Assignment 1: Analyzing Qualities of Randomly Generated Graphs

Matthew MacEachern
York University
4700 Keele St, Toronto, ON M3J 1P3
Canada
mcmaceac@my.yorku.ca

KEYWORDS

Random graph models, Erdős–Rényi model, Watts–Strogatz model, Barabási–Albert model, shortest path, clustering coefficient, node degree, community detection.

1 INTRODUCTION

In this assignment, we were asked to generate 45 random graphs based on three well known graph models to analyze properties associated with each graph and to see how certain properties change when the input parameters to these graphs differ. These properties include the degree distribution of each graph, the local and global clustering coefficient of each graph, the average shortest path as well as the shortest path distribution of each graph, the diameter of each graph, and finally the size of each community when the graphs are split into k=10 communities.

In order to generate and analyze these graphs, the Python library NetworkX was utilized. Throughout this report, the various NetworkX functions used will be discussed in detail, as well as the results that they provided. A total of 18 graphs were generated (five of each model with 10,000 nodes, as well as one of each model with 1,000 nodes to analyze communities in a reasonable amount of time). These graphs can be found along with the plots obtained from the graphs in an adjacency list file which can be later loaded and further analyzed if needed. The parameters chosen to create these graphs (aside from the number of nodes n) were chosen to have a range of values that would give a wider variety of results in order to analyze the changes that occur to the various measurements depending on these parameters.

2 GRAPH MODEL GENERATORS

2.1 Erdős–Rényi Model

To obtain an Erdős–Rényi graph using NetworkX, the erdos_renyi_graph(n, p) function was used. This allowed me to simply provide parameters to the function and allow NetworkX do the heavy lifting of generating the graph. The parameter n was specified to be the number of nodes present in the graph. The parameter p was specified to be the probability that an edge between two nodes is included in the graph independent of any other edge in the graph.

As mentioned previously, a total of six Erdős–Rényi graphs were generated. Five with 10,000 nodes, and one with 1,000 nodes.

ER1 was created with n=10,000 and p=1/6. Unbeknownst to myself at the time of creating the graph, this would create an incredibly dense graph as the probability value is quite high for a graph with many nodes. This created a graph with many more edges than anticipated but was still useful for analysis purposes. ER2 was created with n=10,000 and p=1/100, which created a sparser graph, and gave some interesting results. ER3 was created with n=10,000 and p=1/125. ER4 was created with n=10,000 and p=1/200. ER5 was created with n=10,000 and p=1/50. Finally, ER6 was created with n=1000 and p=1/10, for reasons to be discussed in the community detection section.

2.2 Watts–Strogatz Model

To obtain a Watts–Strogatz graph using NetworkX, the watts_strogatz_graph(n, k, p) function was used. Similar to the Erdős–Rényi graph generation process, the parameter n was specified to be the number of nodes in the graph. The parameter k was specified to be the number of nearest neighbors that each node would be connected to in a ring topology. Finally, the parameter p was specified to be the probability that an edge would rewire to another node.

WS1 was created with n=10,000, k=4, and p=1/6. WS2 was created with n=10,000, k=100, and p=1/10. WS3 was created with n=10,000, k=200, and p=1/16. WS4 was created with n=10,000, k=10, and p=1/5. WS5 was created with n=10,000, k=20, and p=1/50. Finally, WS6 was created with n=1,000, k=20, and p=1/5.

2.3 Barabási–Albert Model

To obtain a Barabási–Albert graph using NetworkX, the barabasi_albert_graph(n, m) function was used. As always, the parameter n was specified to be the number of nodes in the graph. The parameter m was specified to be the number of edges to attach from a new node in the graph to any existing nodes.

BA1 was created with n=10,000, and m=5. BA2 was created with n=10,000, and m=10. BA3 was created with n=10,000, and m=20. BA4 was created with n=10,000, and m=50. BA5 was created with n=10,000, and m=2. Finally, BA6 was created with n=1,000 and m=10.

3 GRAPH MEASUREMENTS

3.1 Degree Distribution

The degree distribution is an important aspect of any graph and gives intuition as to how connected a graph is. This distribution was found using NetworkX's degree_histogram(G) function.

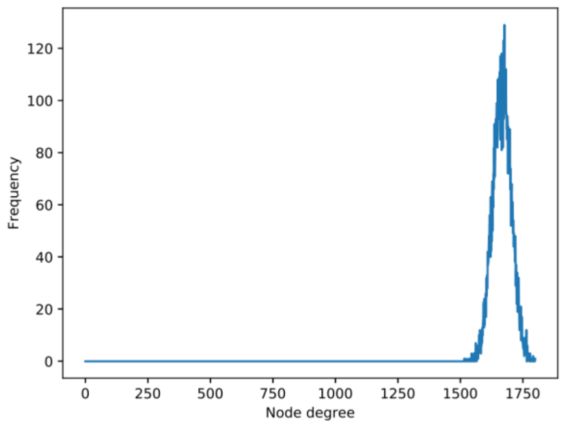

Fig 1. Degree distribution for ER1(n=10000,p=1/6)

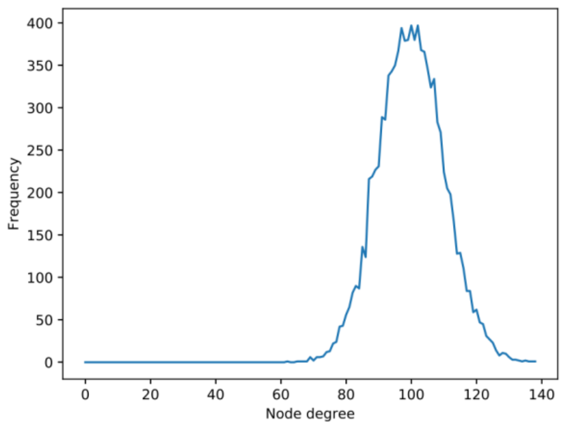

Fig 2. Degree distribution for ER2(n=10000,p=1/100)

The degree distributions for the Erdős–Rényi graphs are shown in Figures 1-5. Unsurprisingly, the distributions seem to be closely correlated to the n and p parameters. With the distribution following a bell-shaped curve with the most frequent degree being equal to n*p.

The degree distributions for the Watts–Strogatz graphs are shown in Figures 6-10, and seem to follow the k parameter, which intuitively makes sense. K is the number of neighbors that each node connects to. So, the initial degree will be the k value, and subsequent degree additions will be determined by the p parameter which dictates if an edge will rewire to another node.

The degree distributions for the Barabási–Albert graphs are shown in Figures 11-15. All of the distributions seem to have relatively similar shapes. For the most part, all of the nodes have a small degree, which has certain implications for the other measurements of the graph.

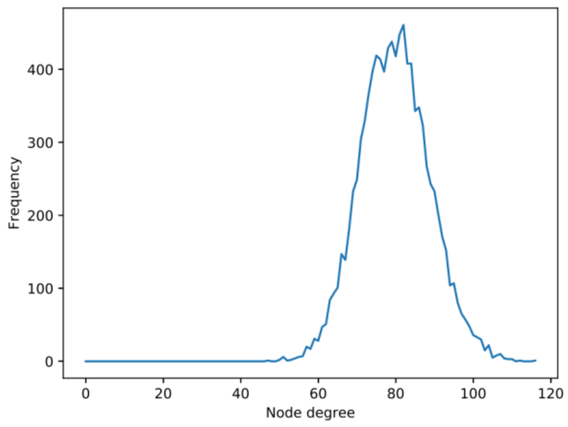

Fig 3. Degree distribution for ER3(n=10000,p=1/125)

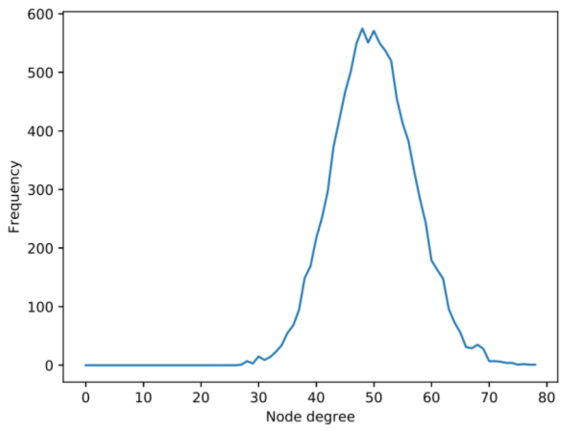

Fig 4. Degree distribution for ER4(n=10000,p=1/200)

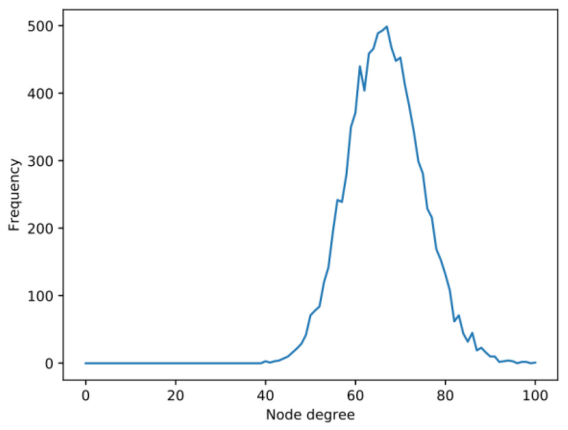

Fig 5. Degree distribution for ER5(n=10000,p=1/150)

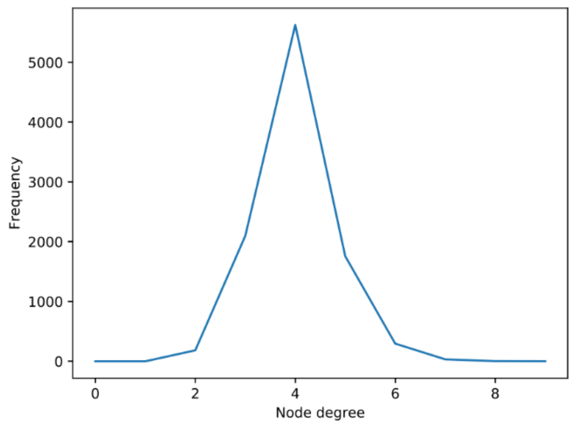

Fig 6. Degree distribution for WS1(n=10000,k=4,p=1/100)

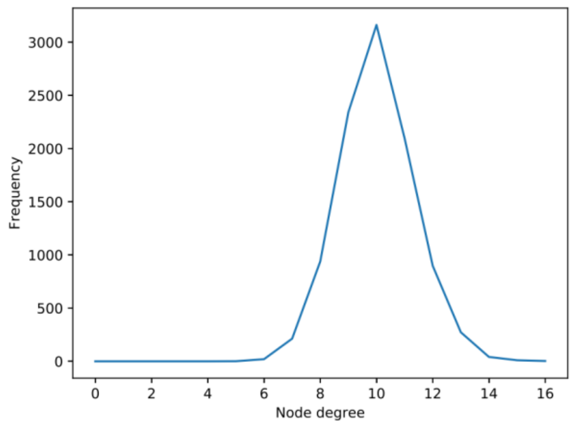

Fig 9. Degree distribution for WS4(n=10000,k=10,p=1/5)

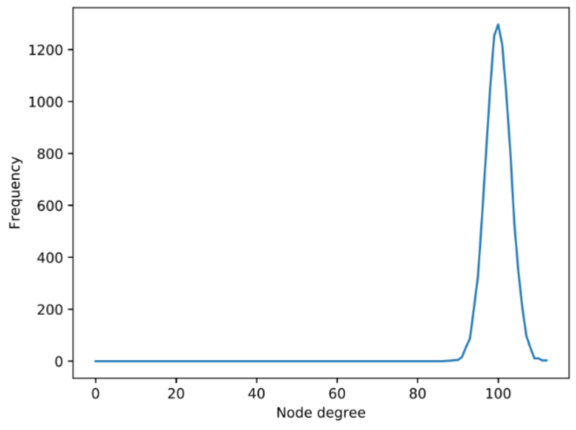

Fig 7. Degree distribution for WS2(n=10000,k=100,p=1/10)

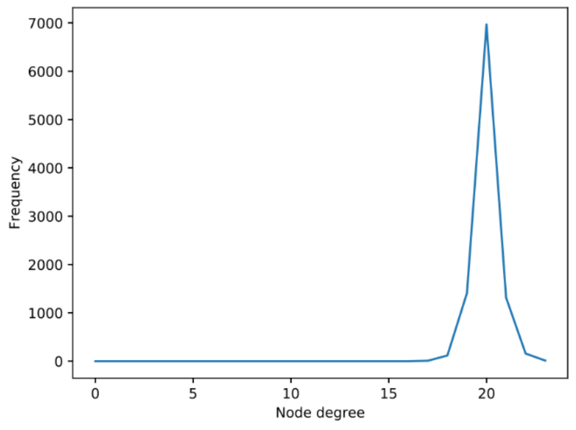

Fig 10. Degree distribution for WS5(n=10000,k=20,p=1/50)

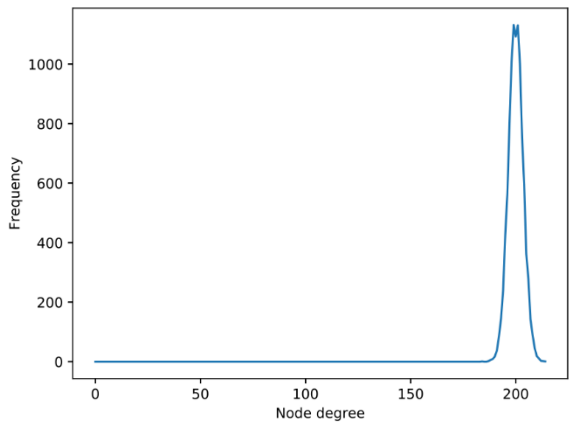

Fig 8. Degree distribution for WS3(n=10000,k=200,p=1/16)

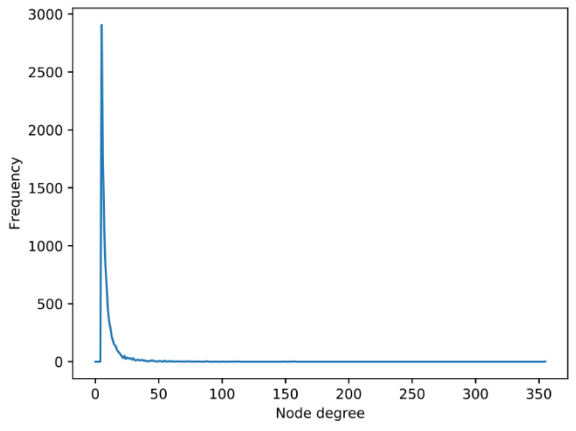

Fig 11. Degree distribution for BA1(n=10000,m=5)

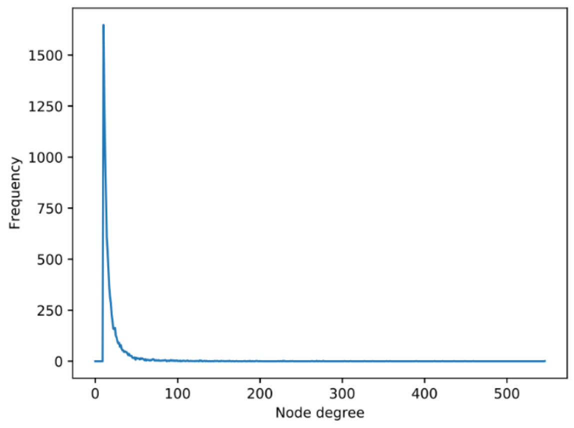

Fig 12. Degree distribution for BA2(n=10000,m=10)

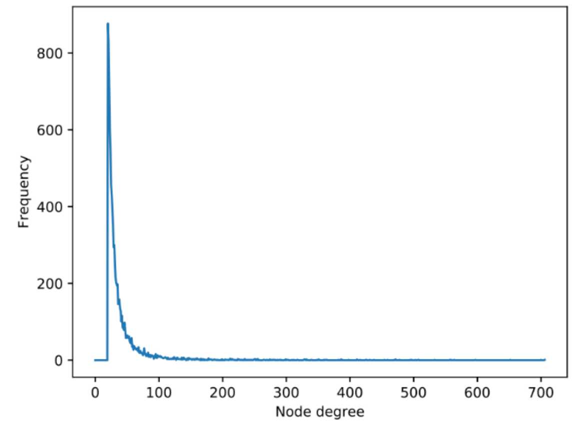

Fig 13. Degree distribution for BA3(n=10000,m=20)

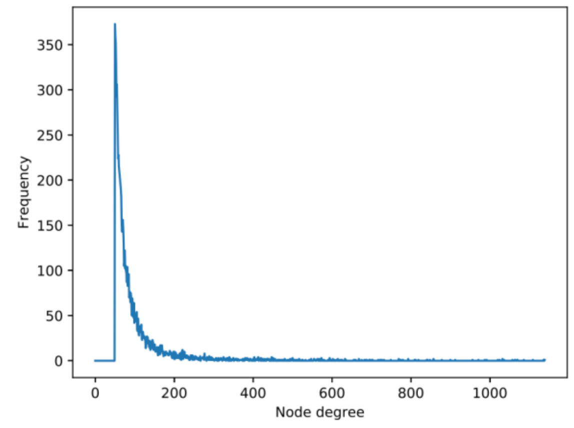

Fig 14. Degree distribution for BA4(n=10000,m=50)

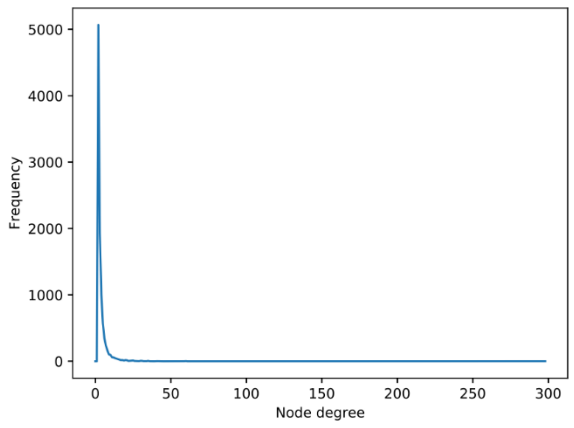

Fig 15. Degree distribution for BA5(n=10000,m=2)

3.2 Local Clustering Coefficient Distribution

The local clustering coefficient distribution can give a sense of how close neighbors of a node are to being a clique, or a complete graph. This distribution was found using NetworkX's clustering(G) function.

The local clustering coefficient distributions for the Erdős–Rényi graphs are shown in Figures 16-20. The clustering coefficient for most nodes seems to be equal to the p parameter of the graph, which larger deviations being present when the p value is smaller. For example, in Figure 16 where p=1/6, the distribution seems to have equal values for every node, while in Figure 19 with p=1/200 there is larger variation between nodes.

The local clustering coefficient distributions for the Watts–Strogatz graphs are shown in Figures 21-25. Compared to the Erdős–Rényi graphs, the clustering coefficient for each node tends to be higher. A particularly interesting plot of distributions came from WS1 in Figure 21. There seems to be four distinct tiers of clustering coefficient values for every node which could be correlated to the k parameter being 4 in this case.

The local clustering coefficient distributions for the Barabási–Albert graphs are shown in Figures 26-30. The clustering coefficients tend to be very low compared to the other graphs. This could be due to the parameter value chosen for m. In particular, Figure 30 shows the distribution when m=2 (although it is very sparse and may be hard to see). The nodes in the graph either have a zero for the clustering coefficient value or a one. This means that there is a small subset of node that are a clique.

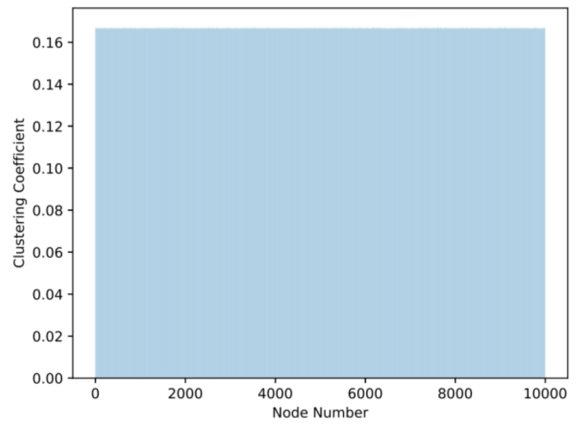

Fig 16. Local clustering coefficient distribution for ER1(n=10000,p=1/6)

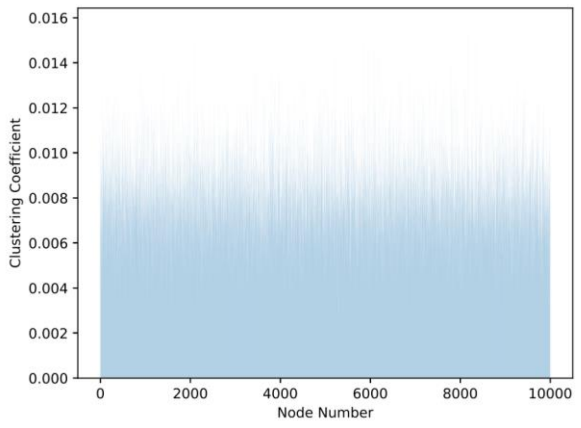

Fig 18. Local clustering coefficient distribution for ER3(n=10000,p=1/125)

3.3 Global Clustering Coefficient

In this section the global clustering coefficient for each of the fifteen graphs generated will be given. To obtain this value, NetworkX's average_clustering(G) function was used. For brevity, I will use GCC to refer to the global clustering coefficient, and round the values to three decimal places. The exact values can be found in the _data.txt files for each graph in the assignment zip folder.

ER1's GCC=0.167, ER2's GCC=0.010, ER3's GCC=0.008, ER4's GCC=0.005, ER5's GCC=0.007. WS1's GCC=0.297, WS2's GCC=0.542, WS3's GCC=0.616, WS4's GCC=0.348, WS5's GCC=0.669. BA1's GCC=0.007, BA2's GCC=0.011, BA3's GCC= 0.018, BA4's GCC=0.035, BA5's GCC=0.005.

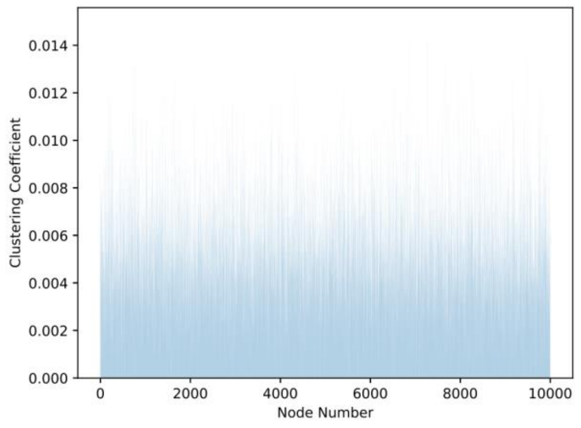

Fig 19. Local clustering coefficient distribution for ER4(n=10000,p=1/200)

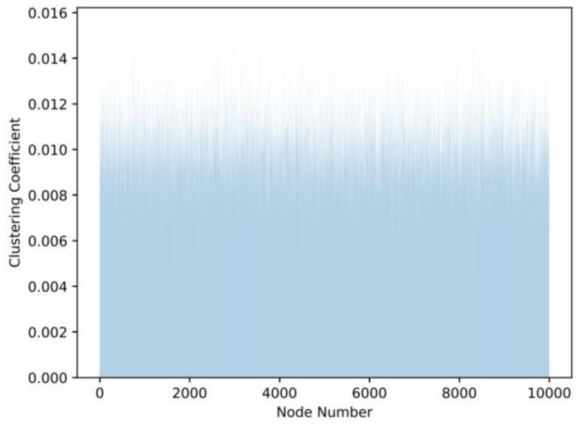

Fig 17. Local clustering coefficient distribution for ER2(n=10000,p=1/100)

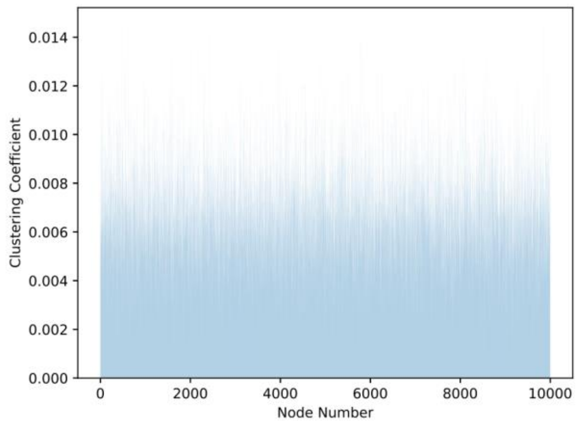

Fig 20. Local clustering coefficient distribution for ER5(n=10000,p=1/150)

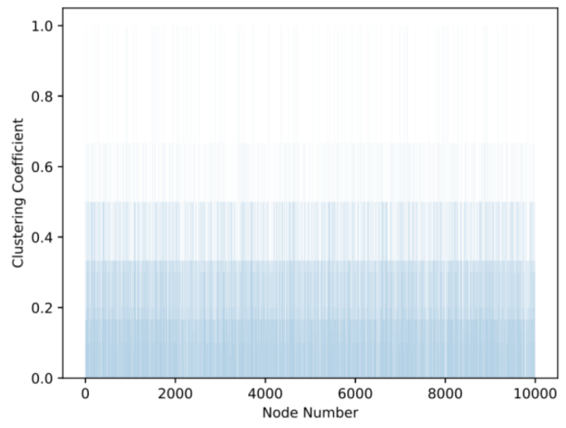

Fig 21. Local clustering coefficient distribution for WS1(n=10000,k=4,p=1/100)

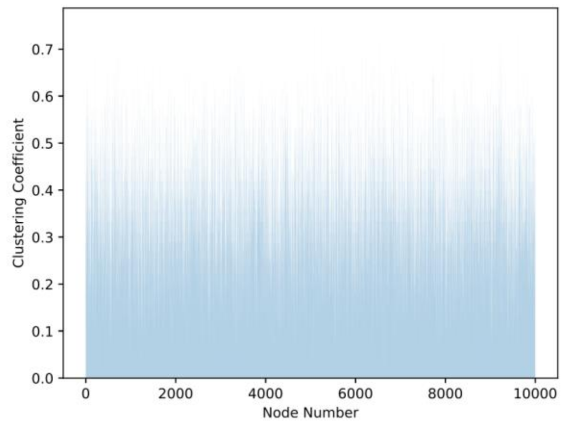

Fig 24. Local clustering coefficient distribution for WS4(n=10000,k=10,p=1/5)

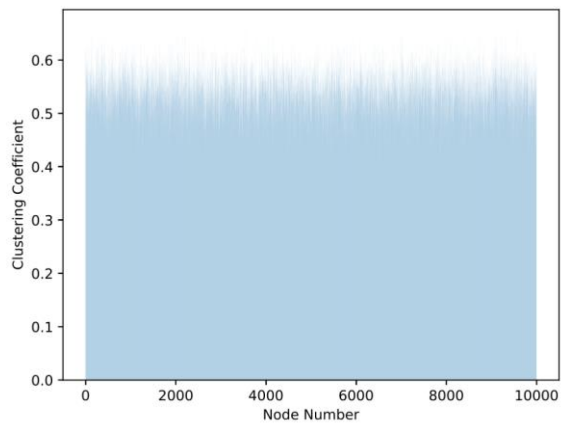

Fig 22. Local clustering coefficient distribution for WS2(n=10000,k=100,p=1/10)

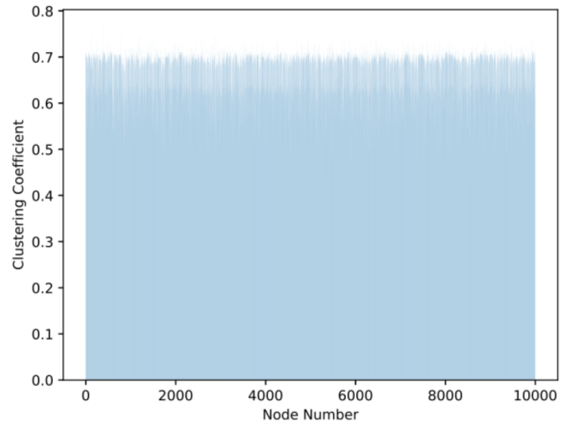

Fig 25. Local clustering coefficient distribution for WS5(n=10000,k=20,p=1/50)

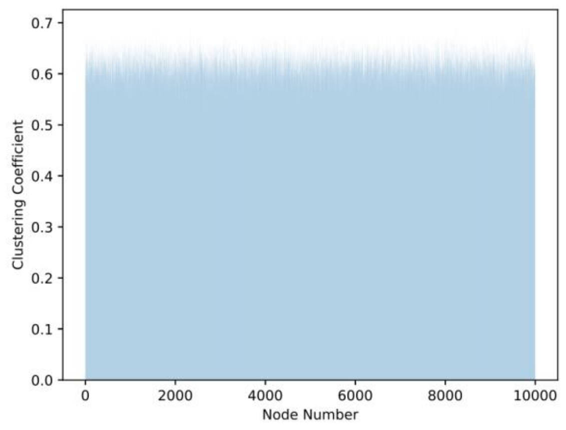

Fig 23. Local clustering coefficient distribution for WS3(n=10000,k=200,p=1/16)

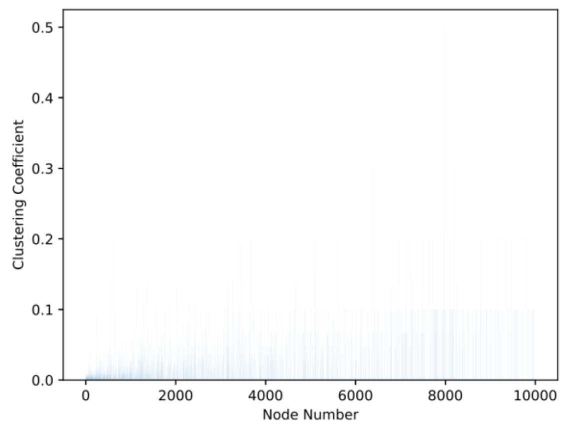

Fig 26. Local clustering coefficient distribution for BA1(n=10000,m=5)

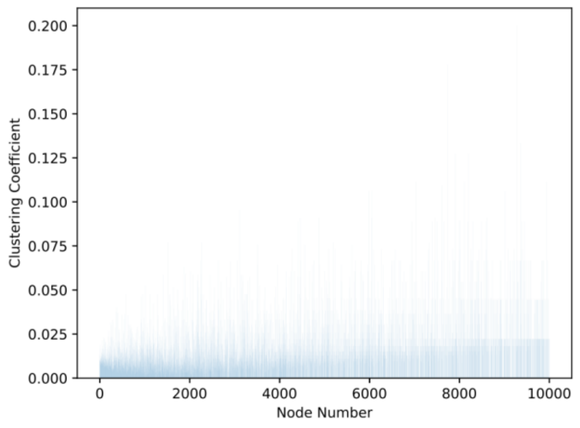

Fig 30. Local clustering coefficient distribution for BA2(n=10000,m=10)

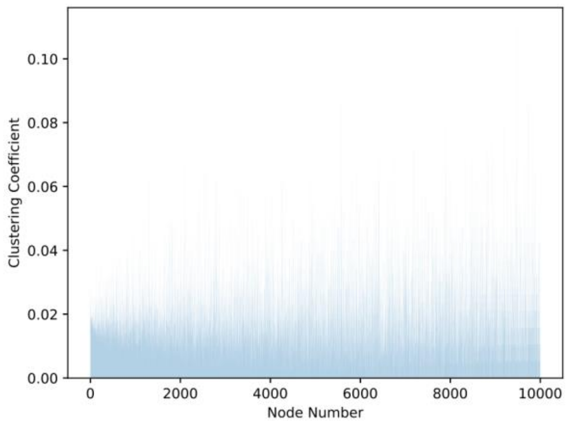

Fig 28. Local clustering coefficient distribution for BA3(n=10000,m=20)

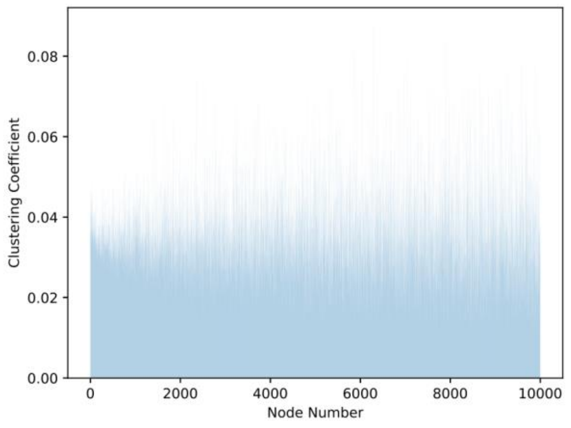

Fig 29. Local clustering coefficient distribution for BA4(n=10000,m=50)

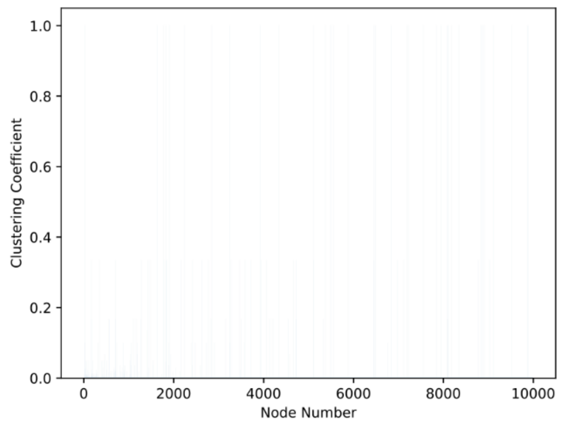

Fig 30. Local clustering coefficient distribution for BA5(n=10000,m=2)

3.4 Shortest Path Length Distribution

The shortest path distribution of a graph can give a sense of how far apart nodes are in a graph. If there are more shortest paths of a certain value, the diameter of the graph is most likely to be around that value. This distribution was found using NetworkX's extract_length_distribution(G) function.

The shortest path distributions for the Erdős–Rényi graphs are shown in Figures 31-35. For most of the graphs, the most frequent shortest path was usually two or three, with the longest shortest path being four when p=1/200. Intuitively this makes sense since there is a very high chance for nodes to be closely connected since the edges are random between any two nodes.

The shortest path distributions for the Watts–Strogatz graphs are shown in Figures 36-40. These distributions varied largely from the Erdős–Rényi graphs. The number of longer shortest paths seemed to depend on the number of neighbors that each node connected to k, as well as the rewiring parameter p. If a node was only connected to few neighbors and didn't have a high chance to rewire, it would take longer to get to nodes that were not part of its neighborhood.

The shortest path distributions for the Barabási–Albert graphs are shown in Figures 41-45. The value of m tended to dictate the most common shortest path length, with lower m values correlating to longer shortest paths, and higher m values correlating to shorter shortest paths.

3.5 Average Shortest Path Length

In this section the average shortest path length will be given for each graph. These values were obtained using NetworkX's average_shortest_path_length(G) function. For brevity, I will refer to a graph's average shortest path length as SPL, and the value will be rounded to two decimal places.

ER1's SPL=1.83, ER2's SPL=2.35, ER3's SPL=2.52, ER4's SPL=2.77, ER5's SPL=2.63. WS1's SPL=10.21, WS2's SPL=2.80, WS3's SPL=2.57, WS4's SPL=5.24, WS5's

SPL=6.09. BA1's SPL=3.65, BA2's SPL=3.07, BA3's SPL=2.76, BA4's SPL=2.30, BA5's SPL=4.99.

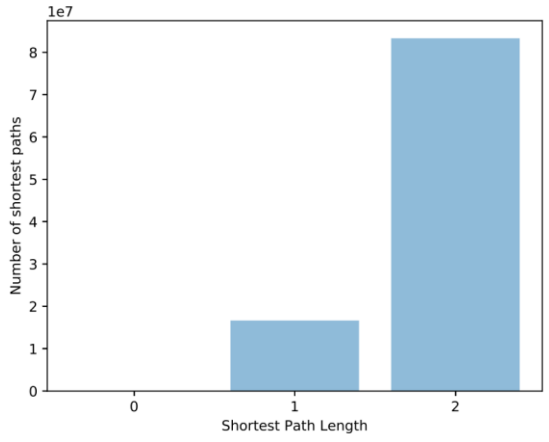

Fig 31. Shortest path distribution for ER1(n=10000,p=1/6)

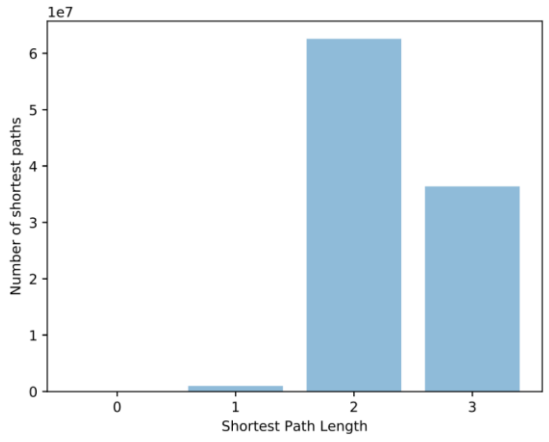

Fig 32. Shortest path distribution for ER2(n=10000,p=1/100)

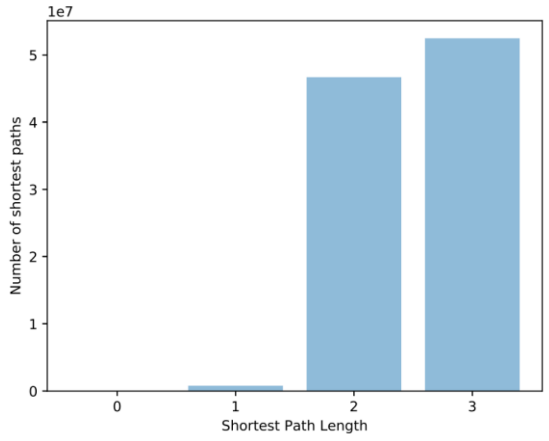

Fig 33. Shortest path distribution for ER3(n=10000,p=1/125)

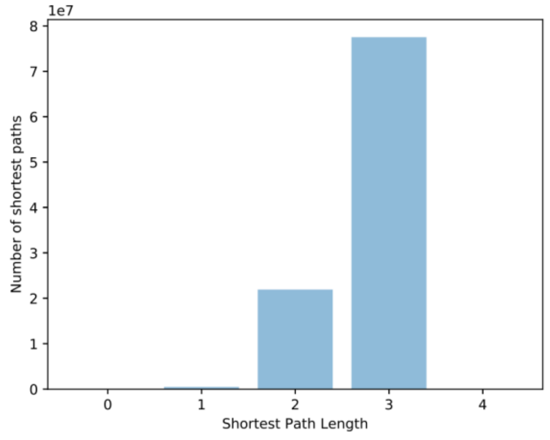

Fig 34. Shortest path distribution for ER4(n=10000,p=1/200)

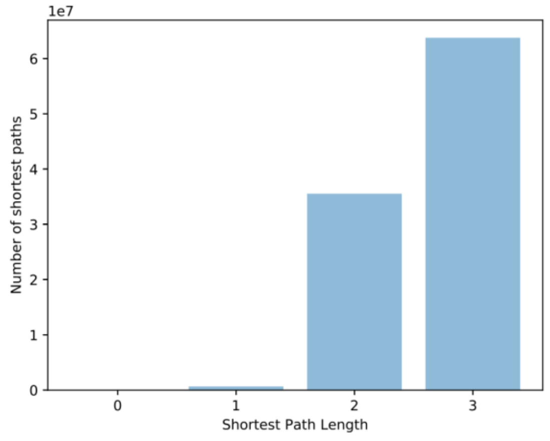

Fig 35. Shortest path distribution for ER5(n=10000,p=1/150)

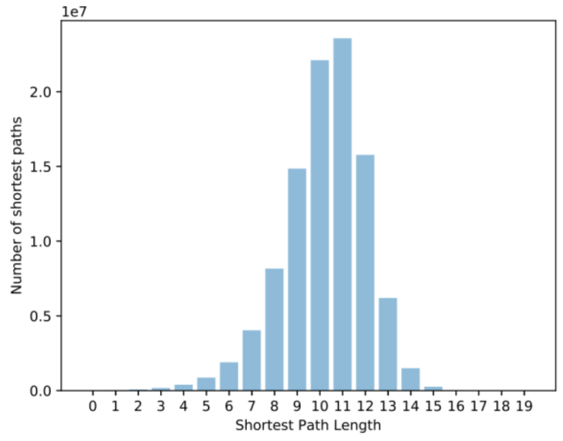

Fig 36. Shortest path distribution for WS1(n=10000,k=4,p=1/100)

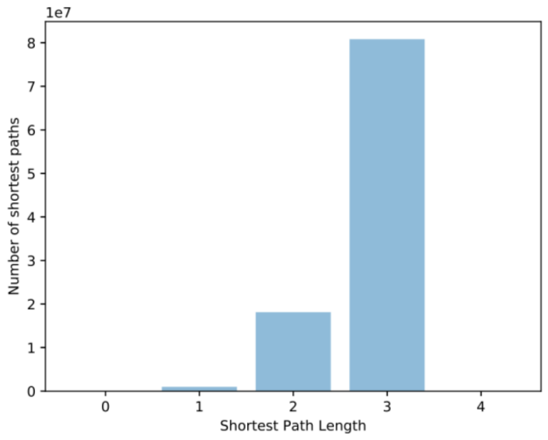

Fig 37. Shortest path distribution for WS2(n=10000,k=100,p=1/10)

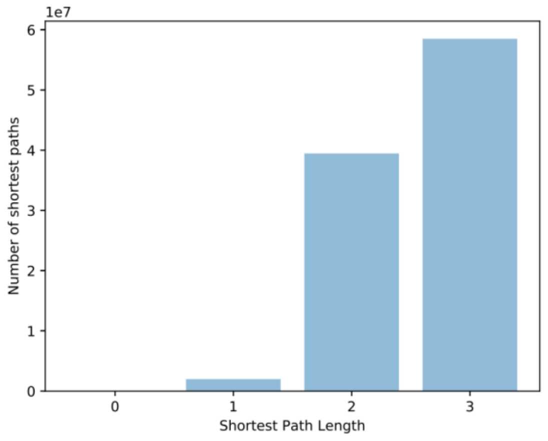

Fig 38. Shortest path distribution for WS3(n=10000,k=200,p=1/16)

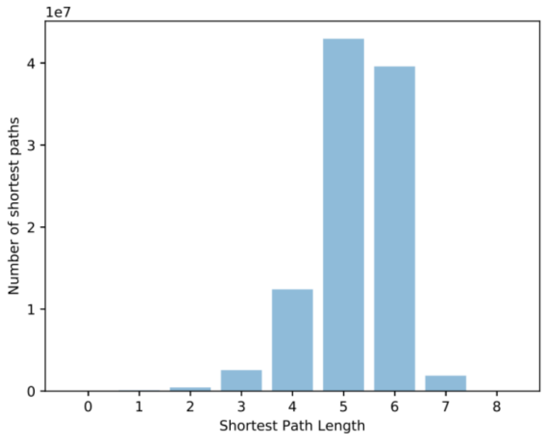

Fig 39. Shortest path distribution for WS4(n=10000,k=10,p=1/5)

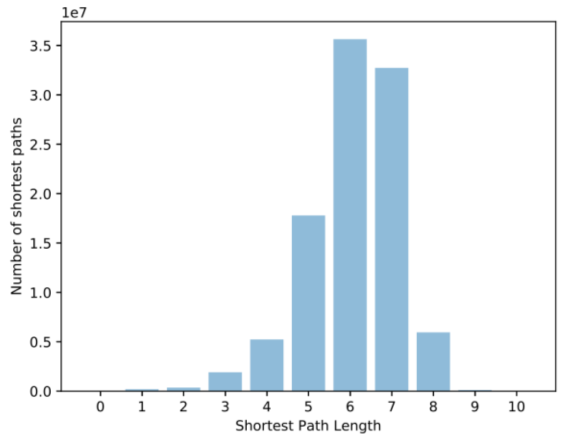

Fig 40. Shortest path distribution for WS5(n=10000,k=20,p=1/50)

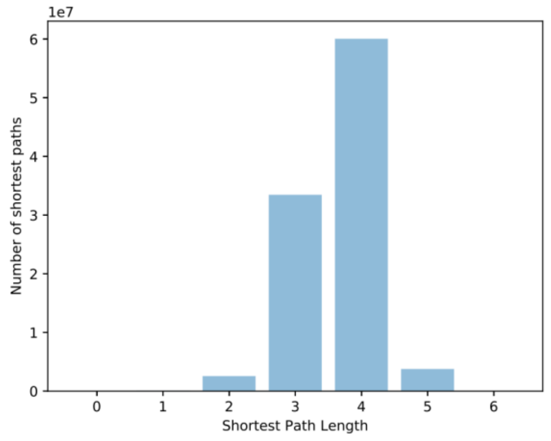

Fig 41. Shortest path distribution for BA1(n=10000,m=5)

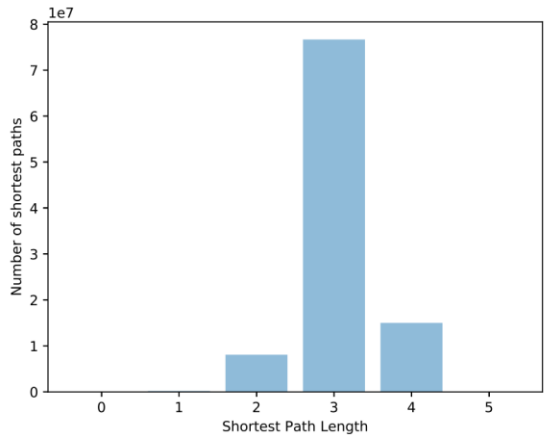

Fig 42. Shortest path distribution for BA2(n=10000,m=10)

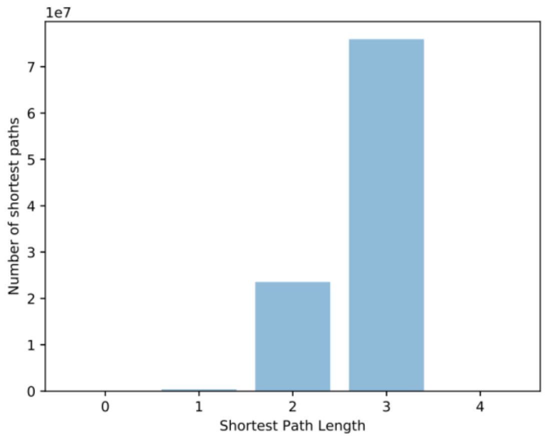

Fig 43. Shortest path distribution for BA3(n=10000,m=20)

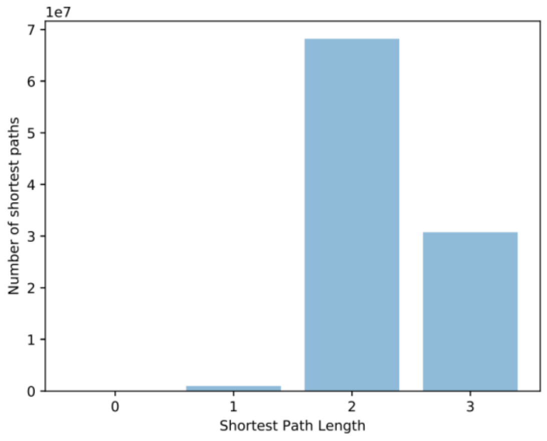

Fig 44. Shortest path distribution for BA4(n=10000,m=50)

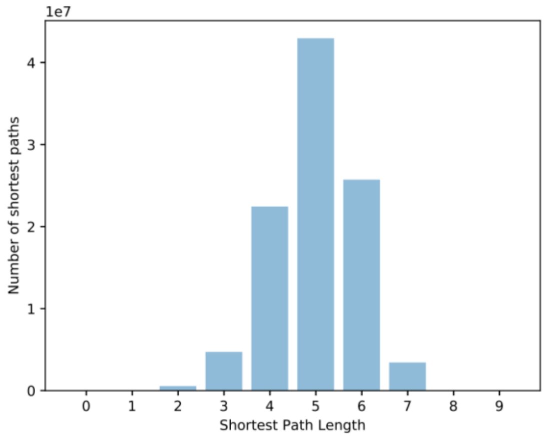

Fig 45. Shortest path distribution for BA5(n=10000,m=2)

3.6 Graph Diameter

In this section the graph diameter for each graph will be given. These values were obtained using NetworkX's diameter(G) function. For brevity, I will refer to a graph's diameter as D.

ER1's D=2, ER2's D=3, ER3's D=3, ER4's D=4, ER5's D=3. WS1's D=19, WS2's D=4, WS3's D=3, WS4's D=8, WS5's D=10. BA1's D=6, BA2's D=5, BA3's D=4, BA4's D=3, BA5's D=9.

4 COMMUNITY DETECTION IN GRAPHS

The detection and analysis of communities in graph networks can be incredibly valuable. The detection of communities is a very computationally heavy process since the recalculation of shortest paths in each iteration is necessary to calculate betweenness scores. Because of this, I used three graphs of 1000 nodes each: ER6(n=1000, p=1/10), WS6(n=1000, k=20, p=1/5), and BA6(n=1000, m=10) to get the ten community sizes in a reasonable amount of time. These communities were found using NetworkX's girvan_newman(G) function.

For ER6, the following array represents the ten community sizes: [991, 1, 1, 1, 1, 1, 1, 1, 1, 1]. I found this peculiar at first, and reran the function using different Erdős–Rényi graphs but obtained the same result. Due to time constrained I was unable to rerun the function on a very large number of graphs. However, it makes sense that there would be a central cluster and several whiskers being detected by the function. Since ER6 is a strictly random graph, it would not make sense that there would be groups of nodes that would cluster together to form a community.

For WS6, the following array represents the ten community sizes: [629, 61, 61, 54, 52, 41, 33, 26, 24, 19]. It makes sense that there is a larger central community as well as several smaller communities. This is due to the nature of the Watts–Strogatz model connecting nodes to neighboring nodes, meaning that communities will inevitably form.

For BA6, the following array represents the ten community sizes: [991, 1, 1, 1, 1, 1, 1, 1, 1, 1]. Similar to ER6, this result makes sense since BA6 is a strictly random model, and clusters of communities do not form.

5 CONCLUSIONS

Many interesting aspects of randomly generated graphs can be extracted to gain a better understanding of the models themselves. Personally, I learned about the various correlations that different random graph models have between the parameters and the results of the clustering coefficients, shortest paths, and communities.