first search, [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.traversal.breadth_first_search.bfs_edges.html#networkx.algorithms.traversal.breadth_first_search.bfs_edges.

[5] ——, (2018). Common neighbor, [Online]. Available: https://networkx.github.io/documentation/networkx-1.9/reference/generated/networkx.classes.function.common_neighbors.html.

[6] ——, (2018). K-clique, [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.kclique.k_clique_communities.html#networkx.algorithms.community.kclique.k_clique_communities.

[7] ——, (2018). Clauset-newman-moore greedy modularity maximization, [Online]. Available: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.modularity_max.greedy_modularity_communities.html#networkx.algorithms.community.modularity_max.greedy_modularity_communities.