ES 404 - Midsem Examination

Nishant Tatar - 21110223

For Questions 1 to 6, the python code has been uploaded to a Colab Notebook.

1 Network Visualisation

For this, I used cartopy library for plotting the network onto a real world graph. Coordinates were taken from Google's Developer Public Data portal.

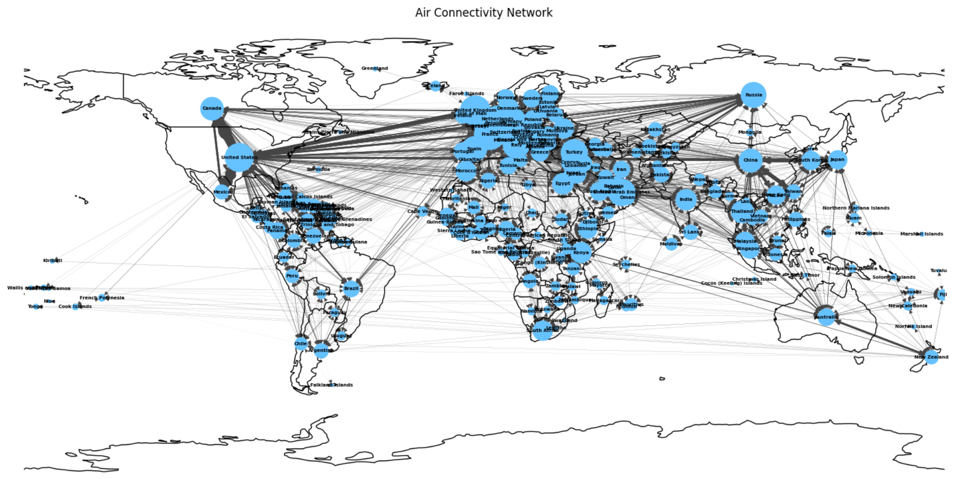


Figure 1: Air Connectivity Network

2 Degree Distibution Plotting and Characterisation

2.1 Degree Distribution plotting

Applying Lograithmic Binning on the degree distribution.

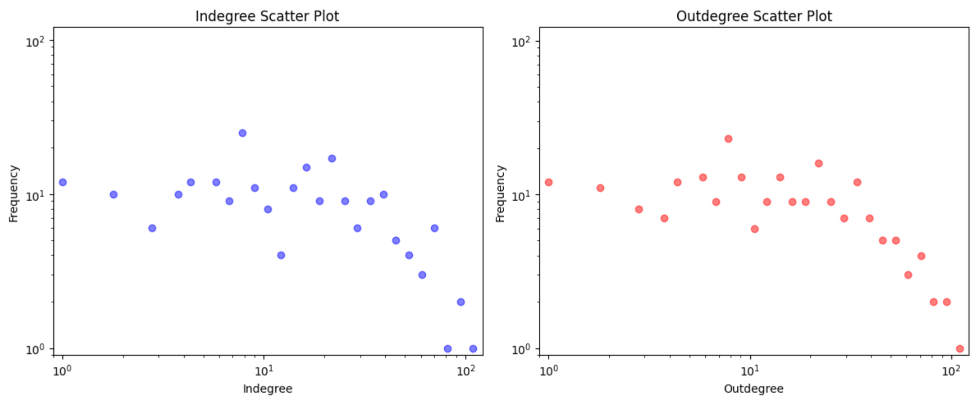


Figure 2: Logarithmic Binning

2.2 Distribution Characterisation

I applied two Goodness of Fit tests on the given degree distribution, KS and Cramer Von Mises test, and then found the p value for the out degree and in degree distributions. I took nearly all

continuous distributions from scipy.stats module, and applied KS and CVM goodness of fit tests. For all distributions with a p value < 0.05, I discarded those distributions.
The following distributions had a p value greater than 0.05 for both KS and CVM tests: fatiguelife, fisk, foldcauchy, genextreme, genpareto, gibrat, gompertz, halfcauchy, halfgennorm, invgamma, invgauss, invweibull, johnsonsb, johnsonsu, kappa3, lognorm, lomax, nct, norminvgauss, pareto, powerlognorm, recipinvgauss, skewcauchy, truncpareto, wald.
While none of these distributions cannot be ruled out, the highest p value for the CVM test was fatiguelife, and as such, the degree distribution is most similar to **fatiguelife** as compared to all other distributions.

3 Pearson Coefficient

The Pearson correlation coefficient, denoted as r, is a measure of the linear correlation between two distributions. It ranges from -1 to 1, where:

- $r = 1$ indicates a perfect positive linear relationship,
- $r = -1$ indicates a perfect negative linear relationship,
- $r = 0$ indicates no linear relationship between the variables.

The formula to calculate the Pearson correlation coefficient between two distributions x and y with n data points is:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Where x_i and y_i are the individual data points, $\bar{x}$ and $\bar{y}$ are the means of the x and y distributions respectively.

3.1 Degree and Betweenness Centrality

The Pearson Coefficient between the Degree and Betweenness Centralities for the network is 0.7368.

3.2 Degree and Closeness Centrality

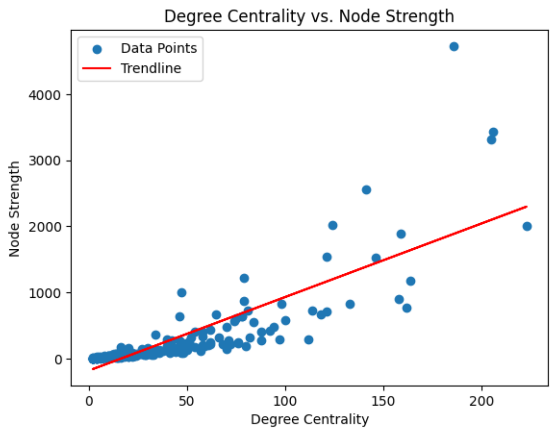
The Pearson Coefficient between the Degree and Closeness Centralities for the network is 0.9006.

3.3 Eigenvector and Degree Centrality

The Pearson Coefficient between the Eigenvector and Degree Centralities for the network is 0.9604.

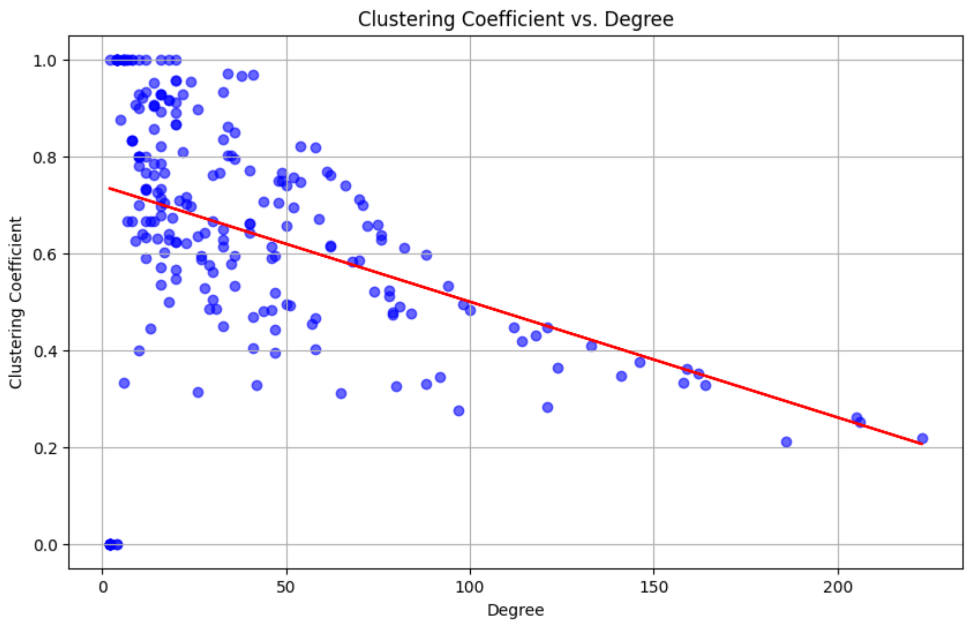
4 Degree Centrality and Strength of Node

The Degree Centrality is the number of nodes that are connected to a particular node, and the Strength of a node refers to the sum of the weights incident on the node.



Linear Correlation Coefficient $(r) = 0.82154$

5 Degree vs Clustering Coefficient



Linear Correlation Coefficient $(r) = -0.383121$

As such, as the value of r is negative, the clustering coefficient of the node goes down.

6 Small World vs Ultra Small World

The average shortest path is calculated using python. The number of nodes in the airport network are 227. Calculating the log and comparing it to the average shortest path, we find that the network is a small world network, as the average shortest path $\approx \log N$

7 Clustering Coefficients BA and ER Networks

The Local Clustering Coefficient is a measure of how close the neighbors of a node are to forming a complete clique. This is a measure of network connectivity, and is defined as $C_i = \frac{2\times(T_i)}{(k_i)\times(k_i-1)}$. Where T_i is the number of triangles passing through the node i and k_i is the degree of node i. Another Way to look at this could be as the ratio of realised connections compared to number of possible connections.

7.1 Barabasi Albert Model

For the Barabasi Albert Model, where one node appears one at a time with m edges ready to connect to the network for a particular node, it connects to existing nodes in the network with a probability proportional to their degrees.
At time when node i arrived, we assume that node j was already present.

$$k_j(i) = \left(\frac{i}{j}\right)^{1/2}$$

The probability that i would connect with j is:

$$P_{i,j} = \frac{m \times k_j(i)}{2 \times m \times i} = \frac{m}{2\sqrt{ij}}$$

Similarly for $P_{j,l}$ and $P_{l,i}$, we get $\frac{m}{2\sqrt{jl}}$ and $\frac{m}{2\sqrt{li}}$.
Hence, the Local Clustering Coefficient is:

$$C_l = \frac{2\int_{i=l,i>j}^{N-1} di \int_{j=1}^{N-1} P_{i,j}P_{j,l}P_{l,i}\, dj}{(k_l)\cdot(k_l-1)}$$

The Entire Integrand can be shown as:

$$2\int_{i=l,i>j}^{N-1} di \int_{j=1}^{N-1} P_{i,j}P_{j,l}P_{l,i}\, dj = \frac{m^3}{8}\int\int\left[\frac{1}{\sqrt{ij}}\cdot\frac{1}{jl}\cdot\frac{1}{li}\right] didj$$

$$= \frac{m^3}{8l}\int\int\left(\frac{1}{i}\cdot\frac{1}{j}\right) didj$$

$$= \frac{m^3}{8l}\left|\ln i\right|_{i=1,i>j}^{N-1}\left|\ln j\right|_{j=1}^{N-1}$$

$$\approx \frac{m^3}{8l}(\ln N)^2$$

Therefore, C_l is:

$$C_l = \frac{2\cdot\frac{m^3}{8}(\ln N)^2}{(k_l)\cdot(k_l-1)}$$

and

$$k_l = m\cdot\left(\frac{N}{l}\right)^{1/2}$$

For sufficiently large N, the equation reduces to:

$$C_l = \frac{m}{4}\cdot\frac{(\ln N)^2}{N}$$

7.2 Erdos Renyi Model

We define C_i as $\frac{2\cdot T_i}{(k_i)\cdot(k_i-1)}$, where T_i is the number of triangles passing through node i, or, the expected number of edges between the neighbors of the node. In an ER network with N nodes and edge probability p, the expected number of edges connected to node i, denoted as k_i, follows a binomial distribution with parameters $N-1$ and p. So, $k_i = (N-1)p$.

The expected value of T_i is determined as follows:

Since edges between pairs of nodes are independent, the expected number of edges between the neighbors of node i is the probability that a pair of neighbors is connected, multiplied by the total number of possible pairs of neighbors. Each neighbor of node i has k_i possible connections, and the probability that any particular pair of them is connected is p.
As such, expected number of edges for node i is:

$$T_i = \frac{k_i(k_i - 1)}{2} \times p$$

Plugging this back into the equation for C_i:

$$C_i = \frac{2 \times \frac{k_i(k_i-1)}{2} \times p}{k_i(k_i - 1)}$$
$$C_i = p$$

8 Emergence of Scalefree nature in BA Model

The Barabási–Albert (BA) model is characterized by a power-law distribution of node degrees, where the degree distribution follows a power-law function. Two essential requirements for the emergence of scale-free properties in the BA model are:

1. **Preferential Attachment:**

   The first essential requirement is the principle of preferential attachment, where new nodes are more likely to connect to existing nodes that already have a high degree.

   Mathematically, the probability Π that a new node attaches to an existing node i with degree k_i is formulated as:
   $$\Pi(k_i) = \frac{k_i}{\sum_j k_j}$$
   where k_i represents the degree of node i, and $\sum_j k_j$ represents the sum of degrees of all nodes in the network. Here, a node that has higher degree is much more likely to have a node connect to it.

2. **Network Growth:**

   The second essential requirement is that the network must grow. In the BA model, at each time step, a new node is added to the network, and it forms connections with existing nodes according to the preferential attachment rule. This continuous growth process, combined with preferential attachment, leads to the emergence of scale-free properties in the network, where the degree distribution follows a power-law.

In summary, the combination of preferential attachment and continuous growth in the BA model drives the emergence of scale-free properties in complex networks. Preferential attachment biases the attachment of new nodes towards high-degree nodes, while the growth mechanism ensures the continuous expansion of the network, resulting in a power-law distribution of node degrees.

References

[1] "Geopy." https://pypi.org/project/geopy/. Accessed: 2024-2-26.

[2] "NetworkX get degree of node with weights applied." https://stackoverflow.com/questions/27878034/networkx-get-degree-of-node-with-weights-applied. Accessed: 2024-2-26.

[3] "Software for complex networks — NetworkX 3.2.1 documentation." https://networkx.org/documentation/stable/index.html. Accessed: 2024-2-26.

[4] "SciPy documentation — SciPy v1.12.0 manual." https://docs.scipy.org/doc/scipy/index.html. Accessed: 2024-2-26.

[5] "Countries.Csv." https://developers.google.com/public-data/docs/canonical/countries_csv. Accessed: 2024-2-26.

[6] "NumPy documentation — NumPy v1.26 manual." https://numpy.org/doc/stable/index.html. Accessed: 2024-2-26.

[7] "ChatGPT." https://chat.openai.com/. Accessed: 2024-2-26.

[8] "10 minutes to pandas — pandas 2.2.1 documentation." https://pandas.pydata.org/docs/user_guide/10min.html. Accessed: 2024-2-26.