Home Assignment

Small World Networks

Izza **AE18B006**
BTech Aerospace Engineering + IDDD Complex Systems and Dynamics 30th September 2021

Overview of Watts Strogatz Paper on Collective Dynamics of small-world networks

Watts and Strogatz started with two graphs that are well known : Regular and Random Graphs. They found the two Properties of the Graphs, Clustering and Path Length.

- **Clustering :** It quantifies the likelihood that two nodes that are connected to the same node are connected to each other.
- **Path Length :** It measures the average distance between all pairs of nodes in a Graph.

The Results they found are as follows:

Type of Graph	Path Length	Clustering Coefficient
Regular Graph	High	High
Random Graph	Low	Low

Table 1: Random vs Regular

Neither of these had the characteristics of Social Networks which have **short path lengths and high Clustering Coefficient**. Hence a random rewiring procedure was formed to create a **generative model** of Small World Networks which satisfied the desired properties.

Steps followed:

- Form a Regular Ring Lattice with n nodes and k nearest neighbours.
- Rewire the edges according to a probability p and plot the same.
- Calculate the Clustering Coefficient.
- Calculate the same for a range of values of p. p = 0 is for Regular and p = 1 is for Random Networks.

The edges are considered in a particular order and each one is rewired with probability p. If an edge is rewired, the first node is left unchanged and the second node is chosen at random. Self loops or multiple edges are not allowed.

Regular Graphs

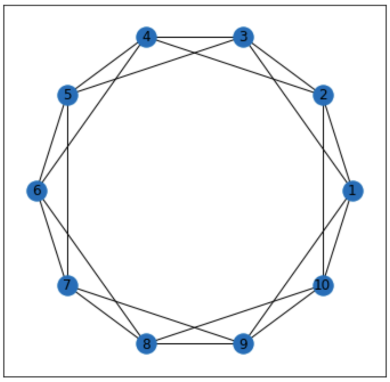


Figure 1: n = 10 , k = 4

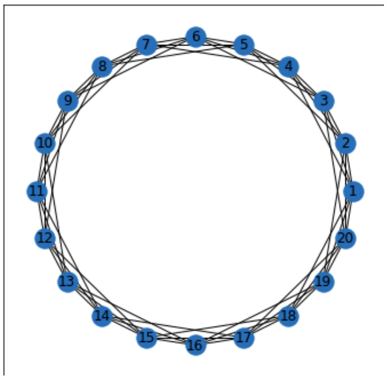


Figure 2: n = 20 , k = 6, p = 0.4

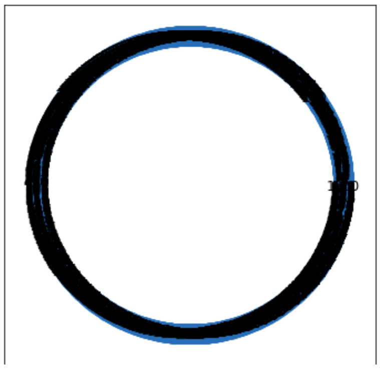


Figure 3: n = 1000 , k = 10

Image of Connectivity Matrix

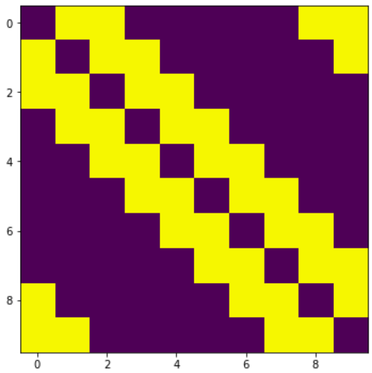


Figure 4: n = 10 , k = 4

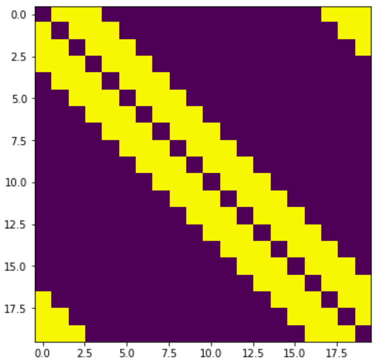


Figure 5: n = 20 , k = 6, p = 0.4

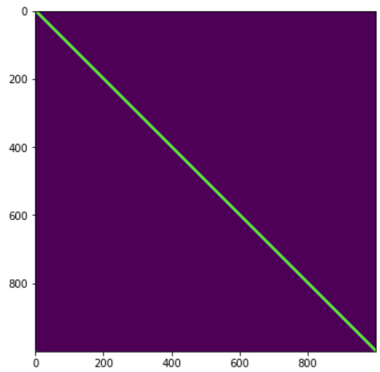


Figure 6: n = 1000 , k = 10

Small World networks

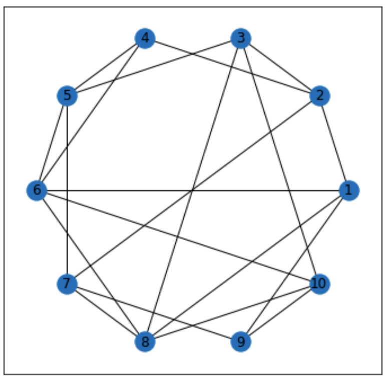


Figure 7: n = 10 , k = 4

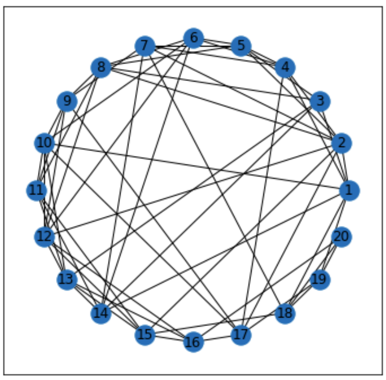


Figure 8: n = 20 , k = 6, p = 0.4

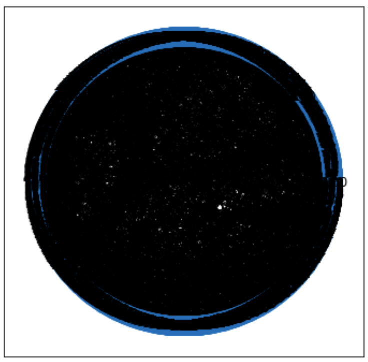


Figure 9: n = 1000 , k = 10

Image of Connectivity Matrix

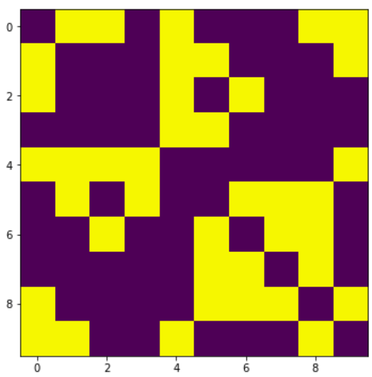


Figure 10: n = 10 , k = 4

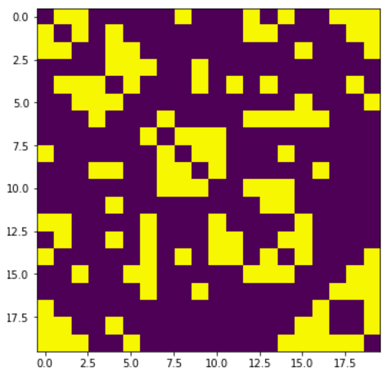


Figure 11: n = 20 , k = 6, p = 0.4

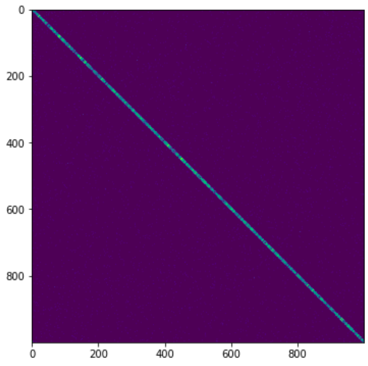


Figure 12: n = 1000 , k = 10

Random Graph Networks

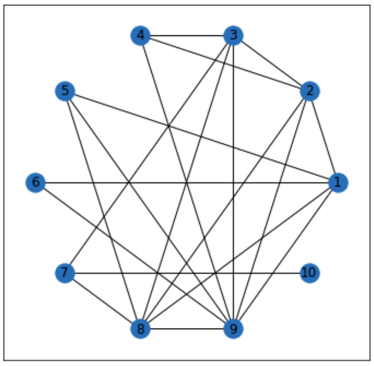


Figure 13: n = 10 , k = 4

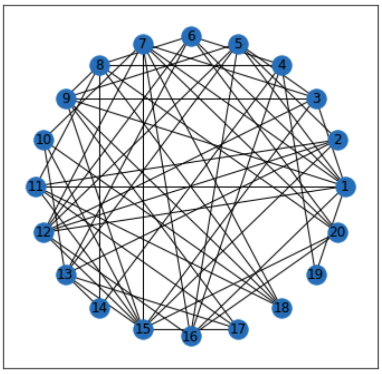


Figure 14: n = 20 , k = 6, p = 0.4

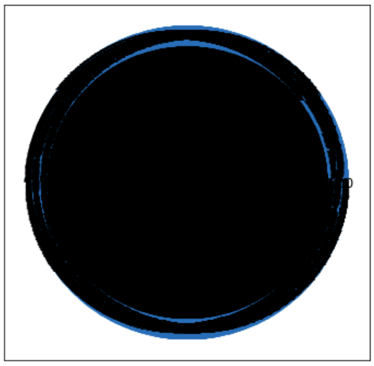


Figure 15: n = 1000 , k = 10

Image of Connectivity Matrix

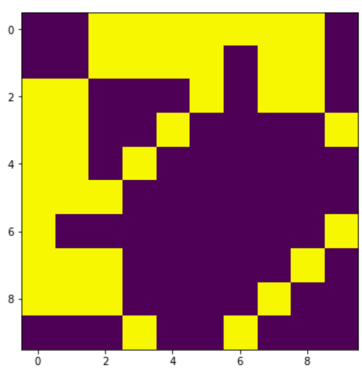


Figure 16: n = 10 , k = 4

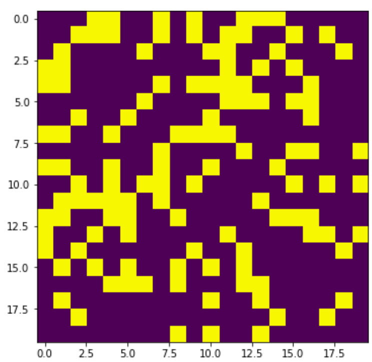


Figure 17: n = 20 , k = 6, p = 0.4

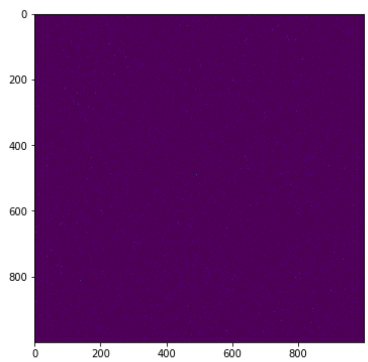


Figure 18: n = 1000 , k = 10

Results

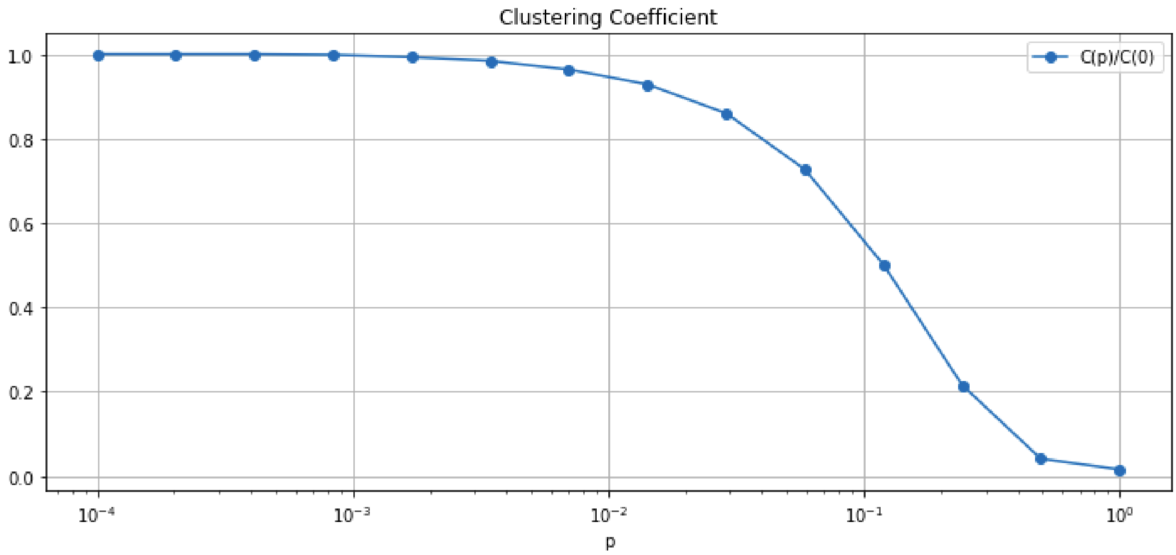
With only the above three types of networks, the calculated Clustering Coefficient is as follows:

n and k	Regular	Small World (p = 0.4)	Random
n = 10 , k = 4	0.5	0.46	0.33
n = 20 , k = 6	0.599	0.323	0.382
n = 1000, k = 10	0.66	0.1465	0.0092

Table 2: Clustering Coefficient

It is seen that Small World Networks don't have such a low value of Clustering Coefficient like the Random Networks. C(p) remains almost constant at its value for a regular lattice indicating that transition to small-world is almost undetectable at local level.

Clustering Coefficient Ratio Plot



Code

```
'''Importing the needed Modules'''

import networkx as nx                    # For Manipulating Graphs
import matplotlib.pyplot as plt          # For Plotting
import numpy as np                       # For getting numpy arrays of data
import random                            # To get Random Values

'''To get the pairs'''
def pairs(nodes):
    for i,u in enumerate(nodes):
        for j,v in enumerate(nodes):
            if i>j:
                yield u, v

'''Getting the Adjacent edges '''
def adjacent_edges(nodes, halfk):
    n = len(nodes)
    for i, u in enumerate(nodes):
```

```
        for j in range(i+1, i+halfk+1):
            v = nodes[j % n]
            yield u, v

'''Making the Regular Network '''
def ring_lattice(n, k):
    G = nx.Graph()
    nodes = range(1,n+1)
    G.add_nodes_from(nodes)
    G.add_edges_from(adjacent_edges(nodes, k//2))
    return G

'''Defining the rewired Graph '''
def rewire(G, p):
    nodes = set(G)
    for u, v in G.edges():
        if np.random.random() < p:
            choices = nodes - {u} - set(G[u]) # Choices are the nodes other
            than itself and the already present connections
            new_v = np.random.choice(list(choices))
            G.remove_edge(u, v)
            G.add_edge(u, new_v)
    return G

'''Clustering Coefficient'''

def clustering_coefficient(G):
    c=[]
    for node in G:
        neighbors = G[node]
        k = len(neighbors)
        if k<2:
            c.append(0)
            continue
        else:
            possible = k*(k-1)/2
            existing = 0
            for v,w in pairs(neighbors):
                if G.has_edge(v,w):
                    existing+=1
            c_v = existing/possible
            c.append(c_v)
    cc = np.mean(c)
    return cc

'''Graph Clustering Coefficient Ratios'''

def C_ratio(lattice,a):
    C_p = []
    m = clustering_coefficient(lattice)
    for p in a:
        C_p.append(clustering_coefficient(rewire(lattice,p)))

    C_ratio = [c/m for c in C_p]
    return(C_ratio)

'''Regular Lattice'''

lattice = ring_lattice(1000, 10)
```

```
pos = nx.circular_layout(lattice)
plt.figure(figsize = (6, 6))
nx.draw_networkx(lattice, pos)


'''Rewired Lattice'''

new  = rewire(lattice,0.0001)

pos1 = nx.circular_layout(new)
plt.figure(figsize = (6, 6))
nx.draw_networkx(new, pos1)

'''Plot'''
x = np.geomspace(0.0001,1,num = 14)
c = C_ratio(lattice,x)
plt.figure(figsize=(12,5))
ax = plt.gca()
ax.semilogx(x,c, marker = 'o', label='C(p)/C(0)')
plt.xlabel('p')
plt.title('Clustering Coefficient')
plt.grid()
plt.legend()
plt.show()
```