SSIE 641 Homework 2

Grant Aguinaldo

October 19, 2021

1 Problem 1

For this problem, we like to point out that the approach that is presented, follows the work of Barabàsi, et al. (Science, 86, 509). To start, we assume that the network contains an initial set of nodes m_0. At each time step, t, the network grows by adding a new node, m, such that $m < m_0$. Further, we assume that the system size, that is, the total number of nodes in the network, at time, t is given by the quantity, $m_0 + t$.

To develop a model of a growing network where new new node is added to with equal probability (i.e., uniform), we note the following: the probability that a *new node* will be connected to node i is given by $\Pi(k) = 1/(m_0 + t - 1)$. Assuming that $t > m_0$, we can simplify by $\Pi(k) = 1/t$. In words, this says that the probability that a new node will be connected to node i is proportional to the amount of time steps that have taken place, which is a proxy for the total number of nodes that are in the system at time t.

By using a mean field approximation, we can estimate the rate the node i acquires edges is given by the partial differential equation below, with an initial condition $k_i(t) = m$.

$$\frac{\partial k_i(t)}{\partial t} = \frac{m}{t} \tag{1}$$

In words, this represents the rate at which the degree of node i changes with time. Solving Equation 1 by separation of variables affords:

$$k_i(t) - k_i(t_i) = m \ln(t) - m \ln(t_i) \tag{2}$$

Substituting in the initial condition, and simplification leads to an expression that explains the degree of node i from the time step t_i to t.

$$k_i(t) = m \ln\left(\frac{t}{t_i}\right) + m \tag{3}$$

Solving Equation 3 for t_i affords:

$$t_i = te^{(1-k/m)} \tag{4}$$

The probability that node i has a degree smaller than k is given by the expression below.

$$\mathcal{P}(k_i(t) < k) = \mathcal{P}(t_i > te^{(1-k/m)}) \tag{5}$$

Assuming that nodes are added to the system (where the total number of nodes is given by the expression, m_0+t) in equal time steps, we can obtain the following expression.

$$\mathcal{P}(t_i > te^{(1-k/m)}) = 1 - \mathcal{P}(t_i \leq te^{(1-k/m)}) = 1 - \frac{te^{(1-k/m)}}{(m_0 + t)} \tag{6}$$

The probability density $\mathcal{P}(k)$ can be then calculated by $\partial\mathcal{P}(k_i(t) < k)/\partial t$, which leads to the solution below.

$$\frac{\partial\mathcal{P}(k_i(t) < k)}{\partial t} = \frac{t}{m(m_0 + t)} \cdot e^{1-(k/m)} \tag{7}$$

Therefore, the probability density for exponential connectivity is given by the expression shown in Figure 1.

$$\mathcal{P}(k_i(t)) = \frac{t}{m(m_0 + t)} \cdot e^{1-(k/m)} \tag{8}$$

Finally, Figure 1 shows the linear relationship between $\mathcal{P}(k)$ (from Equation 8) and k on a semi-log plot. The code used to generate Figure 1 is hosted on Github.[1]

2 Problem 02

Figure 2 shows the visualized network from the Supreme Court Citation Network (i.e., allcites.txt). In this figure, we have selected the largest connected component (LCC) from the Supreme Court Citation Network, used Louvain method to detect communities in the LCC, and created an aggregated network of the LCC where each detected community is represented as a node (represented as a "quotient_graph").

[1] https://github.com/grantaguinaldo/ssie/blob/master/ssie641/homework-02/ssie-641-homework-02-problem-01.py

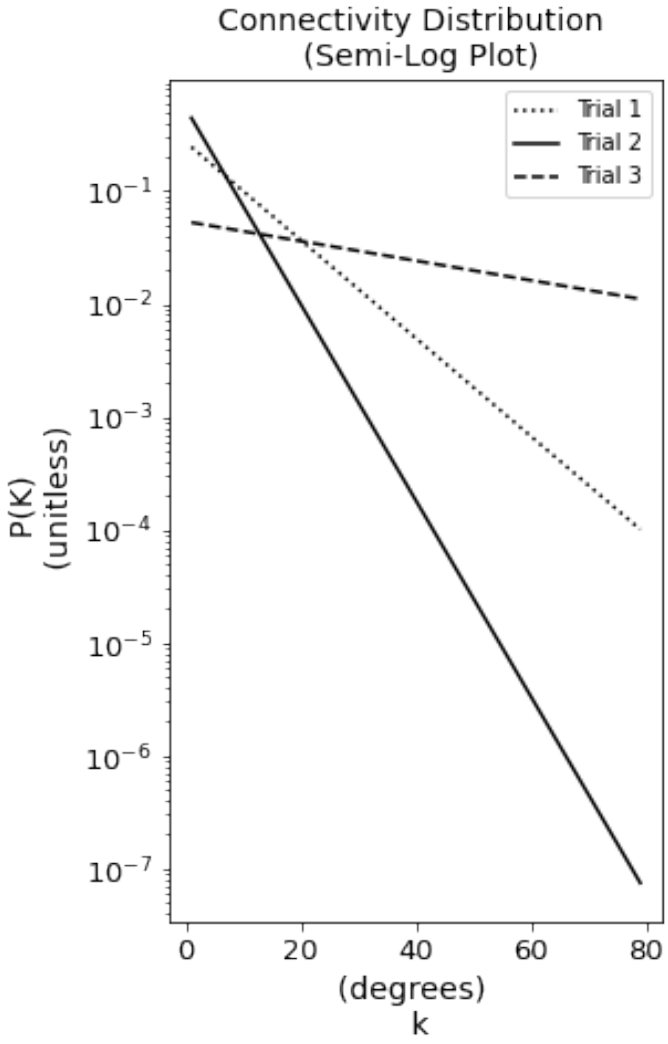

Figure 1: The exponential connectivity distribution for a model without preferential attachment in the case of Trial 1 (t=200,000, m=10, m_0=50), Trial 2 (t=100,000, m=5, m_0=50), and Trial 3 (t=500,000, m=50, m_0=5,000).

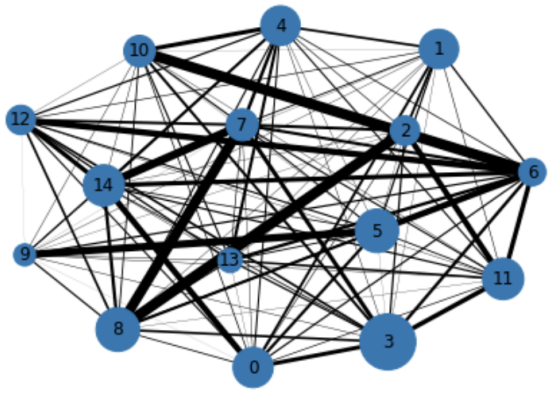

Figure 2: The Supreme Court Citation Network visualized as a quotient graph.

In addition, we have also changed the size of each node to reflect the size of community and also changed the edge weights to reflect the amount of connections between each community. The code that was developed to produce Figure 2 is provided in the Appendix and is also hosted in Github.[2]

[2]https://github.com/grantaguinaldo/ssie/blob/master/ssie641/homework-02/ssie-641-homework-02-problem-01.ipynb

A Appendix A

In [1]:

```
import numpy as np
import networkx as nx
import community.community_louvain as community
# https://stackoverflow.com/questions/53017174/attributeerror-module-networkx-algorithms-community-has-no-attri

g = nx.read_edgelist('allcites.txt')
lcc = max(nx.connected_components(g), key=len)
lcc_sg = g.subgraph(lcc).copy()
bp = community.best_partition(lcc_sg)
cidx = list(set(bp.values()))

nlist = []
for cid in cidx:
    ilist = []
    for i in list(g.nodes):
        try:
            if bp[i] == cid:
                ilist.append(i)
        except:
            pass
    nlist.append(ilist)

qg = nx.quotient_graph(lcc_sg, nlist)
community_nodes = list(qg.nodes)
qg_properties = [qg.nodes[each] for each in community_nodes]

total = sum([each['nnodes'] for each in qg_properties])
freq_comm = [each['nnodes']/total for each in qg_properties]
node_density_dict = [{'node': i, 'node_density': each} for i, each in enumerate(freq_comm)]
community_nodes = list(qg.nodes)

cart_product_node_idx = []
a = [i for i in range(0, len(community_nodes))]
b = [i for i in range(0, len(community_nodes))]
for i in range(len(a)):
    for j in range(len(b)):
        if i != j:
            cart_product_node_idx.append((a[i], b[j]))

edge_density_dict = []
for each in cart_product_node_idx:
    results = dict()
    results['node_01'] = int(each[0])
    results['node_02'] = int(each[1])
```

```
    results['weight'] = qg.edges[community_nodes[int(each[0])], community_nodes[int(each[1])]]['weight']
    edge_density_dict.append(results)

total_edges = sum([each['weight'] for each in edge_density_dict])

for each in edge_density_dict:
    each['weight_density'] = each['weight'] / total_edges
```

In [2]:

```
G = nx.Graph()
G.add_edges_from(cart_product_node_idx)
nx.draw(G,
        node_size=[each['node_density']*10000 for each in node_density_dict],
        with_labels=True, width=[each['weight_density']*250 for each in edge_density_dict])
```