Analyzing Networks

Byron Fong

Introduction

This investigation aims to test three graph algorithms to generalize properties of models of real-world networks. The three graph algorithms calculate 1) Diameter, 2) Clustering coefficient, 3) Degree Distribution. Diameter is defined as the length of the longest path in a graph. Clustering coefficient is defined as the ratio of three times the number of triangles over the number of paths of length 2. Degree distribution is the number of vertices in the graph with that degree for each possible degree.

Procedure

The implementation of the three graph algorithms are as follows:

def get_diameter -> int: *

Select a random vertex **r** in the graph, set Dmax = 0
Perform a BFS from the random vertex
Select the farthest node (one if multiple) that is returned, **w**
If the distance from **r** to **w** is larger than Dmax, set Dmax to this distance, let **r=w** and repeat the last two steps
Return Dmax

def get_clustering_coefficient -> float:

Compute a degeneracy ordering, **L,** by repeatedly finding and removing the vertex of smallest degree then adding it to the end of the list
While computing the degeneracy ordering, store a list $\mathbf{N_v}$ that stores lists of neighbors of a vertex **v** that come before **v** in L by checking if neighbor **w** is in L already
For each vertex **v** in the order of **L:**
For each pair of vertices **u**, **w** adjacent to **v** and earlier in the ordering (determined by if **u** and **w** are in the list $\mathbf{N_v}$):
If **u** and **w** form an edge in the graph add one to the triangle count
To compute number of 2-edge paths, take the sum of degree(**v**)*degree(**v**-1)/2 for all vertices **v** in the graph
Return 3 * triangle count / number of 2-edge paths

def get_degree_distribution -> HashMap[int: int]:

For each vertex **v** in the graph, compute its degree
If its degree is not in the hash map, add its degree to the hash map with a value of zero.
Increment the value of degree(**v**) by 1 in the hash map and return the hash map when all vertices are processed.

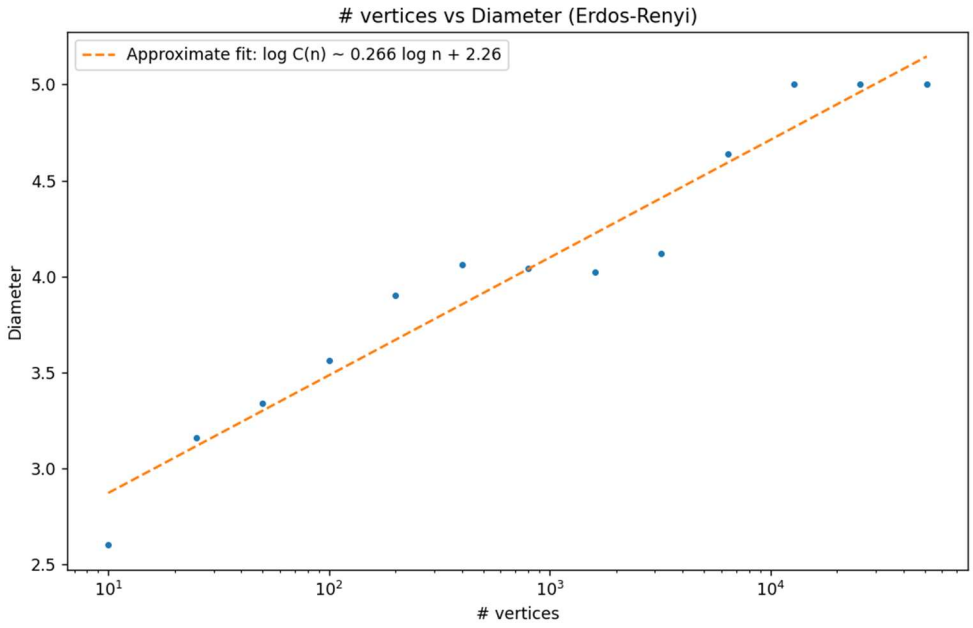
*Note: the function used to find diameter may be inaccurate since it uses a heuristic function that may fail on some cases in exchange for considerably faster time performance.

These algorithms were tested on two types of graphs: Erdos-Renyi random graphs with probability p=2(ln n)/n, and Barabasi-Albert random graphs with parameter d = 5 for number of neighbors each new vertex chooses. 50 trials of the algorithms were tested to reduce noise. Note that all graphs are also connected so diameter cannot be infinity.

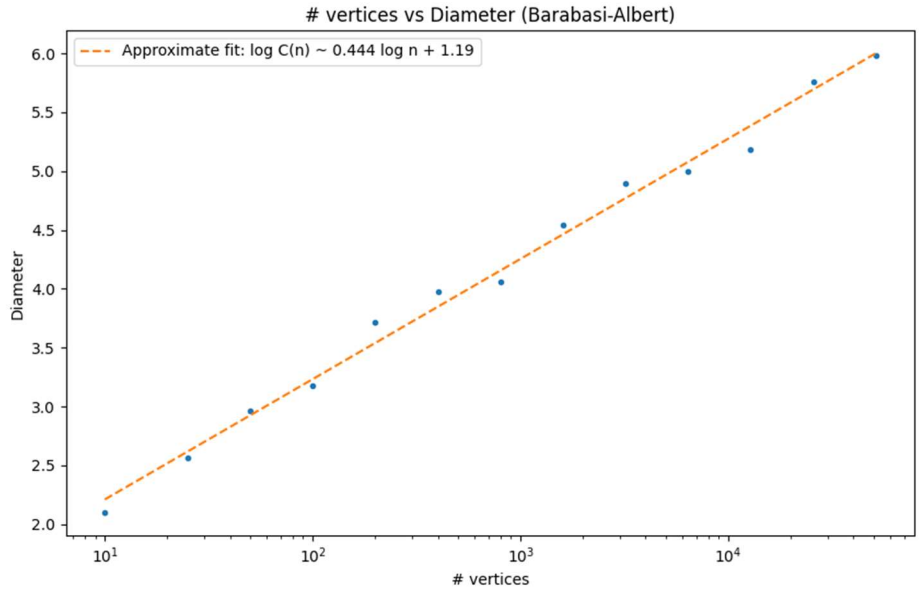
The diameter and clustering coefficient algorithms were tested on random graphs with increasing size n, ranging from n=10 to n=51200. The degree distribution algorithm was tested on both Erdos-Renyi random graphs and Barabasi-Albert random graphs with sizes n=1,000, n=10,000, and n=100,000.

Findings

Diameter

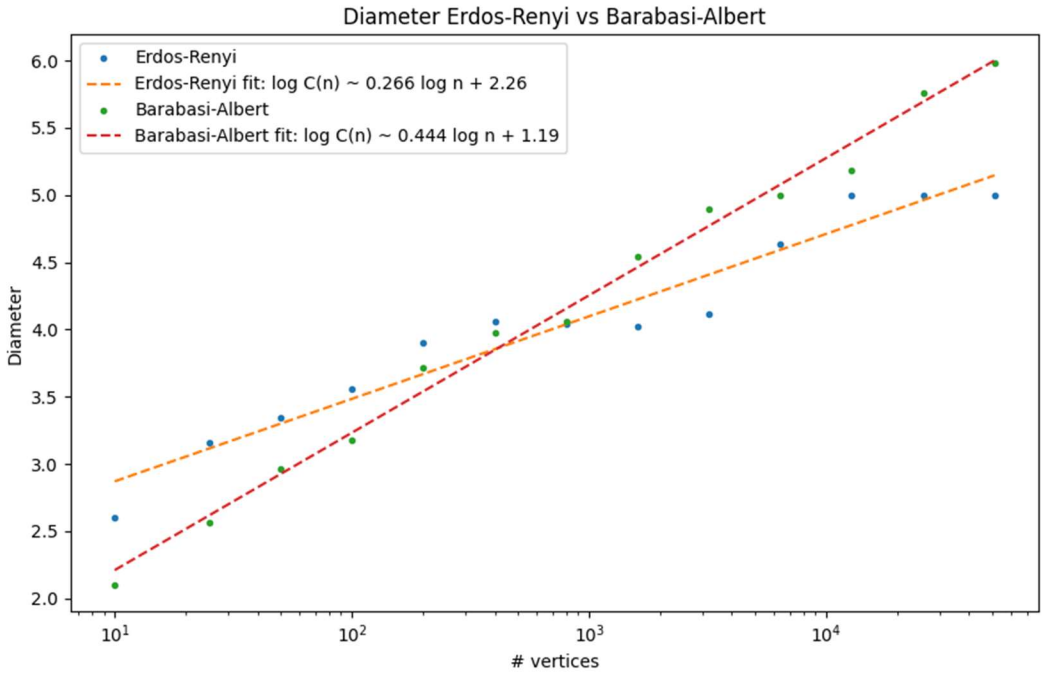


The diameter grows as a function of n on andomly generated Barabasi-Albert graphs. Diameter grows proportional to the function log n and is given by the linear fit log C(n) ~ 0.266 log n + 2.26. This linear fit is inaccurate for small values of n as diameter of any graph is 0 when there are no vertices.

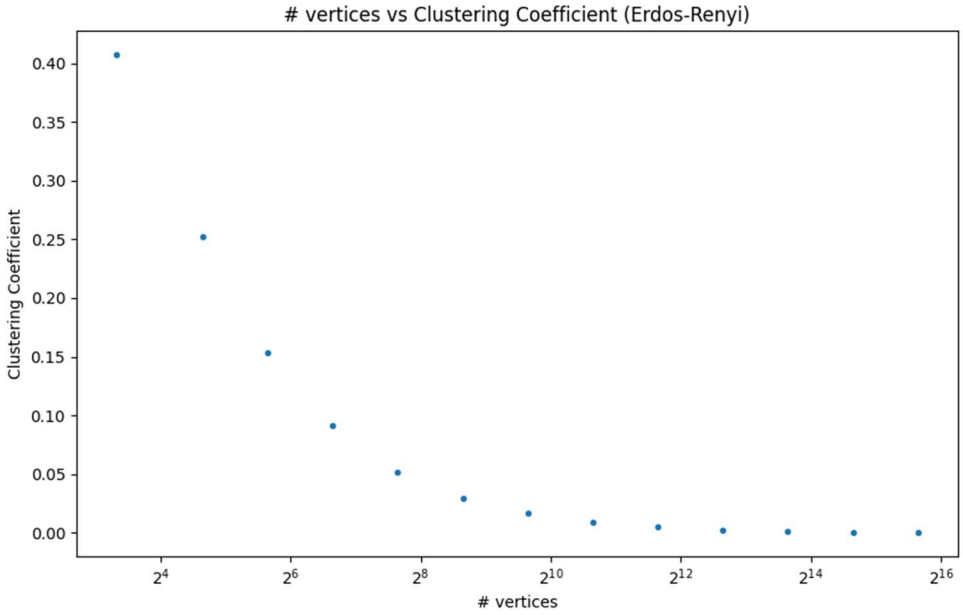


The diameter grows as a function of n on randomly generated Barabasi-Albert graphs. Diameter grows proportional to the function log n and is given by the approximate linear fit log C(n) ~ 0.444 log n + 1.19. Similarly to the linear fit for Erdos-Renyi graphs, this linear fit is inaccurate for small n. The diameter of randomly generated Barabasi-Albert graphs grows faster as a function of n than the diameter of randomly generated Erdos-Renyi graphs.

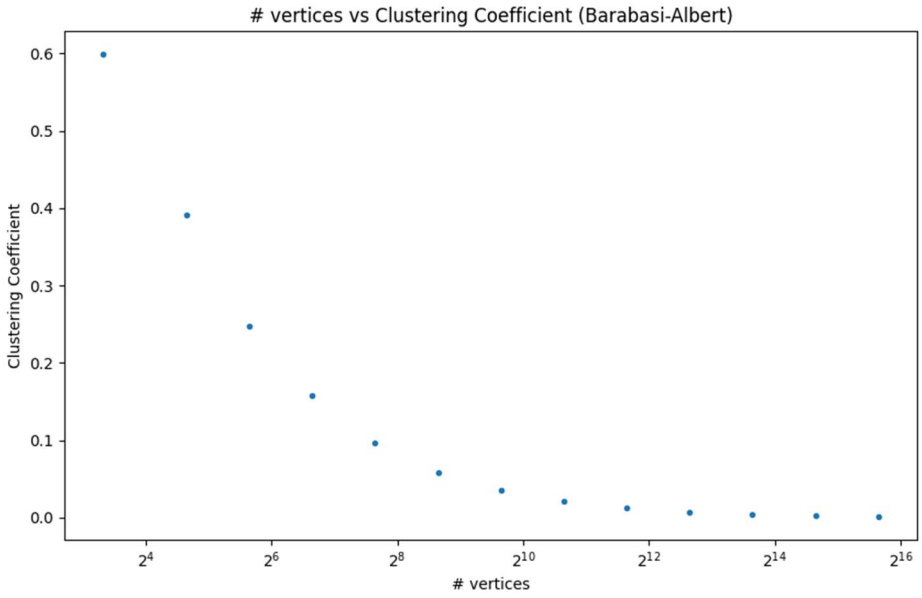
Combined graph for comparison



Clustering Coefficient

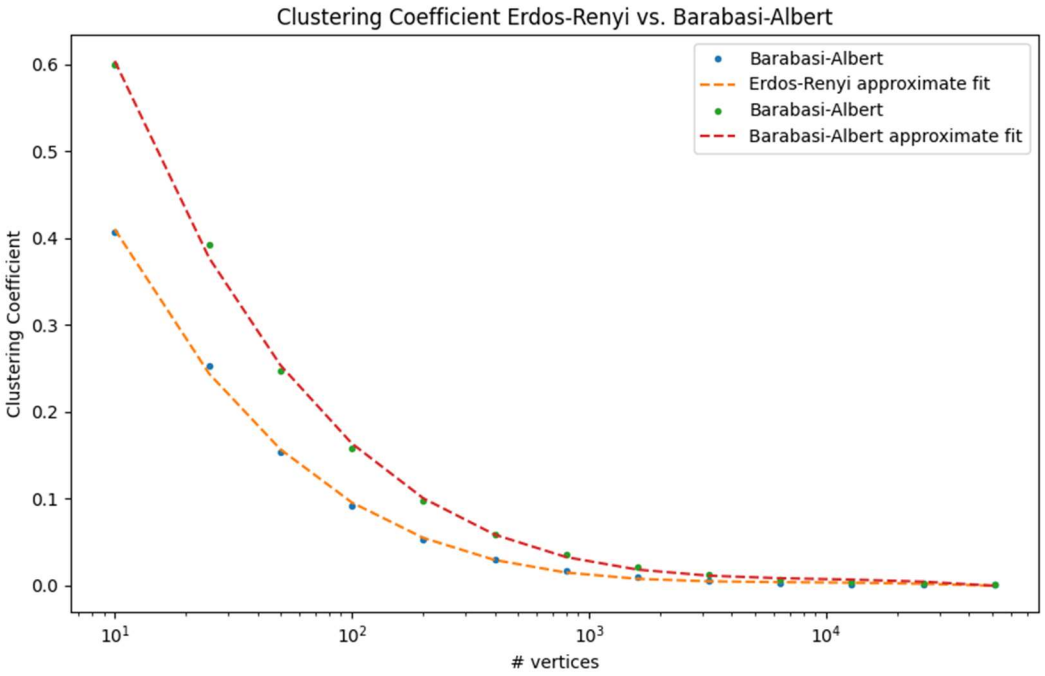


Clustering coefficient decreases as a function of n on randomly generated Erdos-Renyi graphs. The clustering coefficient does not decrease proportional to log n, and instead decreases faster, following the growth rate of exponential decay.



Clustering coefficient decreases as a function of n on randomly generated Barabasi-Albert graphs, similar to Erdos-Renyi graphs. It also follows a growth rate of exponential decay, but decays faster than Erdos-Renyi graphs.

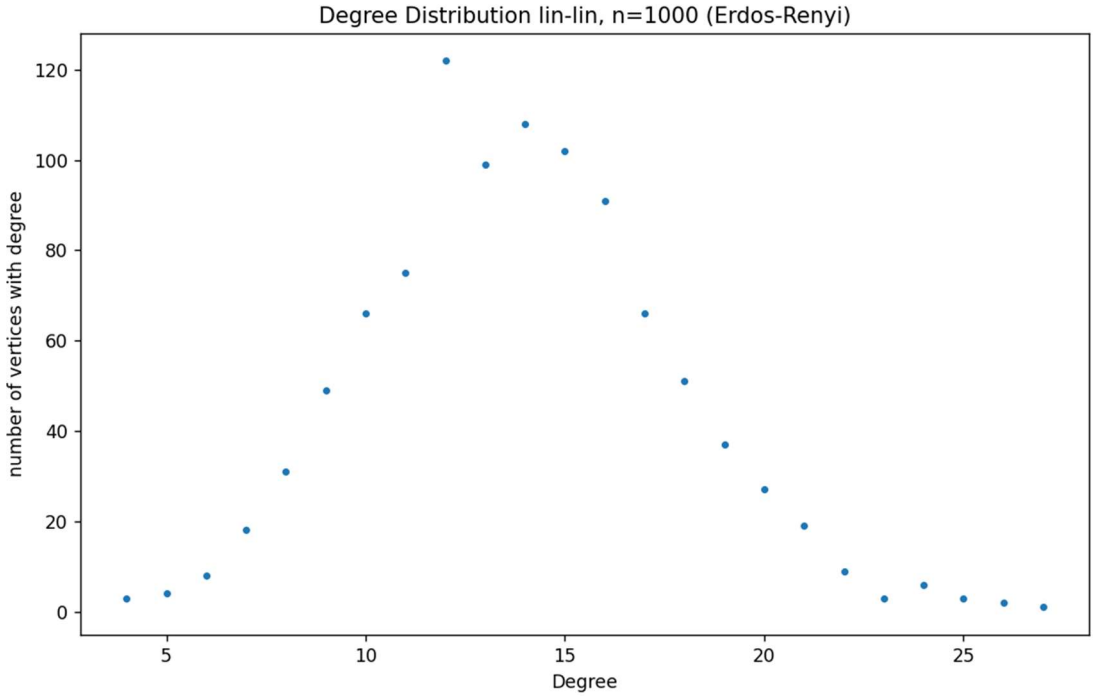
Combined graph for comparison

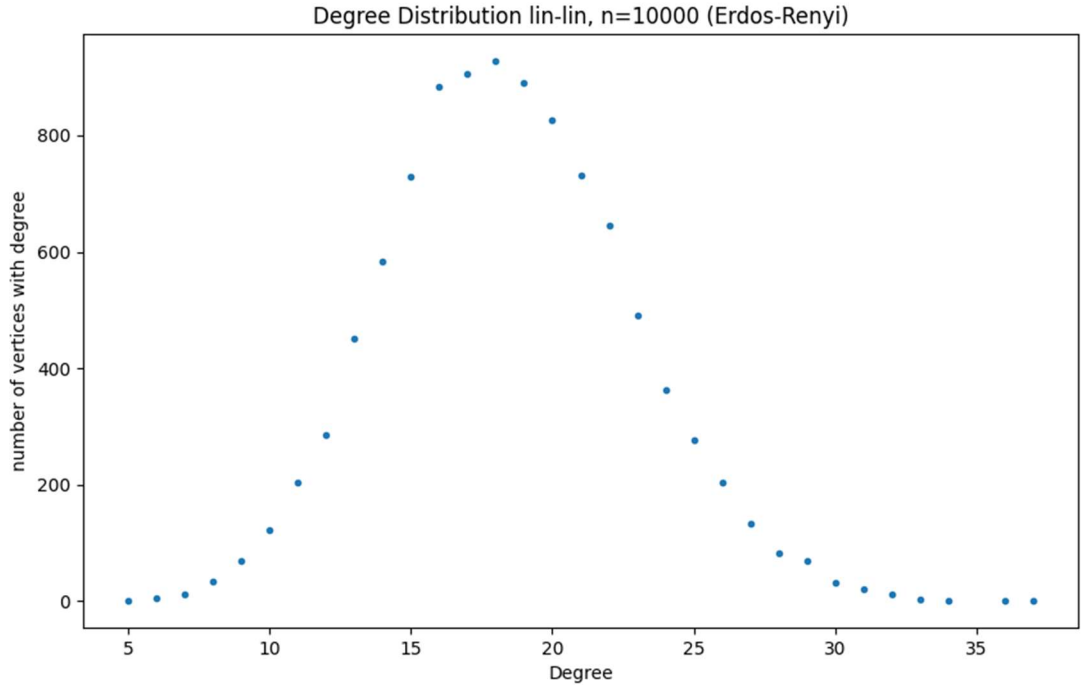


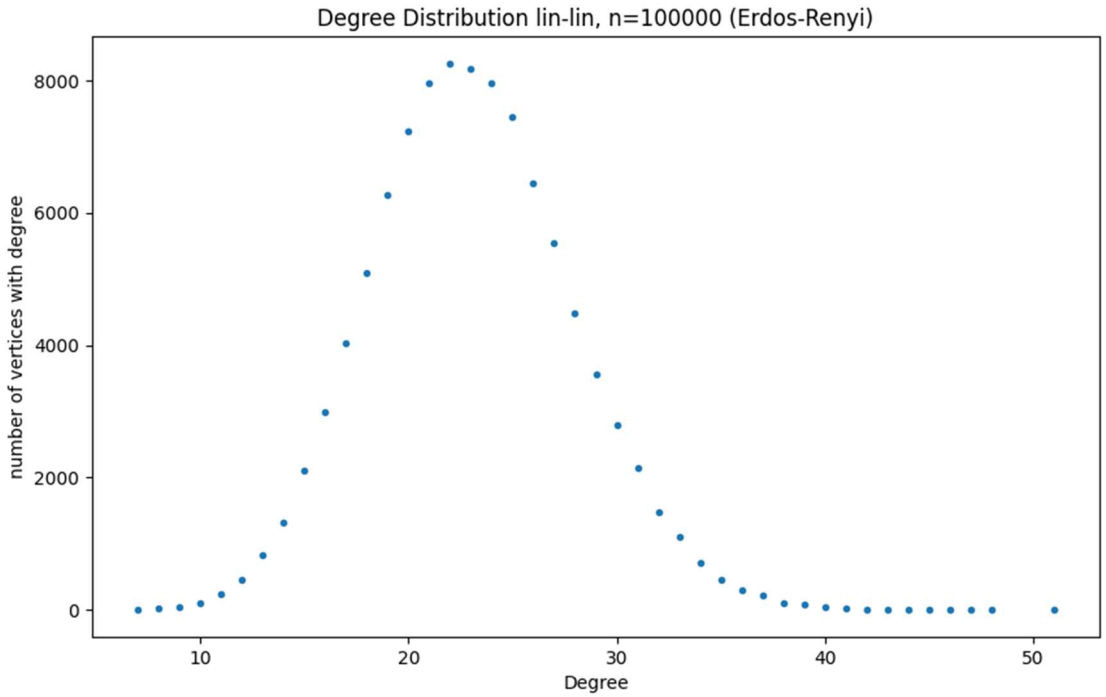
Degree Distribution (Erdos-Renyi)

The degree distributions for random Erdos-Renyi graphs appear to follow a normal distribution. This distribution therefore does not have a power law. The distributions are plotted on a linear-linear scale and log-log scale, shown below.

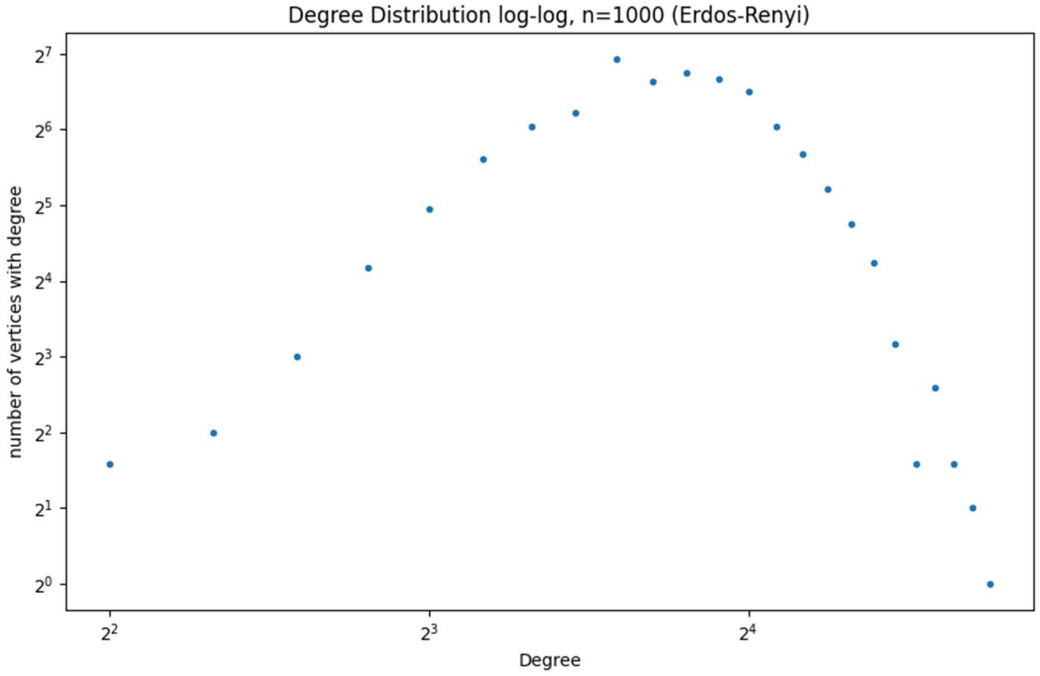
(Lin-lin scale)

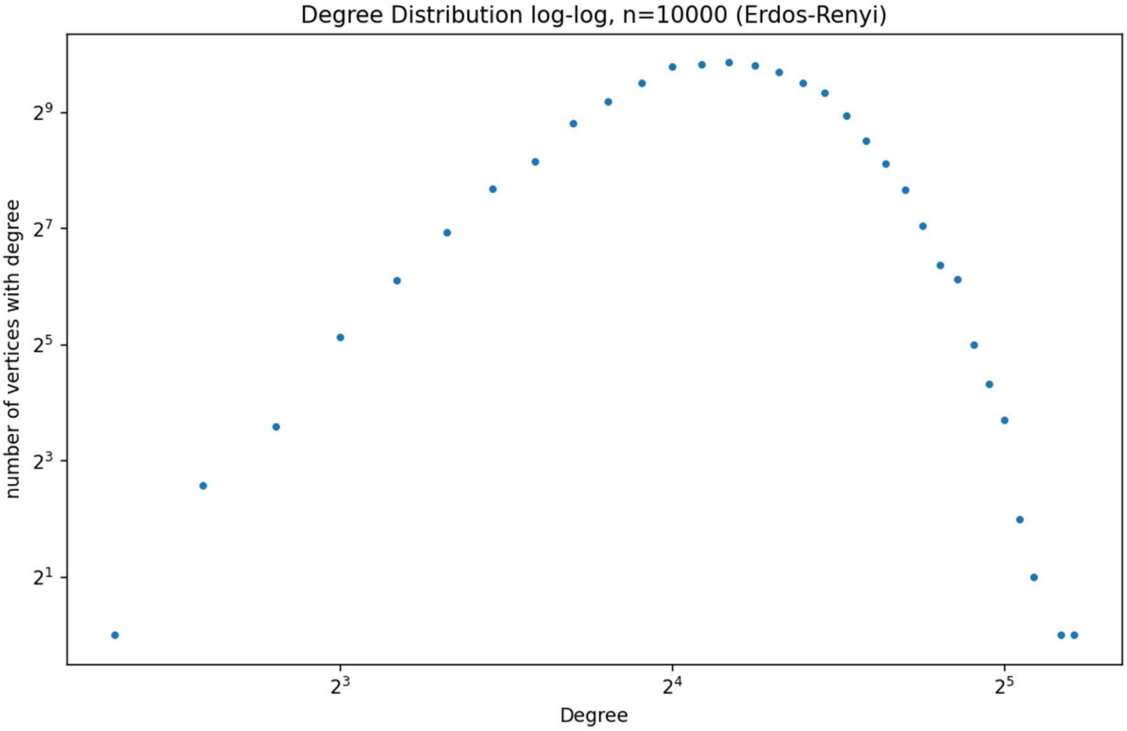


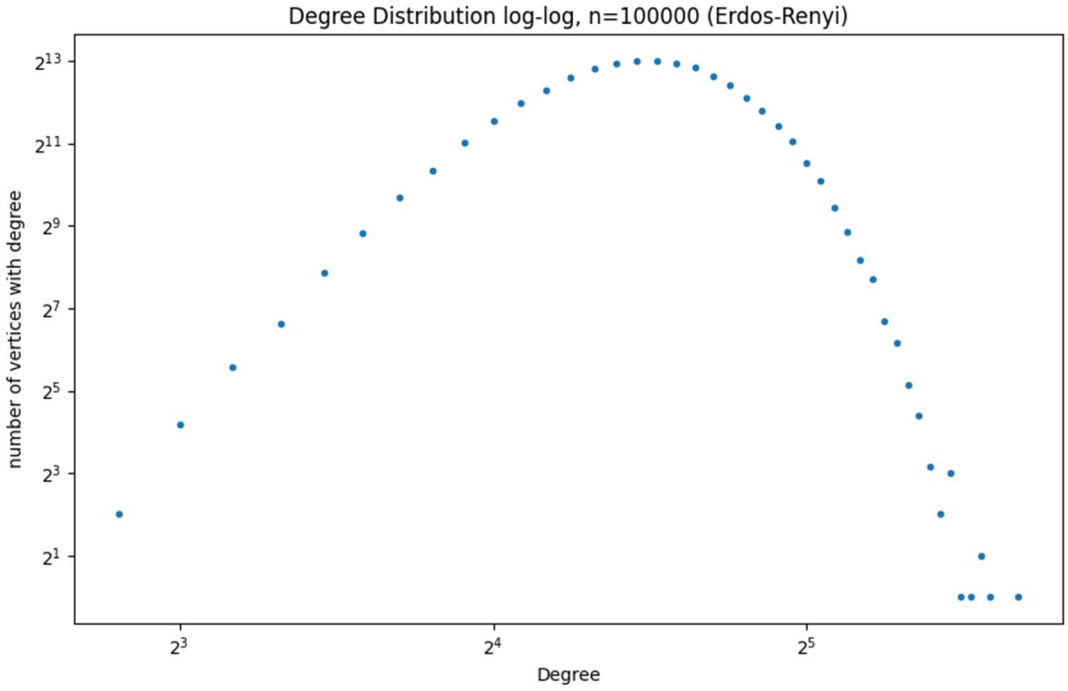




(Log-log scale)



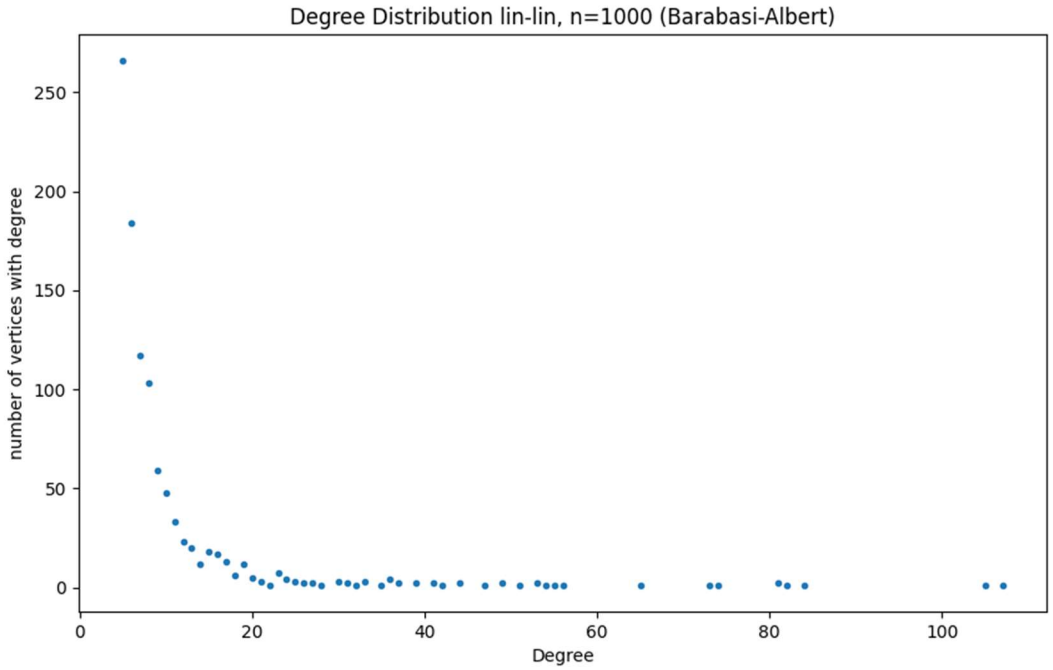

Degree Distribution log-log, n=10000 (Erdos-Renyi)
number of vertices with degree
2^9
2^7
2^5
2^3
2^1
2^3
2^4
2^5
Degree

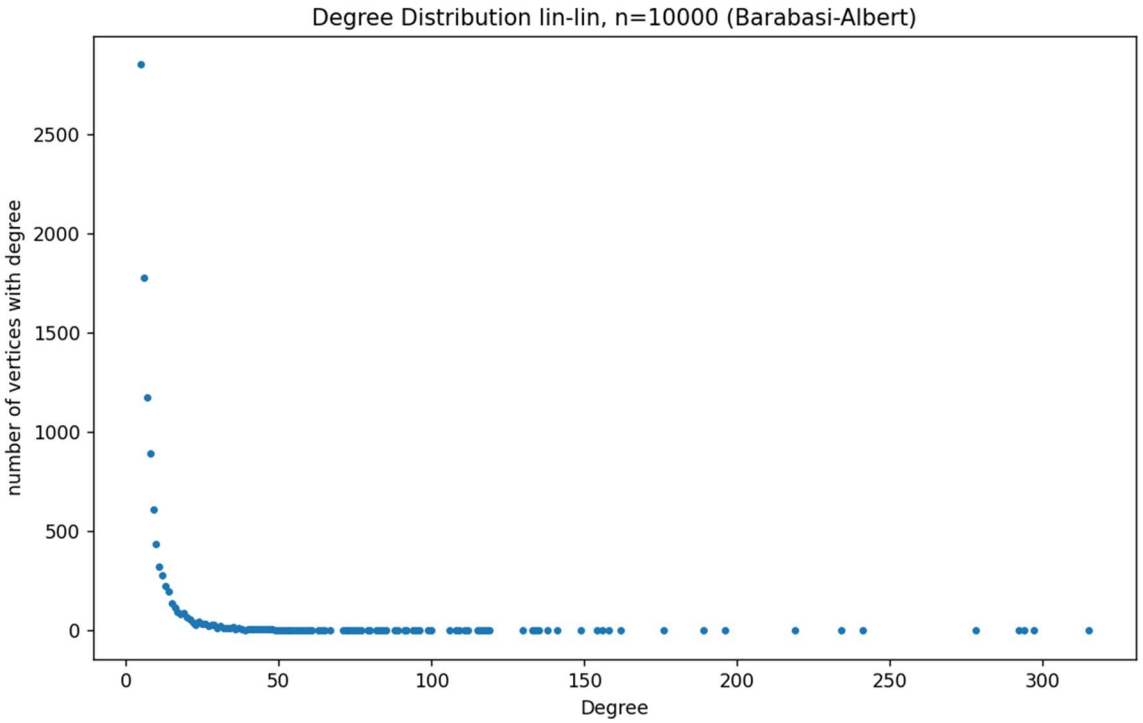

Degree Distribution log-log, n=100000 (Erdos-Renyi)
number of vertices with degree
2^{13}
2^{11}
2^9
2^7
2^5
2^3
2^1
2^3
2^4
2^5
Degree

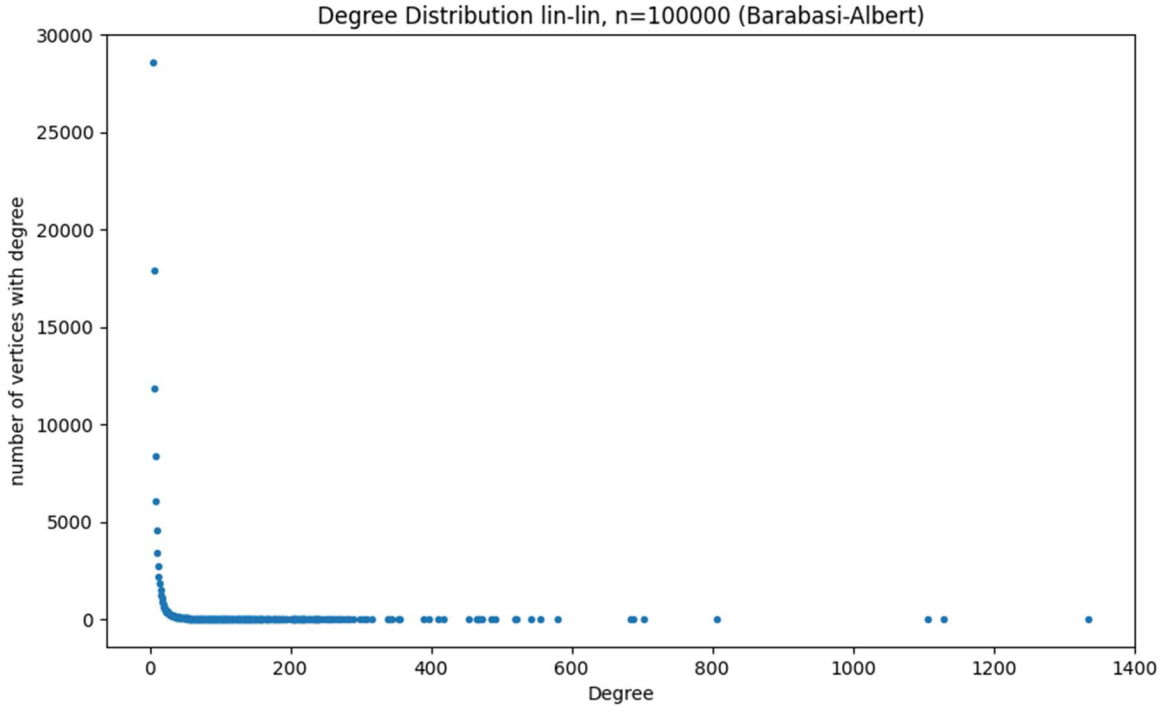
Degree Distribution (Barabasi-Albert)

The degree distributions for random Barabasi-Albert graphs do not follow a normal distribution. The distribution is heavily skewed to the right, being long tailed. This distribution therefore has a power law. The distributions are plotted on a linear-linear scale and log-log scale, shown below. A slope of a best-fit line is found in the log-log scale and an exponent of that power law is determined.

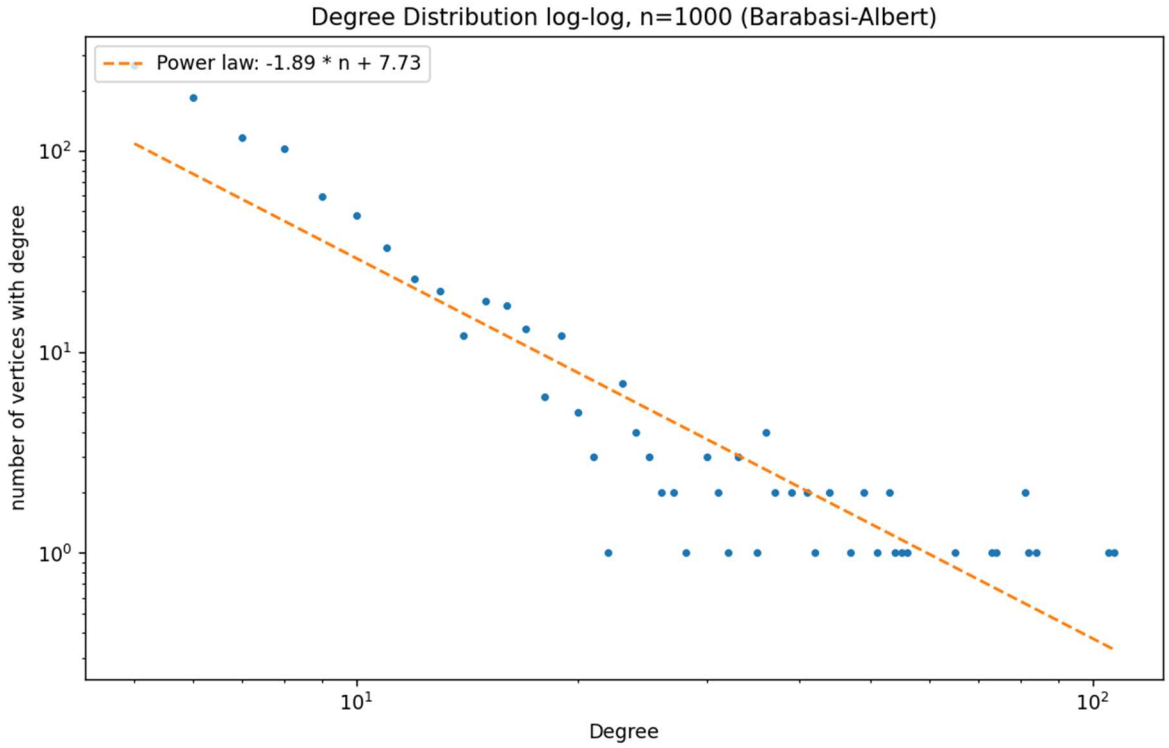
(Lin-lin scale)

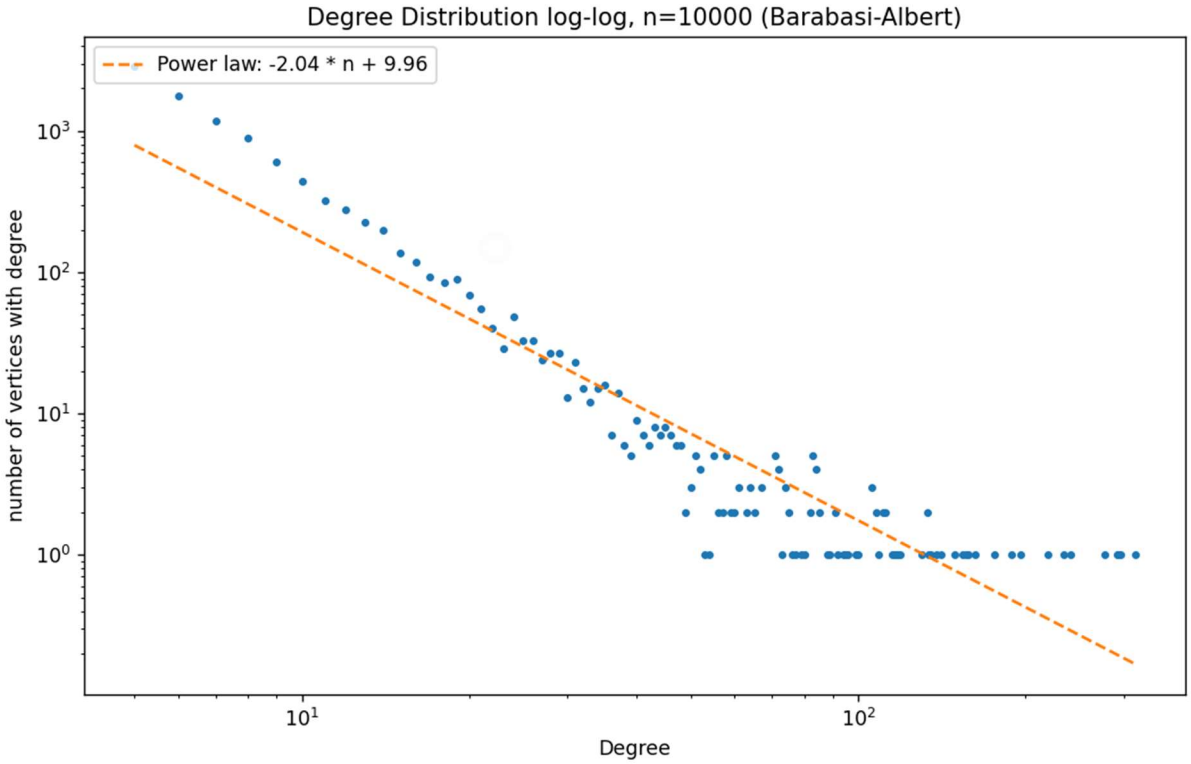


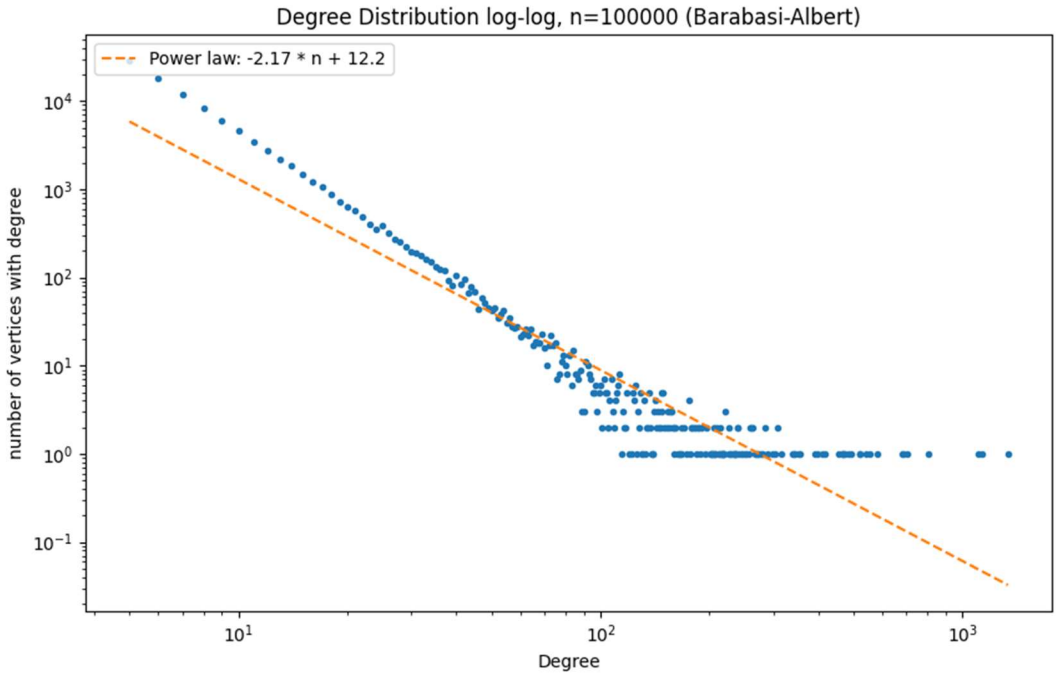

Degree Distribution lin-lin, n=10000 (Barabasi-Albert)
number of vertices with degree
Degree
0
500
1000
1500
2000
2500
0
50
100
150
200
250
300


Degree Distribution lin-lin, n=100000 (Barabasi-Albert)
number of vertices with degree
Degree
0
5000
10000
15000
20000
25000
30000
0
200
400
600
800
1000
1200
1400

(Log-log scale)

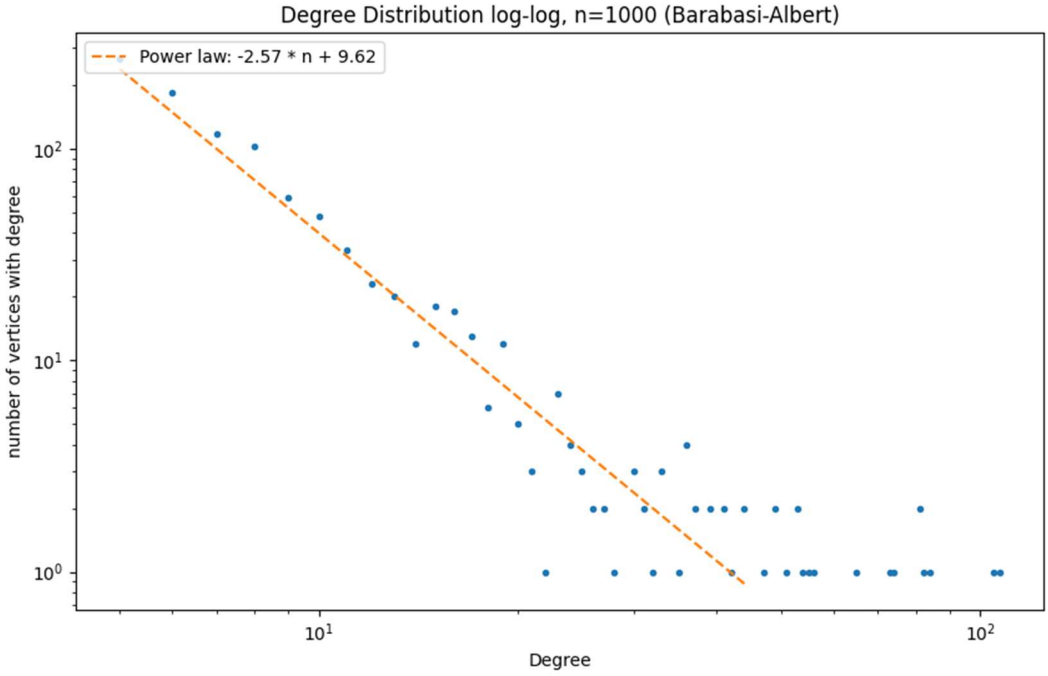


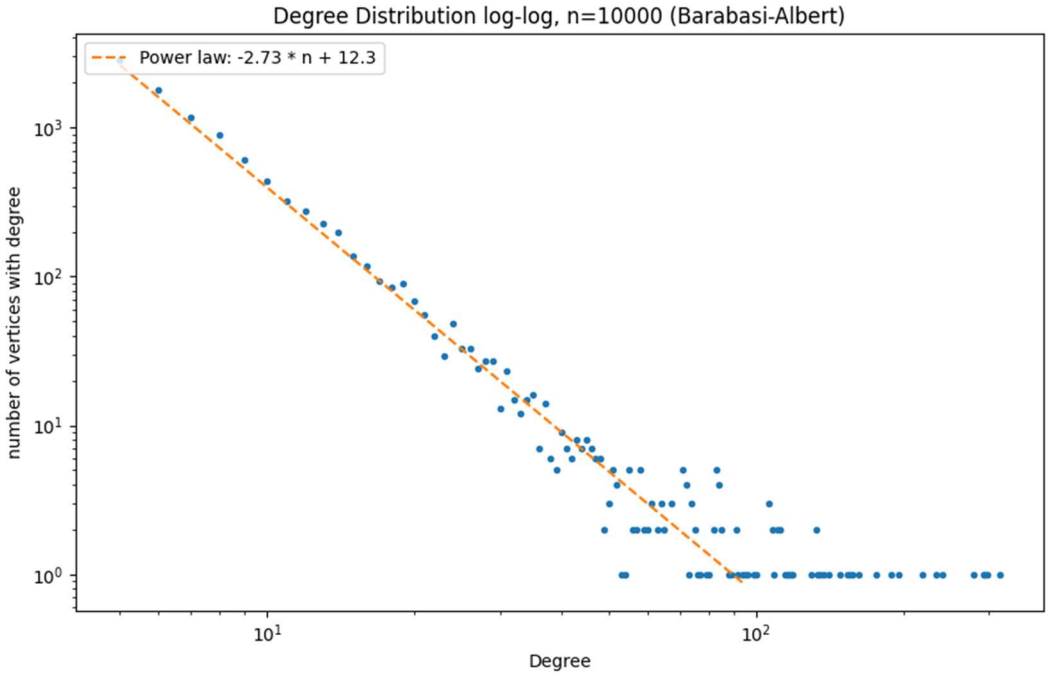


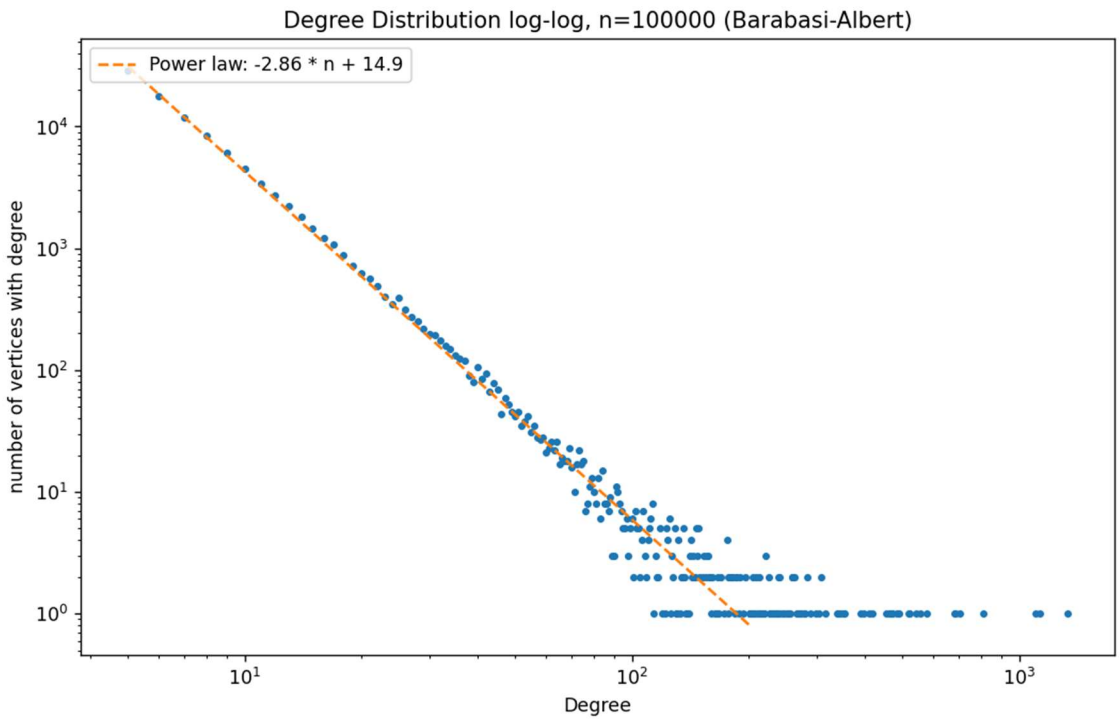


The exponent of the power law relationship is, for n=1000,10000,100000 respectively, 1.89, 2.04, and 2.19. The exponents grow as n grows.

The best fit line for the power law relationship appears to accommodate for the tail end of the data too heavily and does not reflect an accurate fit for the first half of the degrees. Shown below are the same log-log graphs showing the power law relationship with a best fit line that ignores the last 30% tail end of the data, allowing for a more accurate exponent to be reported.


Degree Distribution log-log, n=1000 (Barabasi-Albert)
Power law: -2.57 * n + 9.62
number of vertices with degree
Degree
10^0
10^1
10^2


Degree Distribution log-log, n=10000 (Barabasi-Albert)
Power law: -2.73 * n + 12.3
number of vertices with degree
Degree
10^0
10^1
10^2
10^3



The values of the exponents change to 2.57, 2.73, and 2.86 respectively for n=1000,10000,100000. The values of the exponents still grow when n grows.

Conclusion

The diameter or maximum eccentricity of a network increases when the number of vertices increase for both Erdos-Renyi and Barabasi-Albert random graphs, proportional to a rate the function log n. The clustering coefficient of a network decreases when the number of vertices increase for both types of random graphs, and decreases at an exponentially decaying rate, faster than a rate given by the function log n. The degree distribution for Erdos-Renyi random graphs follows a normal distribution and does not follow a power law. The degree distribution for Barabasi-Albert random graphs is skewed right and follows a power law. The exponent characterized by the power law increases as the number of vertices increase.