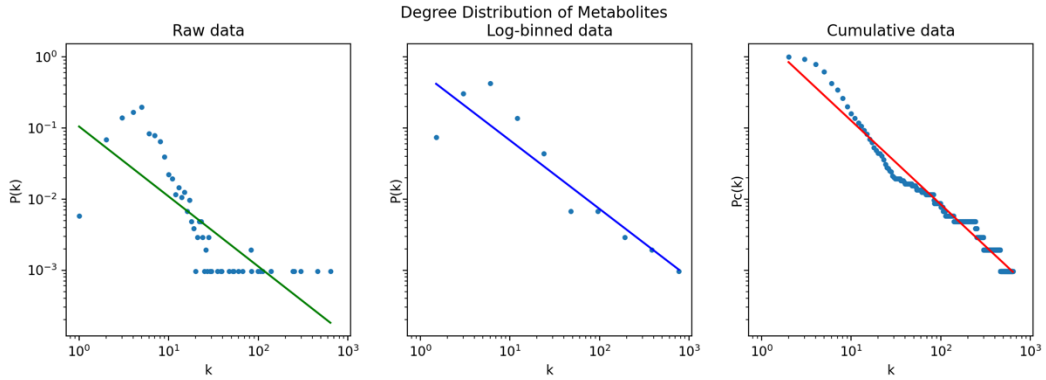
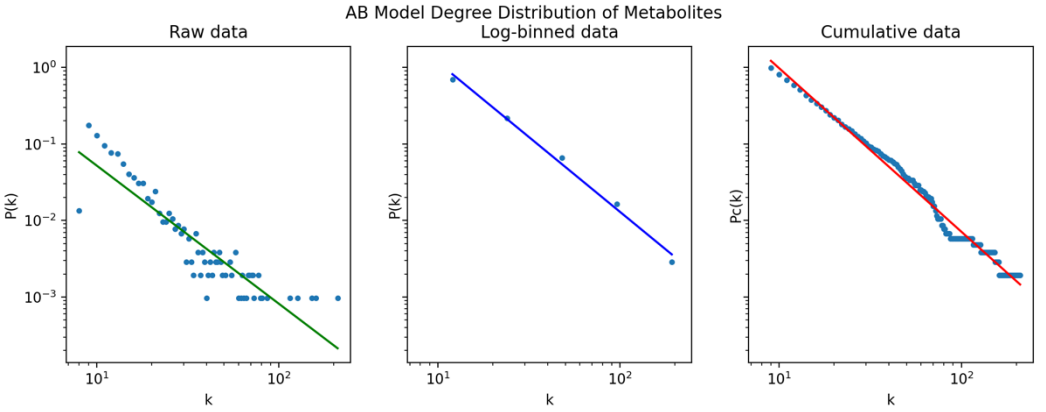
COMP 596 Assignment 1

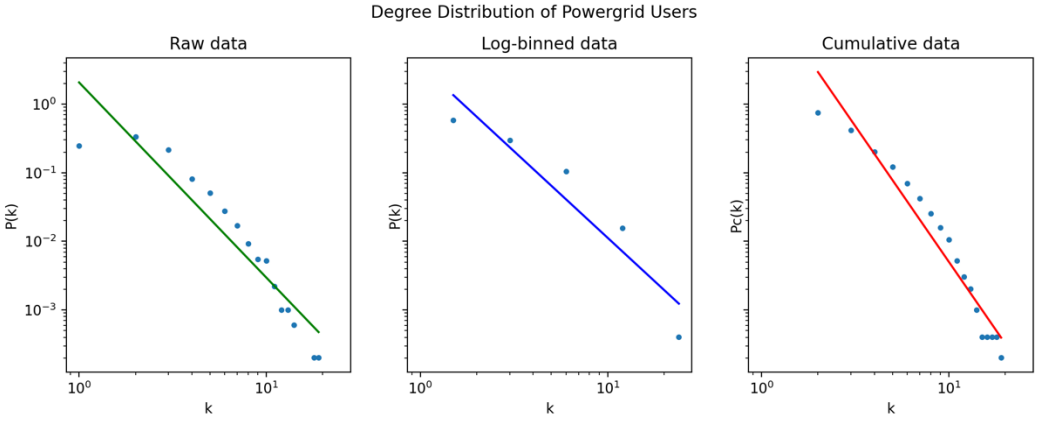
Jessica Chan
Sep. 20, 2020
260734339
jessica.chan5@mail.mcgill.ca

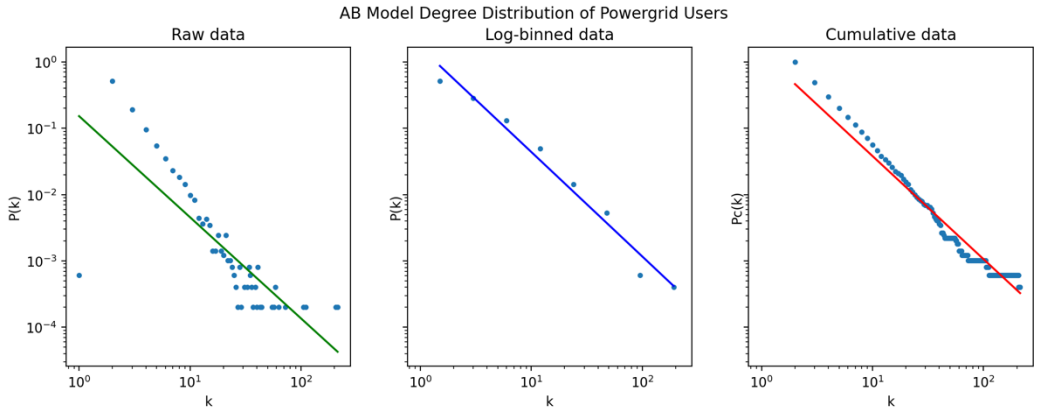
Degree Distribution $O(|V|^2)$

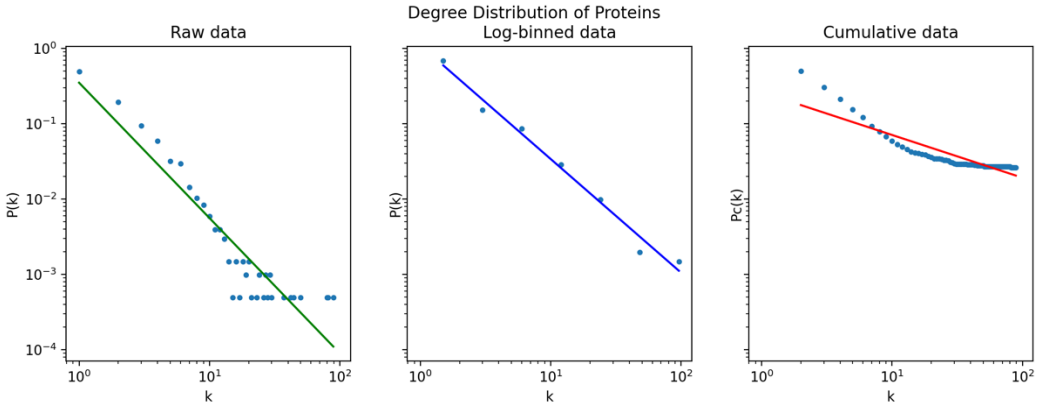
Used np.polyfit to get best fit coefficients for power law

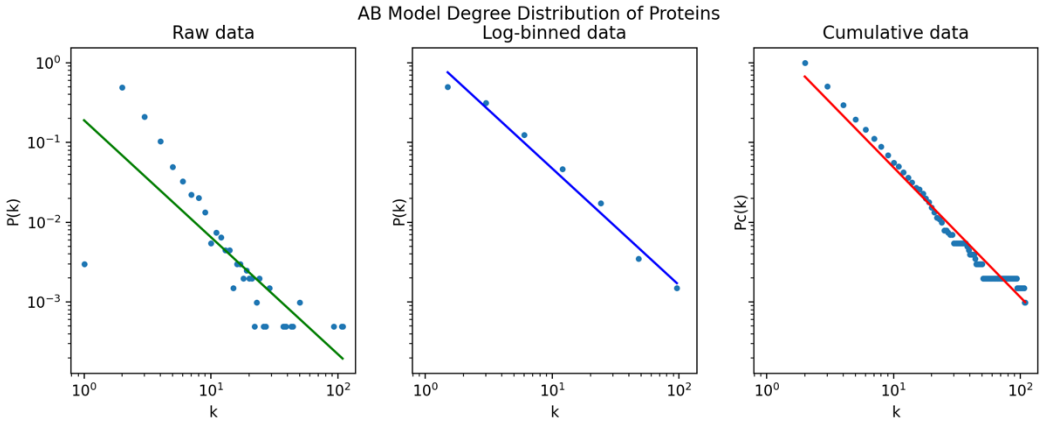




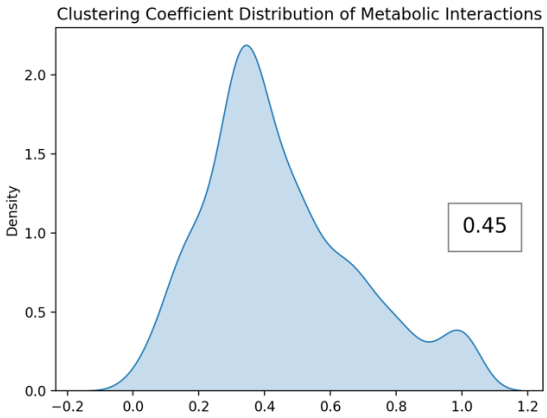


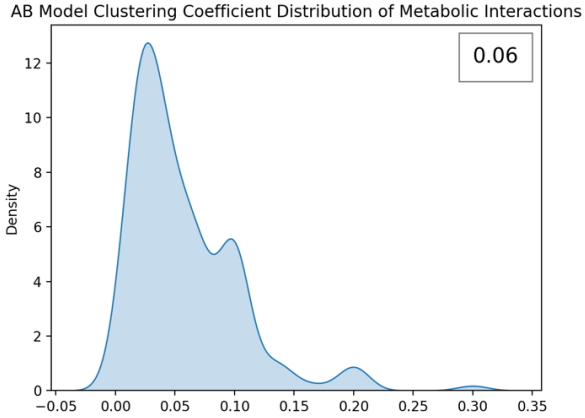

AB Model Degree Distribution of Powergrid Users
Raw data
Log-binned data
Cumulative data
P(k)
Pc(k)
k

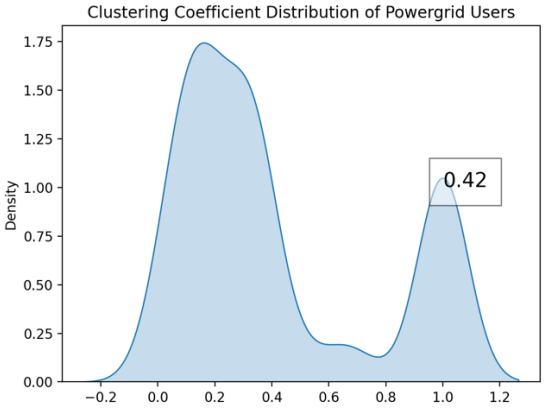

Degree Distribution of Proteins
Raw data
Log-binned data
Cumulative data
P(k)
Pc(k)
k

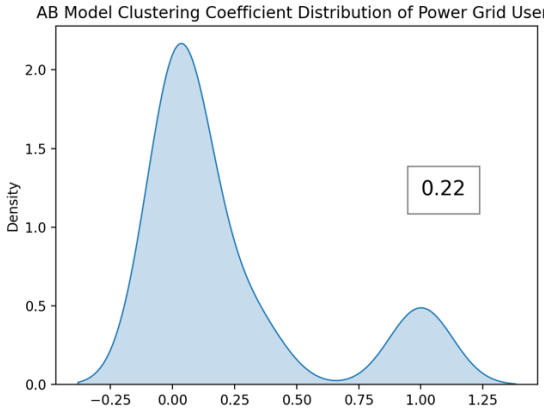

AB Model Degree Distribution of Proteins
Raw data
Log-binned data
Cumulative data
P(k)
Pc(k)
k

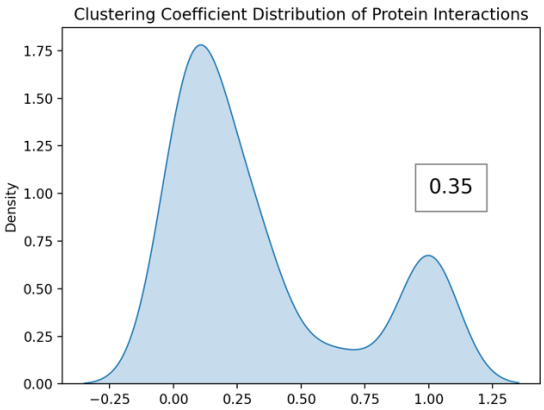
Clustering Coefficient Distribution $O(|V|^3)$

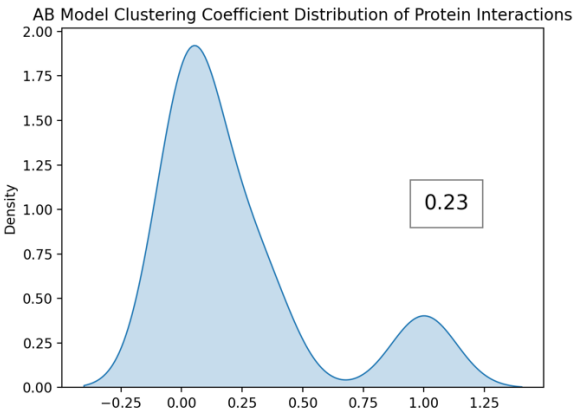






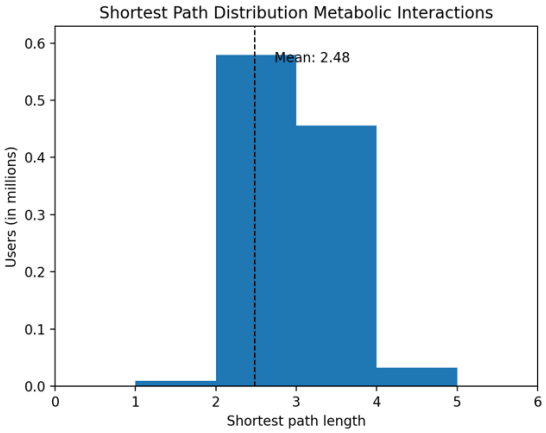


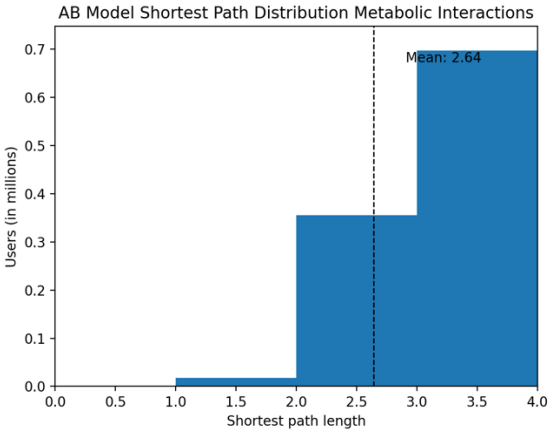


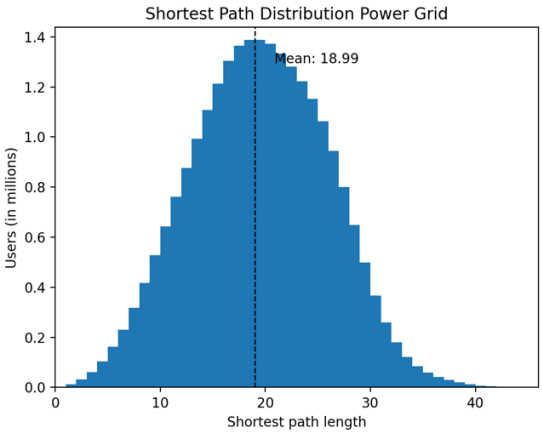


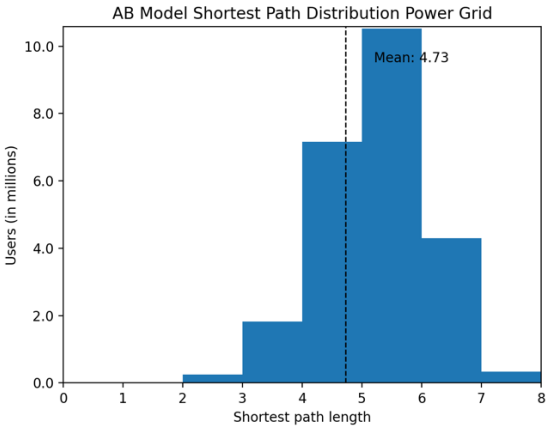
Shortest Paths Distribution

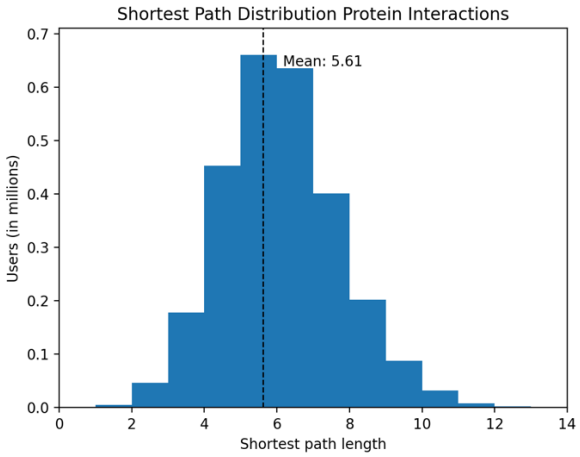
O(|V| + |E|) *(Dijkstra's with Fibonacci heaps)*

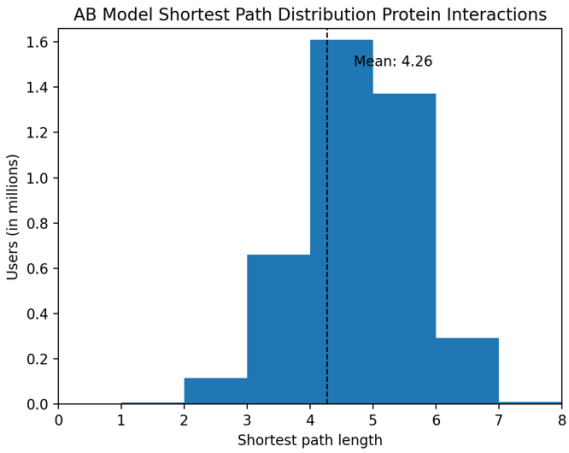






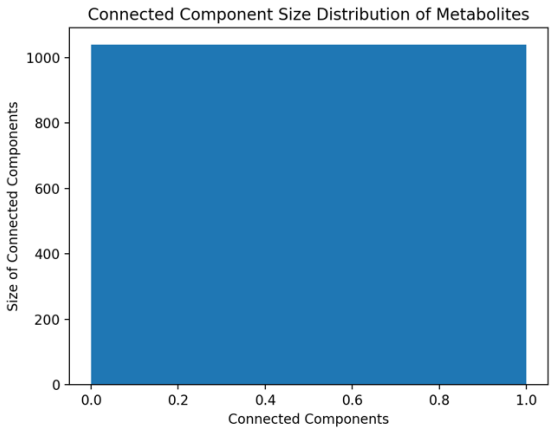


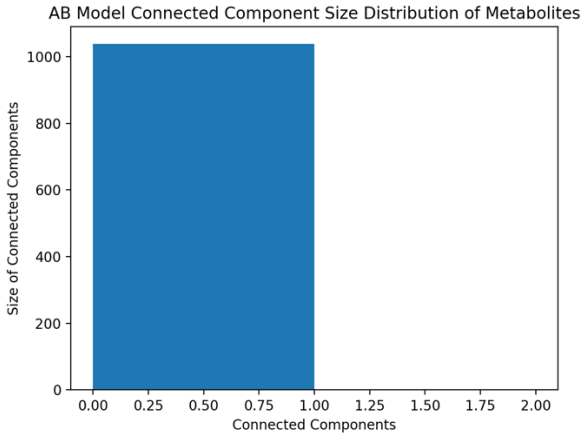




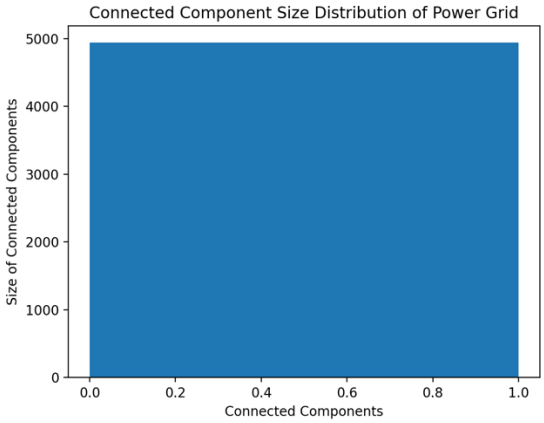
Connected Components O(|V| + |E|) *(DFS from unvisited nodes)*

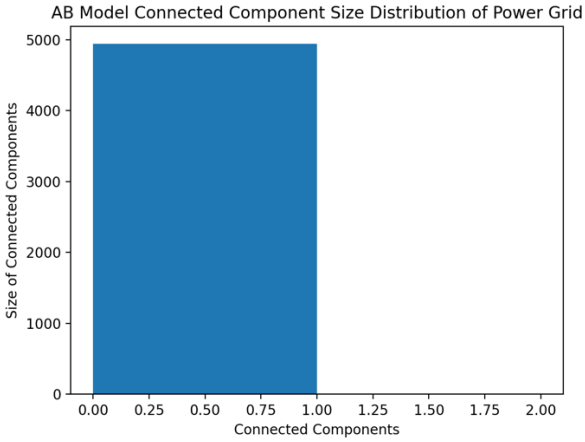
Used connected_components from scipy.sparse.csgraphs



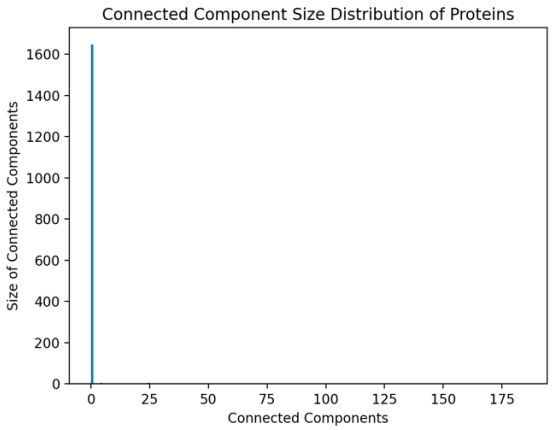


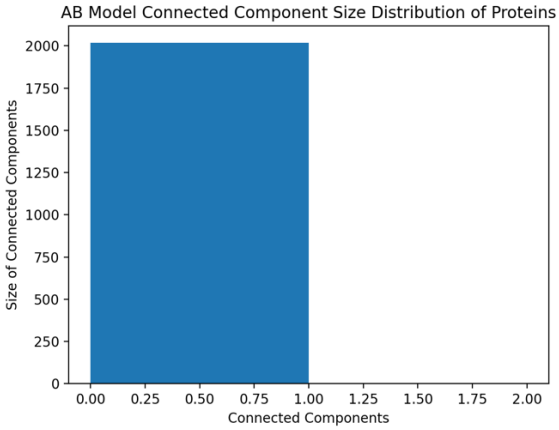
Nodes in GCC: 1039/1039 *1038/1039*





Nodes in GCC: 4941/4941 *4940/4941*

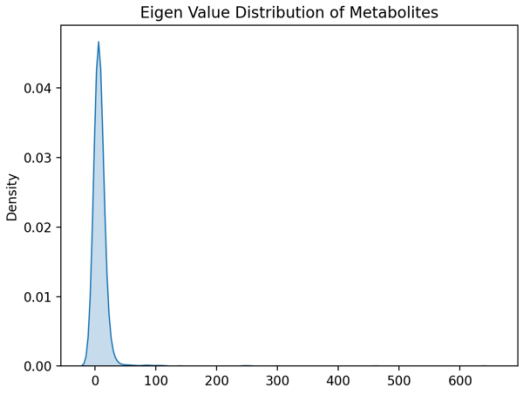


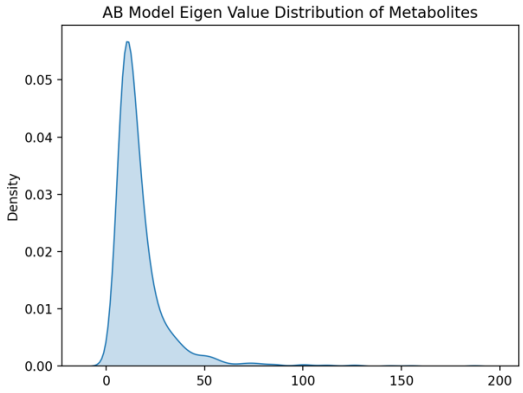


Nodes in GCC: 1647/2018 *2017/2018*

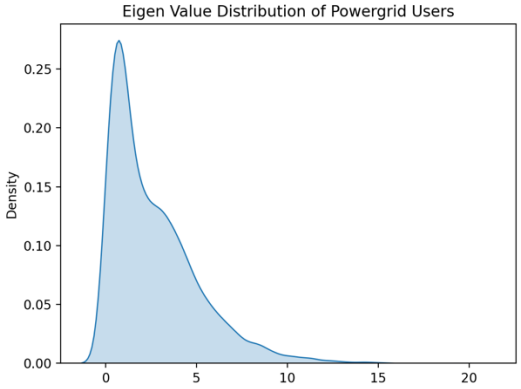
Eigen Value Distribution $O(|N|^3)$

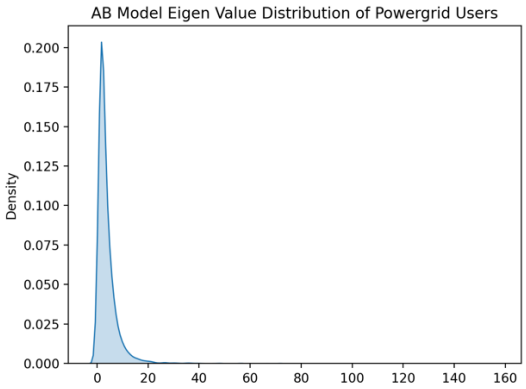
Used np.linalg.eigh to get all eigen values



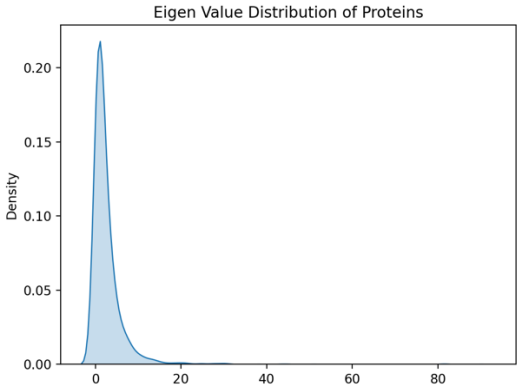


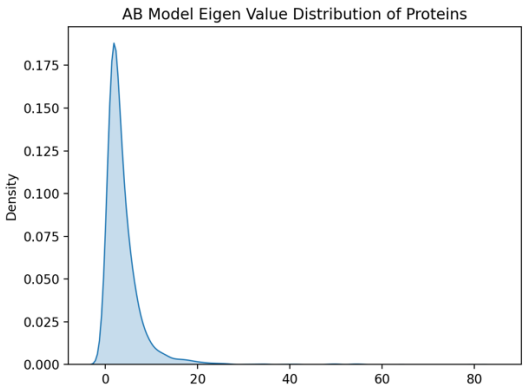
Spectral gaps: 0.535242521634216 3.6053812001996244





Spectral gaps: 0.0007592122113561372 0.5157255200674538

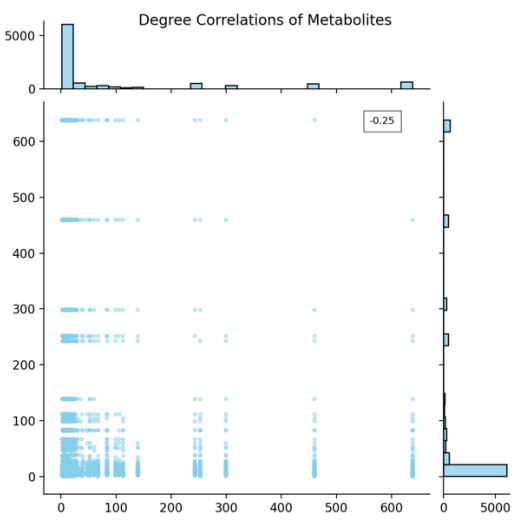


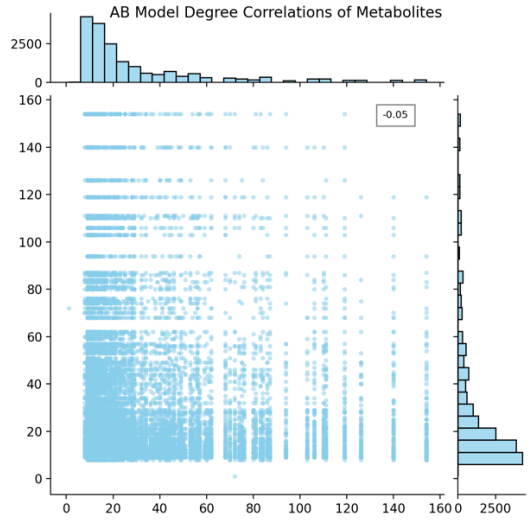


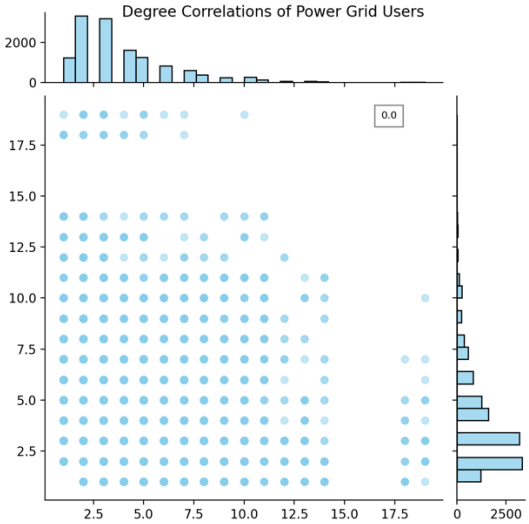
Spectral gaps: 0.041823143729712 0.5095942692760469

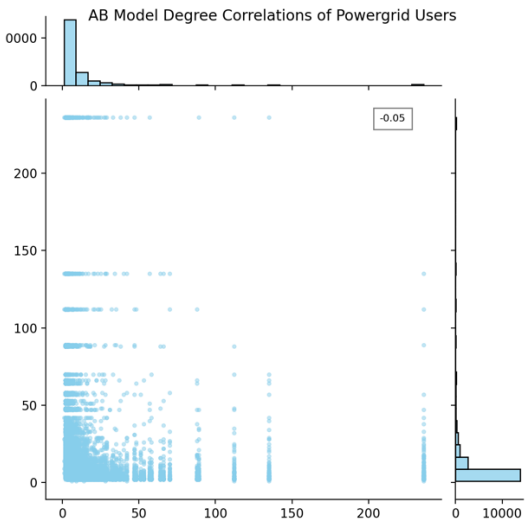
Degree Correlations O(|V|) *(Sum of edges for each V)*

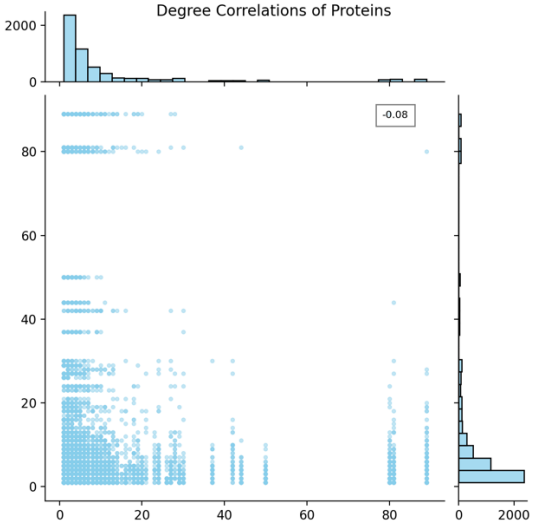
Used Pearsonr correlation from scipy.stats to get overall correlation

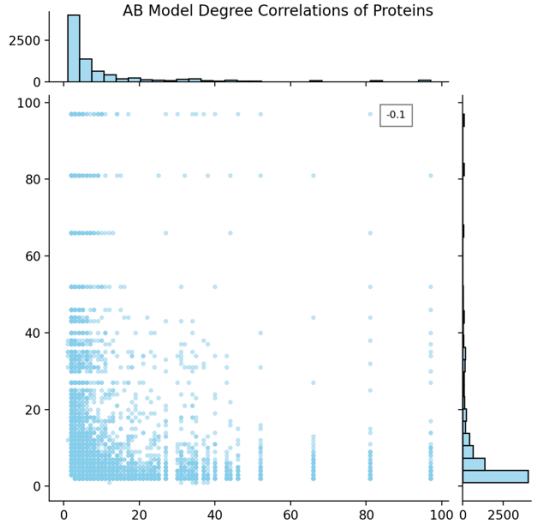






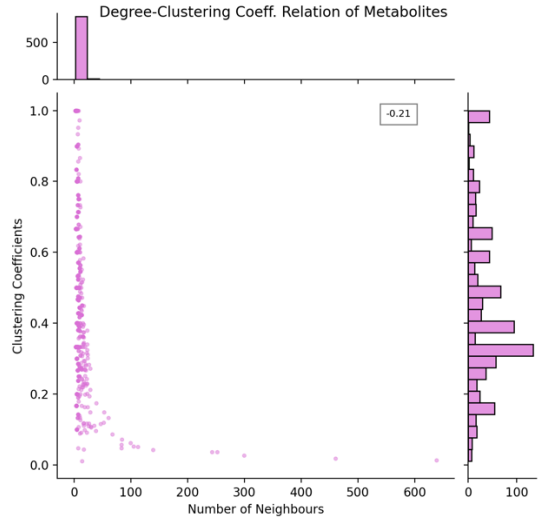


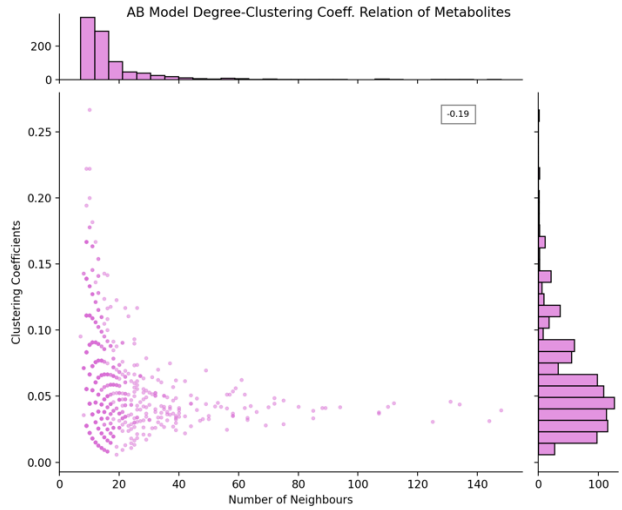


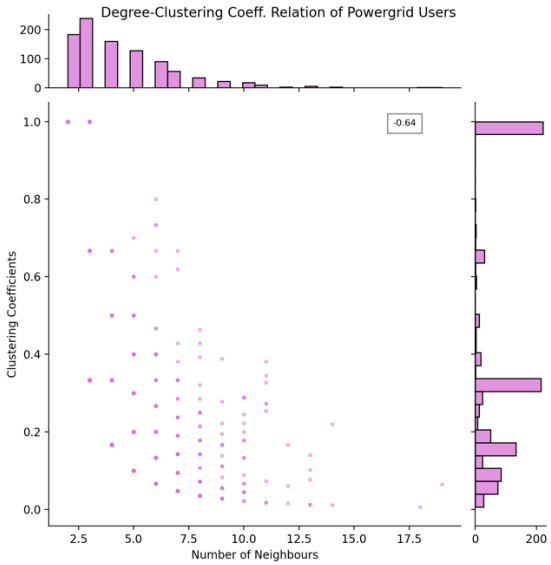


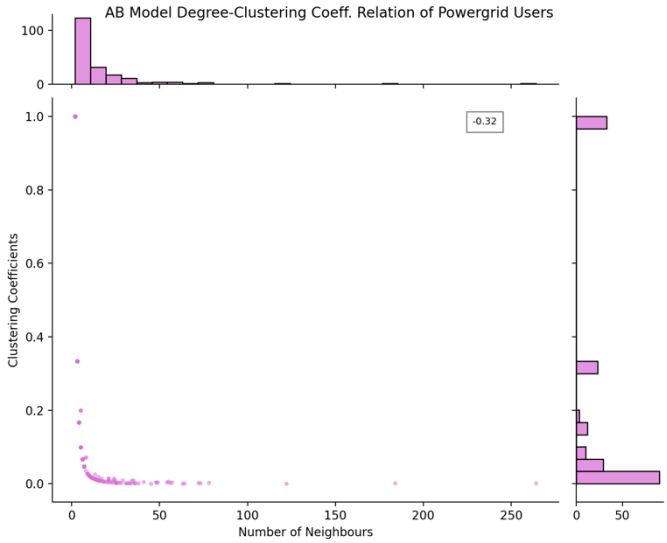
Degree-Clustering Coefficient Distribution $O(|V|^3)$ *(O(V) degrees + $O(V^3)$ cc)*

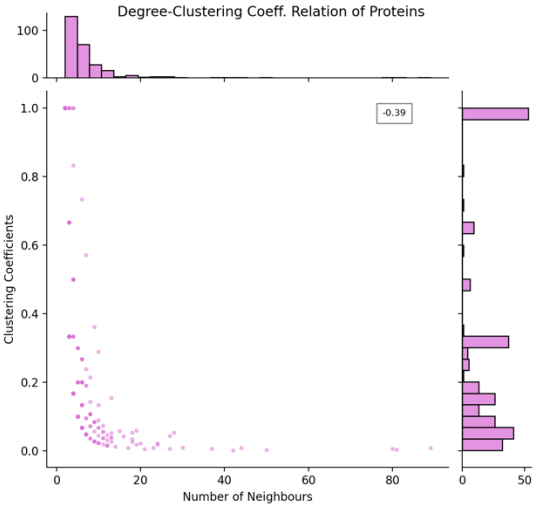
Used Pearsonr correlation from scipy.stats to get overall correlation

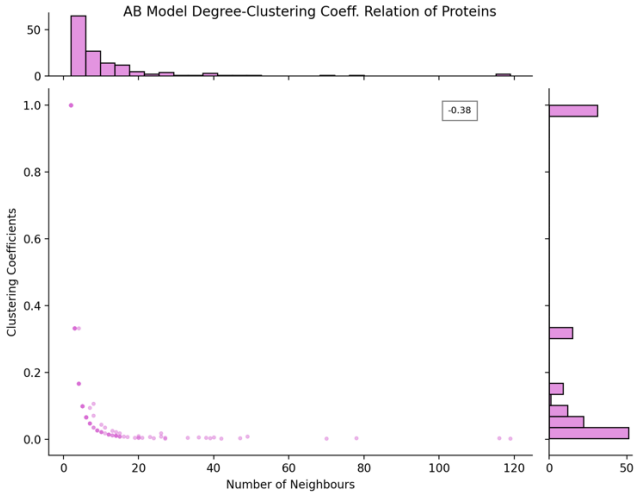












AB Model

The AB model implementation used is a hybrid of the LCD (Linearized Chord Diagram) model of AB outlined in Barabasi's text and the original AB calculation. Probabilities are calculated as below:

$$p(i) = \begin{cases} \frac{1}{\sum_j d_j + 1}, & if\ d_i = 0 \\ \frac{d_i}{\sum_j d_j + 1}, & otherwise \end{cases}$$

In this model, self-loops are considered with a small probability when node i is being added to allow for the possibility of disconnected components. The model used borrows the first time step of LCD, creating a self loop for the first node added to the graph. All self-loops are removed at the end to maintain simple graph properties. np.random was used to randomly select the vertex to link to.

In the above plots, AB_LCD_hybrid_model(m, n, A) takes 3 inputs:

m = average degree of nodes in A
n = number of nodes in A
A = original network adjacency matrix

The values for each network are as follows:

Metabolic network:	m = 2	n = 4941
Power grid network:	m = 9	n = 1039
Protein network:	m = 2	n = 2018

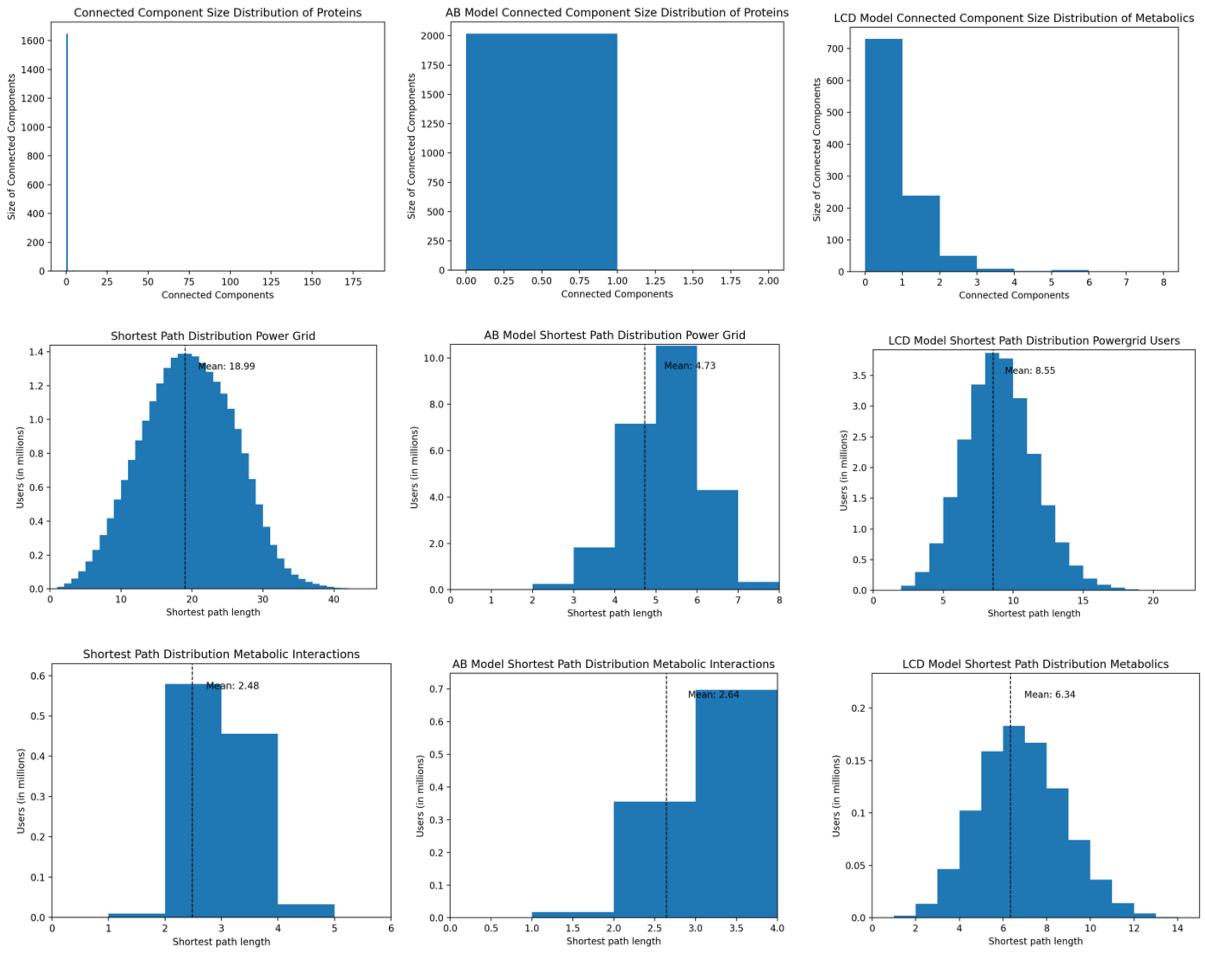
In the code, m edges are added per node where possible, where m is the average degree from the original network. I've found that lower values of m result in more similarly behaving networks. Higher values of m increase the average degree of the resulting AB model network, and causes the AB network to behave differently from the original. Specifically, AB model networks with high average degree seem to have far more edges than the original network.

In the code, one can also find the LCD model that adds one node and edge at a time, calculated with the use of the time step t:

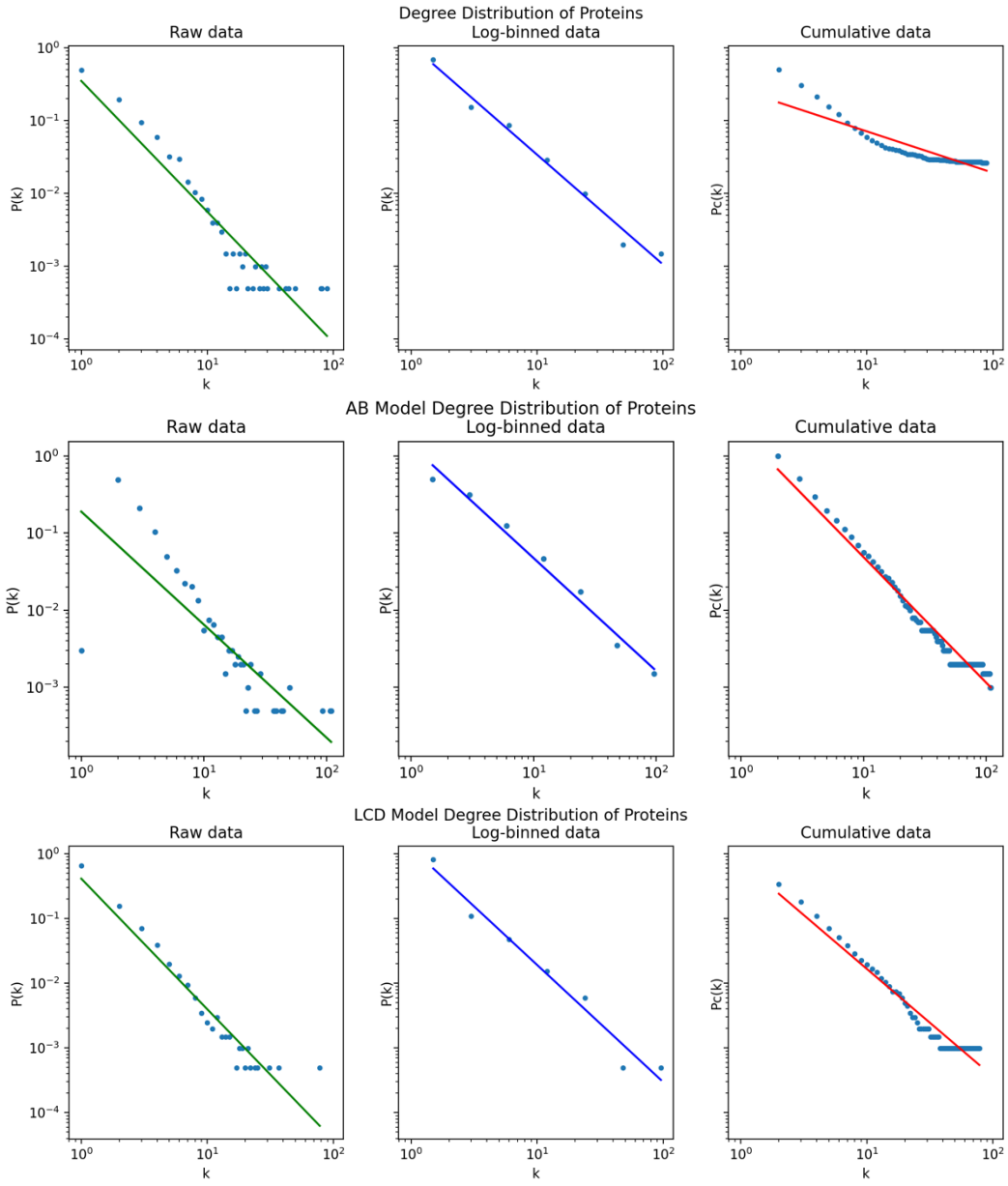
$$p(i) = \begin{cases} \frac{d_i}{2t-1}, & if\ \ 1 \leq i \leq t-1 \\ \frac{1}{2t-1}, & if\ i = t. \end{cases}$$

The issue with using the LCD probability for adding multiple edges per node is that the time step does not increase as the degrees increase. LCD expects that one node per edge is added at each time step. Violating this results in a probability sum greater than 1.

I've included some LCD network distributions that are of note below.



Here we can see that the LCD algorithm achieves a more accurate representation of connected components in proteins, a network with a low average degree (m=2) and high number of nodes (n=2018). The LCD model also seems to achieve a more accurate representation of shortest paths distributions in networks with higher average degree (eg. Power grid m=9). The AB hybrid model performs better for networks with lower average degree (eg. metabolic m=2).



In future I would like to try the LCD algorithm with a probability function and time step that changes with the number of edges added, rather than the number of nodes added. This way it could be possible to limit the amount of edges created in the graph, though some problems could arise with high values of m in sparsely connected networks. I also experimented with randomizing the selection of vertices to add, though this seems to have little to no effect on the data.