CSCI - 5352 (Network Analysis and Modeling) - Problem Set 4

Name of Student : Rajarshi Basak

October 24, 2018

[1] (a) (i) Given that n = number of vertices, c = constant mean degree of each vertex, $p_{in} = \frac{c_{in}}{n}$ = probability of an edge existing within a group, $p_{out} = \frac{c_{out}}{n}$ = probability of an edge existing between two groups,
we have,

$$c_{in} = np_{in}$$
$$c_{out} = np_{out}$$

We also have,

$$2c = c_{in} + c_{out} = np_{in} + np_{out} \tag{1}$$

and

$$\epsilon = c_{in} - c_{out} = np_{in} - np_{out} \tag{2}$$

To solve for p_{in} and p_{out} in terms of only constants c, n and the parameter $\epsilon = c_{in} - c_{out}$, we solve equations (1) and (2) simultaneously.

Solving

$$np_{in} + np_{out} = 2c \tag{3}$$
$$np_{in} - np_{out} = \epsilon \tag{4}$$

simultaneously, we have,

$$2np_{in} = 2c + \epsilon$$

and

$$2np_{out} = 2c - \epsilon$$

from which we have

$$\boxed{p_{in} = \frac{2c + \epsilon}{2n}}$$

and

$$\boxed{p_{out} = \frac{2c - \epsilon}{2n}}$$

[1] (a) (ii) Given $n = 50$, $q = 2$, $c = 5$, we the following values of p_{in} and p_{out} for the range of values of ϵ given by ϵ = 0,4,8:

ϵ	p_{in}	p_{out}
0	0.1	0.1
4	0.14	0.06
8	0.18	0.02

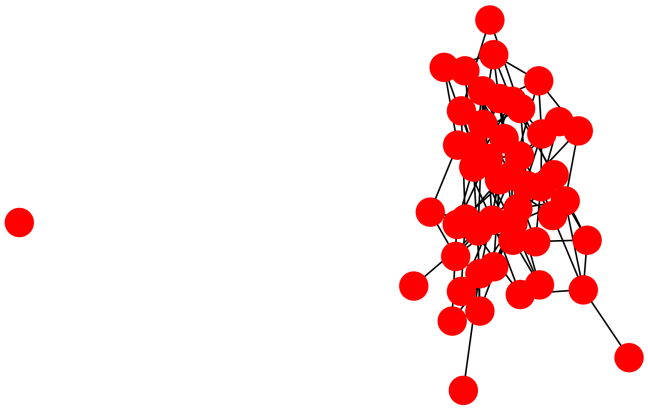

Figure 1: Visualization of graph for $\epsilon = 0$

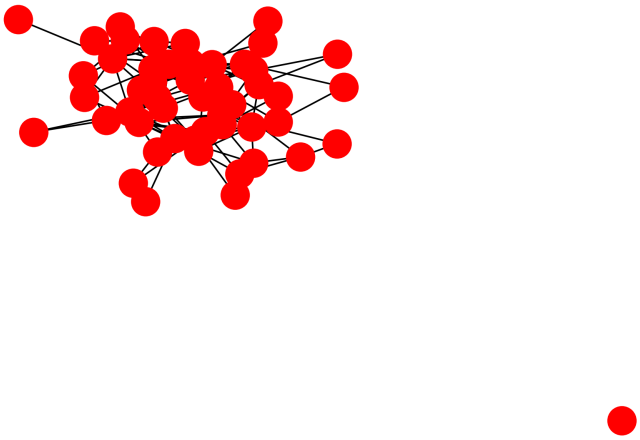

Figure 2: Visualization of graph for $\epsilon = 4$

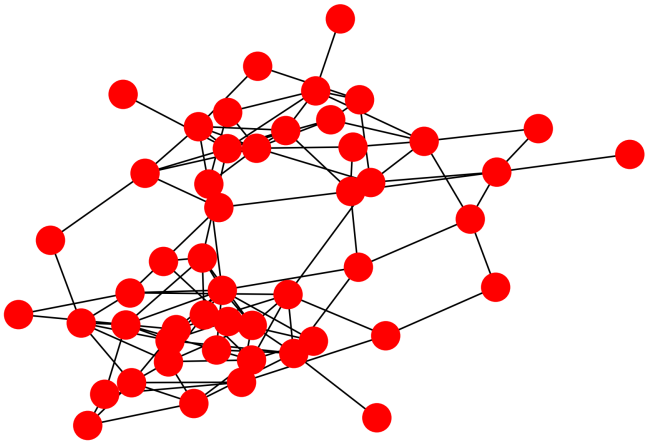

Figure 3: Visualization of graph for $\epsilon = 8$

The in-built visualization tool in networkx called 'draw' is what was used to create the visuals of the graphs above.

It is evident from the visualizations that the strength of community structure increases as the value of ϵ increases from 0 (no community structure at all) to 4 (slight community structure, though not clearly visible) to 8 (some community structure, which is clearly visible).

[1] (b) For the simulations, the probability was varied in the interval [0,1] in steps of 0.01, i.e. there were 100 values of the probabilities, and for each value of probability, a network was generated and the simulation executed a 100 times.

To compare our plots with those when we conducted single runs for each value of the transmission probability, we first show the plots for the single runs:

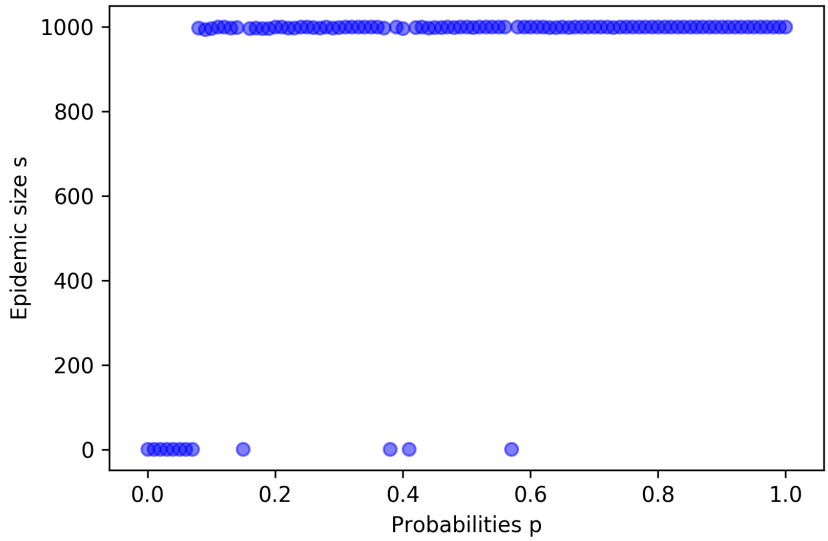

Figure 4: Plot of Epidemic Size s vs. Probability p for a single run in the Planted Partition Model

In this figure, we note that the value of the epidemic size is either a really small value close to or equal to 0 (which means the epidemic has not spread), or a really large value close to or equal to 1(implying the epidemic has been spread). It is interesting to note that there are no values of the epidemic size anywhere in between.

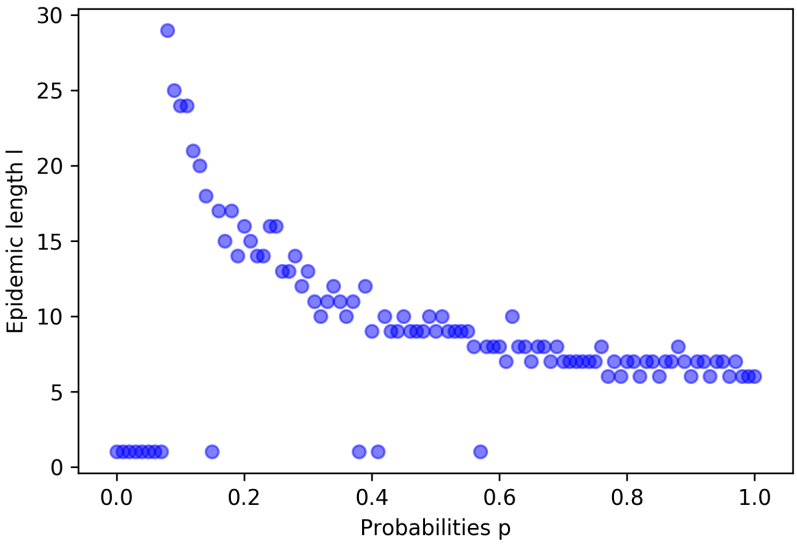

Figure 5: Plot of Epidemic Length l vs. Probability p for a single run in the Planted Partition Model

The figure above tells us that the epidemic length value starts with 0, and suddenly jumps to the maximum value (this is when the phase transition happens and the epidemic spreads), and then gradually falls.

Using $n = 1000$, $c = 8$, and $\epsilon = 0$, the plots for the average epidemic size and the average epidemic length as a function of p in [0,1] are shown below.

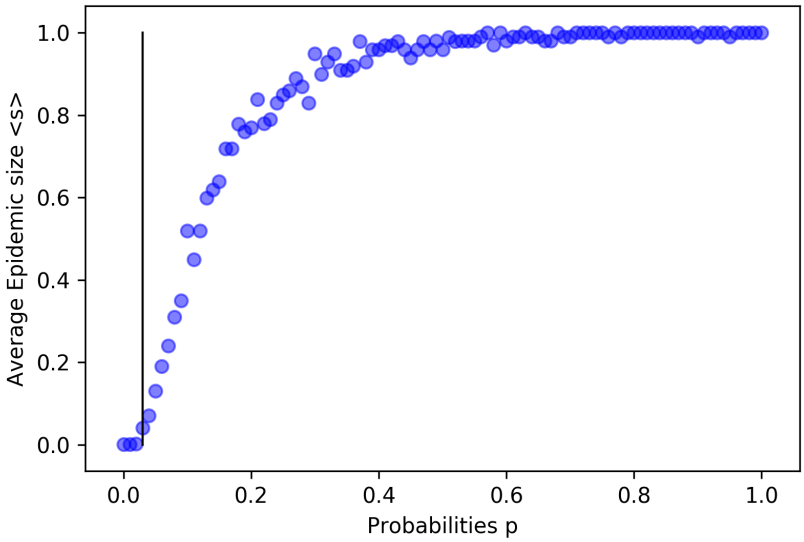

Figure 6: Plot of Average Epidemic Size $< s >$ vs. Probability p in the Planted Partition Model

In the figure above, a vertical line has been shown at the critical value of p, i.e. where a phase transition occurs. This value of p is $\boxed{p_{crit} = 0.03}$

Some experimentalists e.g. those studying protein folding consider the mid-point of the steep rise of the curve to be the point at which the phase transition 'occurs' - this value of $\boxed{p^*_{crit} = (0.03 + 0.45)/2 = 0.24}$ for our simulation.

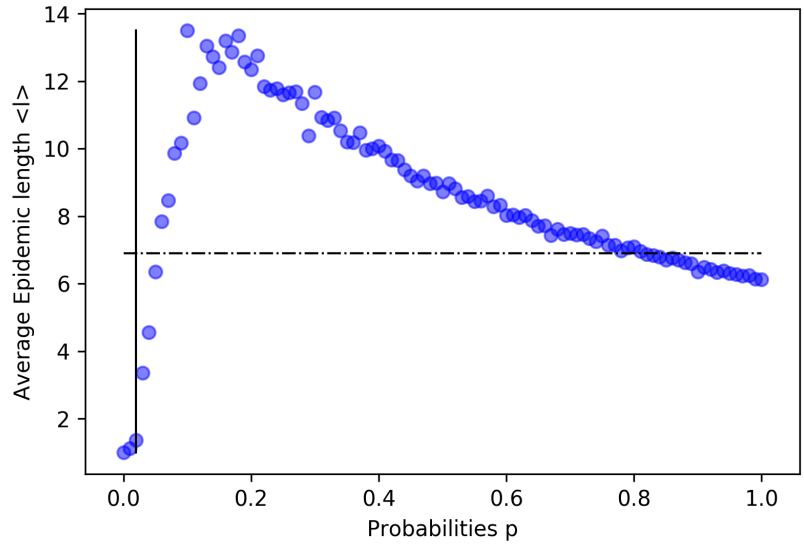

Figure 7: Plot of Average Epidemic Length $< l >$ vs. Probability p in the Planted Partition Model

In the figure above, the vertical line represents the critical value of p at which a phase transition occurs, given by $\boxed{p_{crit} = 0.02}$, and the horizontal line shows $< l >= log(n) = 6.91$. Using the experimentalist's definiton, the phase transition occurs at $\boxed{p^*_{crit} = (0.02 + 0.18)/2 = 0.1}$ for our simulation.

Comments on the qualitative behavior of the measured relationships: In the plot for Average Epidemic Size $< s >$ vs. Probability p, the average epidemic size starts at a value of 0 and remains there for

a small number of values of p, and suddenly starts to rise sharply at the critical value of p (this is where the phase transition occurs) until it reaches the maximum possible value (=1) at some value of p and stays there.

In the plot for Average Epidemic Length $< l >$ vs. Probability p, the average epidemic length starts at a value of 0 similarly starts at a value of 0 and remains there for a small number of values of p, and suddenly starts rising sharply at the critical value of p where is where the phase transition occurs, reaches a maximum value, and then drops gradually as the probability increases to 1.

Discussion of results relative to expectations: We note that in the network, a susceptible node becomes infected when the randomly generated value of the probability is less than or equal to the value of transmission probability p. We also note from the plots shown initially that the value of the epidemic size in reality for a single run is either 0 (or a value very close to 0) or 1 (or a value very close to 1). The values in between shown in the plot above is basically an average over multiple runs for a given transmission probability.

Since the the mean degree of every node is 8 (or the probability of an edge existing between any node is 0.008), initially when the transmission probability is really small, there is a very high chance that the randomly generated probability will be lower than the transmission probability - and since we have on an average only 8 neighbors for the infected node, there is a very low chance that one of them will be infected. This is governed by the number of times (out of 8) that a randomly generated probability value will be less than or equal to the transmission probability value (which is very small). Since the chances are very low, almost all of the time no neighboring vertex will be infected. In the rare case that a vertex is infected, it is even rarer that in the next step a neighboring vertex of the recently infected vertex would be infected. Hence the epidemic is complete in the first (or second step at most).

As the transmission probability grows, the chances of the neighboring vertices being infected also grows, and this is a chain reaction (every time all possible 8 neighbors of the infected vertex are considered for infection in the next step). This means that if there are 4 infected vertices in a step, then in the next step there is a chance that 4*8 = 32 can be infected, and if all 32 are in fact infected, then in the following step, 32*8 = 256 vertices have a chance of being infected. Since the number of possible vertices that can be infected at each step grows exponentially, the number of vertices also grows (although not exponentially). The more the number of infected vertices, the more that will be infected in the next step. Eventually, all the vertices are infected after a certain number of iterations (the epidemic length) and the epidemic is complete. Now the higher the transmission probability, the more the number of vertices infected in a given iteration, and the lesser the number of iterations it takes for the infection to spread to all the vertices. Hence the epidemic completes in a relatively smaller number of steps, and consequently the epidemic length decreases as the transmission probability increases.

What is described above is what is expected from the simulations, and that is we precisely what the results show. As the value of the mean degree (and hence p_{in} and p_{out} are increased, we expect epidemic to complete at lower values of the transmission probability - in other words, as the value of the mean degree increases, the plots should shift to the the left.

Estimate of p_{crit} and why it is special: An estimate of p_{crit} would be $p_{crit} \propto C(p_{in} - p_{out})$, where C is some proportionality constant. This is because p_{crit} is highly dependent on the measure of community structure is the network, and the higher the level of community structure in the network, the higher the expected value of p_{crit}.

This value of p_{crit} is special because as the transmission probability increases to a value above this, the level to which an 'infection' spreads in the network increases it's chances of completion by a huge number.

[1] (c) (i) For the simulations, the ϵ was varied uniformly in the interval [0,16] in steps of 0.1 (there were 160 values of ϵ), and for each value of ϵ, the transmission probability was varied in the interval [0,1] in steps of 100 (there were 100 values of the transmission probability), and for each value of the transmission probability, we did a 100 simulations (a network was generated a 100 times).

Using $n = 200$, $c = 8$, and considering various combinations of the two parameters p in [0,1] and ϵ in [0, 2c] described above, the 3-dimensional plots for the average epidemic size and the average epidemic length as a function of p and ϵ are shown below.

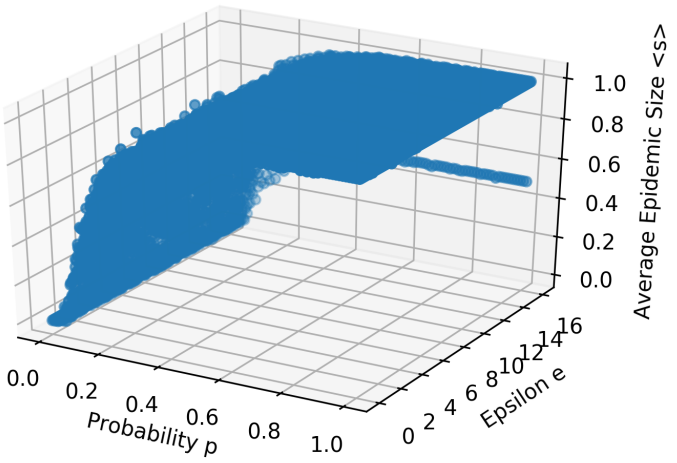

Figure 8: Plot of Average Epidemic Size $< s >$ vs. Transmission Probability p and Epsilon e in the planted partition model

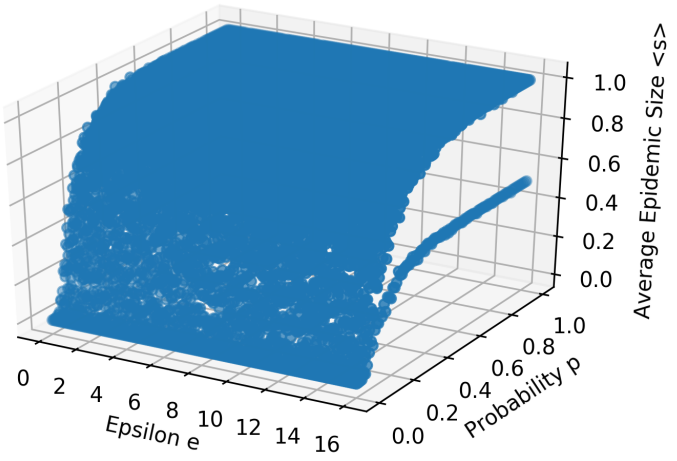

Figure 9: Plot of Average Epidemic Size $< s >$ vs. Epsilon e and Transmission Probability p in the planted partition model

For a better understanding of what the 3-d plots look like, the transmission probability p and the epsilon ϵ has been interchanged in the two figures above.

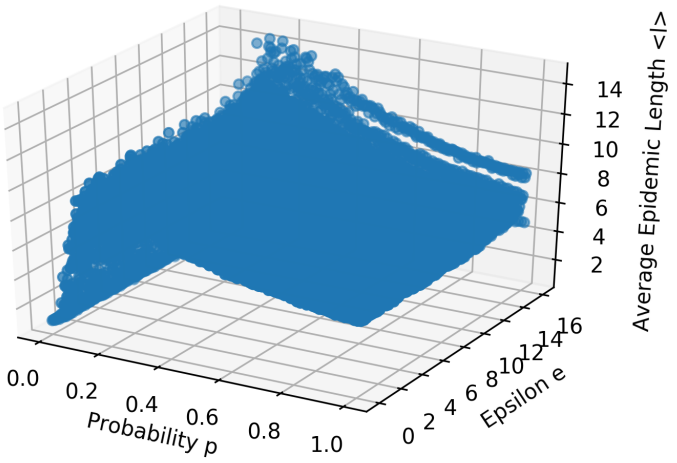

Figure 10: Plot of Average Epidemic Length $< l >$ vs. Transmission Probability p and Epsilon e in the planted partition model

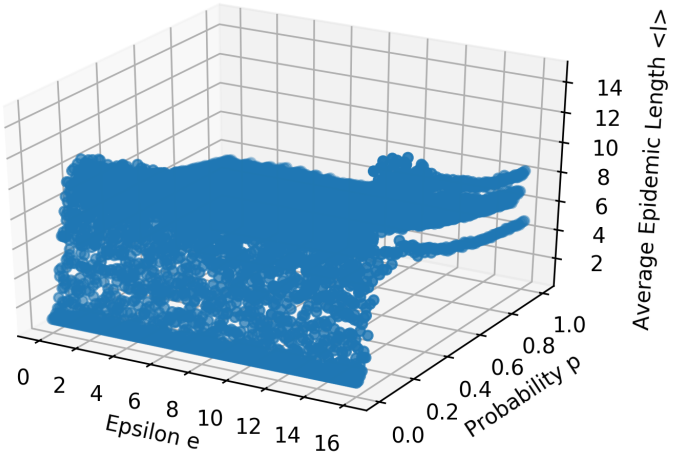

Figure 11: Plot of Average Epidemic Length $< l >$ vs. Epsilon e and Transmission Probability p in the planted partition model

[1] (c) (ii) ***Discussion of the qualitative shape of the functions:*** For the plot of the Average Epidemic Size, the same trend is observed as before - the curve rises sharply at a critical value of the transmission probability p, reaches the maximum value of 1 at some value of p and stays there as we increase p further to 1 - for almost the entire range of values of ϵ, except for the final value of $\epsilon = 16$. At this value, it is observed that the curve jumps sharply at a lower value of the transmission probability p to a value of 0.5 (with a slightly lesser slope) and stays there as we increase p all the way up to 1.

For the plot of the Average Epidemic Length, again a similar trend is observed as before - the curve rises sharply at a critical value of the transmission probability p, reaches the maximum value of around 10 as some value of p and then falls gradually as we increase p further to 1 - for three quarters of the range of values of ϵ. In the last quarter of the range of values of ϵ, the value at which the Average Epidemic Length peaks increases (to about 15 from 10), and consequently it settles at higher value after it falls as we increase the probability to 1. For the final value of ϵ, the curve peaks at a much lower value, and hence settles at a lower value as we increase p to 1.

[1] (c) (iii) ***Explanation for why community structure impacts the shape of the epidemic:*** As we increase the community structure in the network (by increasing the value of ϵ), we increase the number of edges within each group and decrease the number of edges between the two groups.

In the plot for the Average Epidemic Size, this affects the curve to the extent that it takes a little longer for the curve to reach the maximum value of 1, since it takes more time for the epidemic to 'complete' due to the lesser number of edges connecting the two groups. At the maximum value of $\epsilon = 16$, however, all the edges are within each group, and there are no edges between the two groups - hence at this value the curve peaks at half the previous values ($= 0.5$) since the epidemic only 'completes' within the group it started with, and does not spread to the other group owing to the lack of edges between the two groups.

The plot for the Average Epidemic Length is also affected in a similar way - it peaks at a higher value since it takes more time (more iterations of the infection spreading algorithm) for the epidemic to 'complete'. The time it takes to complete increases gradually as we increase ϵ, and for a large measure of community structure, there is a slight sharp rise of the curve (as evident from the plots). At the last value of $\epsilon = 16$, the curve peaks at a much lower value as the infection can only spread to half the number of nodes (compared to before) - hence it takes lesser 'time' for the epidemic to 'complete'.

[2] (a) For the n x n grid network with $n = 50$ nodes, the simulations were run 50 times for each value of the transmission probability in the range [0,1]. The plots are shown below.

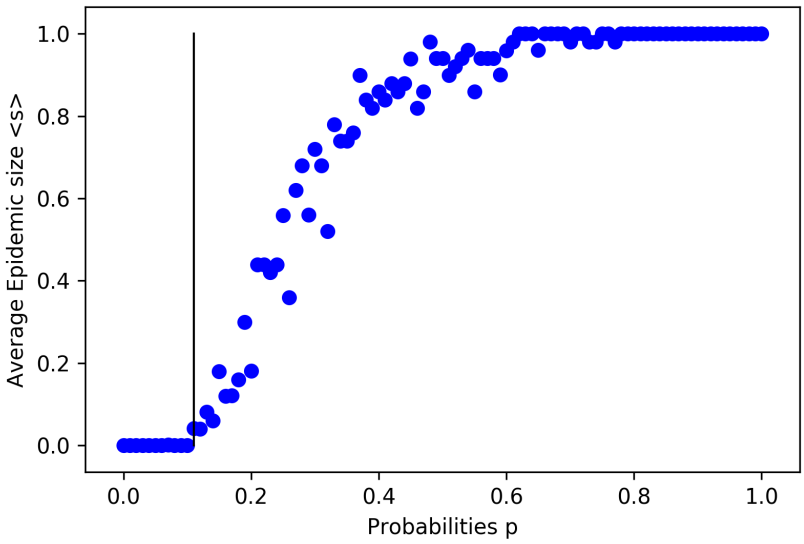

Figure 12: Plot of Average Epidemic Size $< s >$ vs. Probability p in the grid/lattice network

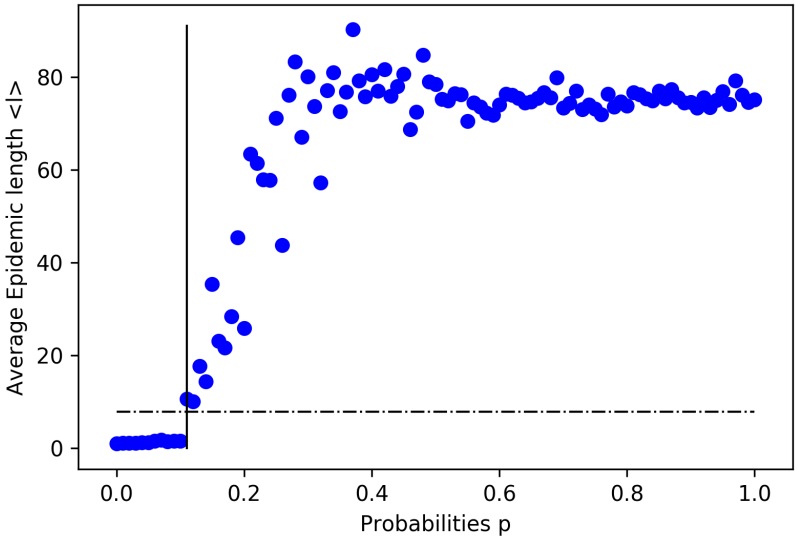

Figure 13: Plot of Average Epidemic Length $< l >$ vs. Probability p in the grid/lattice network

Brief discussion of the shape of the network: We note that each node of grid network (always) has 4 neighbors irrespective of the size of the network. This would a large degree if the network has 4 x 4 = 16 nodes, but a very small degree when the network has 50 x 50 = 2500 nodes (our case).

Since the network has a really small (and almost fixed, except for the vertices on the boundaries) degree, we expect the epidemic size to stay at 0 for longer and take off at higher values of the transmission probability, and that is what we observe. In other words, the plot is shifted to the right.

For identical reasons, the plot for the epidemic length also shows a trend similar to that of the plot of the epidemic size, but there is a key difference. After reaching the maximum value for the epidemic length, the curve then drops a little and stays there for the remaining probability values. Unlike the plot for the planted partition model, where the epidemic length curve falls gradually after peaking, the curve of the epidemic length for the grid network shows greater fluctuations at larger values of the transmission probability and lower fluctuations at larger values of the transmission probability always hovering around a fixed value of the epidemic length.

This trend for the epidemic length curve can be explained as follows: In the grid network, due to the small degree of each vertex (in our case, relative to the size of the network), the number of infected nodes at each iteration of the spreading process is more limited (despite the exponential number of possible infections at each step). Furthermore, due to the structured organization of the nodes, the infection takes a certain amount of time (starting from the randomly infected first vertex) to reach the boundary (vertices with low degree and betweenness centrality) of the network (to complete the epidemic). In the planted partition model (with no community structure), due to the random organization of vertices, the infection spreads from the randomly infected vertex in the beginning to the boundary of the network with greater ease.

[2] (b) For the grid with 'long range' connections, 3-d plots were generated for the long-range edge connection probability q in the range [0,1] in steps of 0.1 (10 values), and the same for the transmission probability p.

The 3-d plots for the Average Epidemic Size and the Average Epidemic Length as a function of both p and q are shown below.

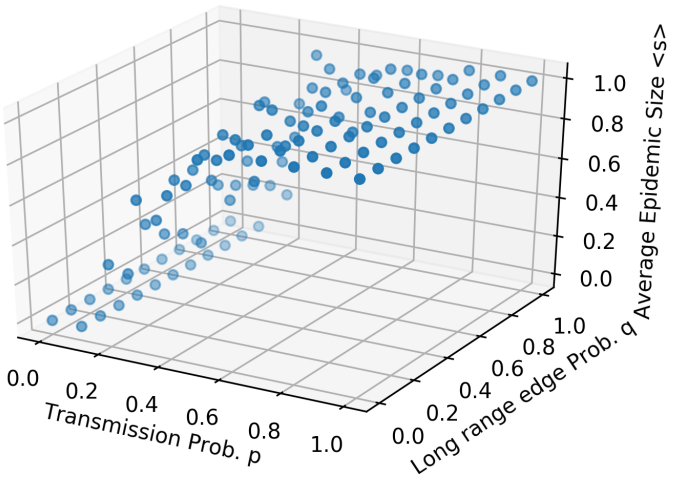

Figure 14: Plot of Average Epidemic Size $< s >$ vs. Transmission Probability p and the Long Range edge Probability q in the grid/lattice network with long range edges

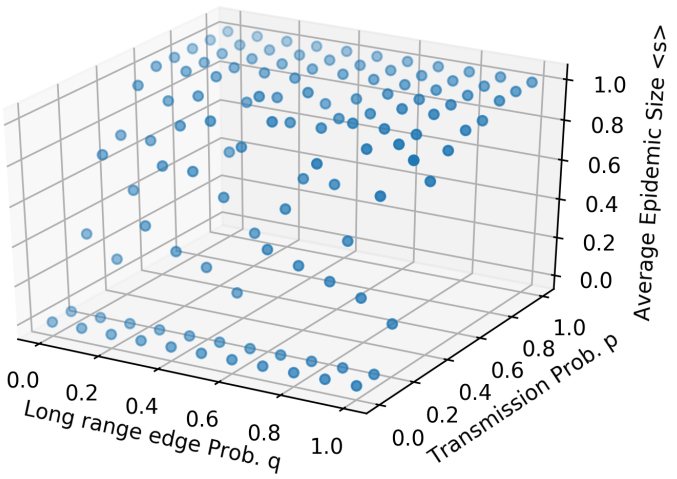

Figure 15: Plot of Average Epidemic Size $< s >$ vs. Long Range edge Probability q and the Transmission Probability p in the grid/lattice network with long range edges

The axes for the Transmission Probability p and the Long Range Edge Probability have been flipped in the figures above to illustrate the plots better.

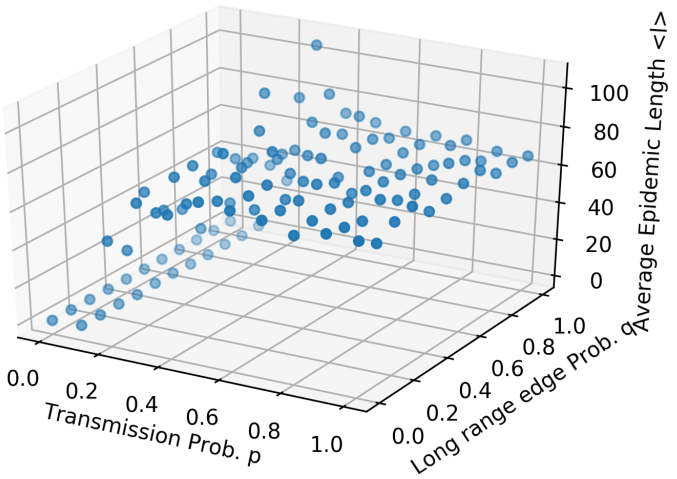

Figure 16: Plot of Average Epidemic Length $< l >$ vs. Transmission Probability p and the Long Range edge Probability q in the grid/lattice network with long range edges

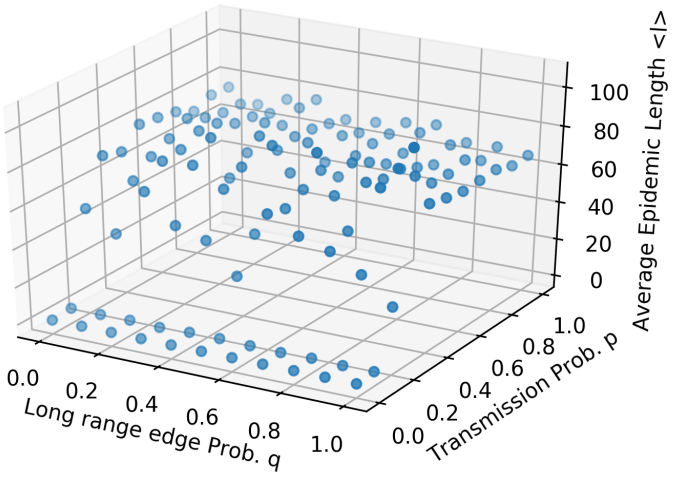

Figure 17: Plot of Average Epidemic Length $< l >$ vs. Long Range edge Probability q and the Transmission Probability p in the grid/lattice network with long range edges

The axes for the Transmission Probability p and the Long Range Edge Probability have been similarly flipped in the figures above to illustrate the plots better.

Effects of Long range edges on Epidemic Dynamics: In the curve for the Average Epidemic Size, as the long range edge probability increases, the curve reaches highest value of the size at a lower value of the transmission probability, i.e. the epidemic completes at a lower value of the transmission probability - this is because the presence of a long range edge basically allows the infection to spread faster (since a long range edge effectively transmits the infection to nodes that are further apart in the network).

For the curve for the Average Epidemic Length, the curve shifts to the left as the value of the long range edge probability increases, and it also settles at a lower value of the transmission probability. This implies that as the long range edge probability increases, an epidemic spreads for a lower value of the transmission probability (as there are more nodes further apart in the network to contribute to the 'spreading' of the infection). Further, for a large value of the long range edge probability, the epidemic takes lesser 'time' to 'complete', owing to the

same reason mentioned above.

Note: Due to lack of computing power, and the duration it took to run each simulation, we had to settle for the grainy resolution in the 3-d plots above.

The code for Problem [1] (a) (ii) has been shown below.

```
import networkx as nx
import matplotlib.pyplot as plt

n = int(50)
c = int(5)
e = [0,4,8]
sizes = [25, 25]

for eps in e:
    p_in = (2*c + eps)/(2*n)
    p_out = (2*c - eps)/(2*n)
    probs = [[p_in, p_out],
             [p_out, p_in]]
    g = nx.stochastic_block_model(sizes, probs, seed=0)
    if (eps == 0):
        nx.draw(g)
        plt.savefig('graph_vis_eps_0', dpi = 300)
        plt.show()
    elif (eps == 4):
        nx.draw(g)
        plt.savefig('graph_vis_eps_4', dpi = 300)
        plt.show()
    elif (eps == 8):
        nx.draw(g)
        plt.savefig('graph_vis_eps_8', dpi = 300)
        plt.show()
```

The code for Problem [1] (b) has been shown below.

```
import networkx as nx
import random
import numpy as np
import matplotlib.pyplot as plt
import time
import math

# Count time
start_time = time.time()

# Initialize all variables
e = 0
q = 2
n = int(1000)
k = int(n/q)
c = int(8)
p_in = (2*c + e)/(2*n)
p_out = (2*c - e)/(2*n)

# Initialize arrays for storing epidemic parameters, probabilities
# and number of runs for each probability
probabilities = np.arange(0.0, 1.001, 0.01)
epidemic_size = []
epidemic_length = []
runs = np.arange(1.0, 100.1, 1.0)
```

```
for pr in probabilities:

    total_infected_size = 0
    total_infected_length = 0

    for run in runs:

        graph = nx.planted_partition_graph(q, k, p_in, p_out, seed=42
                                           , directed=False)
        #nx.draw(graph)
        # Append all nodes to the list nodes
        nodes = []
        for i in graph.nodes():
            nodes.append(i)

        #print(random.randint(0,n))
        # Initialize variables
        infected = []
        inf = random.randint(0,n-1)
        infected.append(inf)
        to_be_infected = [n-1]
        count = 0

        # while new_infected is not equal to infected
        # or no new nodes are infected
        while (to_be_infected != []):
            to_be_infected = []



            #if (prob <= p_in and prob <= p_out):

                #print(infected)
            for i in infected:
                #print('Already infected node:',i)
                for j in graph.neighbors(i):
                    #print('Neighbors of infected node' ,i, 'is node' ,j)
                    if (j not in infected):
                        # Generate a random number between 0 and 1
                        prob = random.uniform(0,1)
                        # Check if the generated number is less than or equal to p
                        if (prob <= pr):
                            # Add all of the current node in 'infected' list's neighbours
                            # to 'infected' that are not in 'infected'
                            to_be_infected.append(j)
                        infected = list(set(infected) | set(to_be_infected))

            if (to_be_infected is not None):
                count += 1

        total_infected_size += len(infected)
        total_infected_length += count

    # Find number of infected nodes and update array
    epidemic_size.append(total_infected_size/len(runs))
    # Find number of steps and update array
    epidemic_length.append(total_infected_length/len(runs))
```

```
# Modifications for plotting
epidemic_size_normalized = []

for i in epidemic_size:
    epidemic_size_normalized.append(i/(n))

for j in range(len(epidemic_size_normalized)):
    if (epidemic_size_normalized[j] >= 0.02):
        #print(probabilities[j])
        p_crit_1 = probabilities[j]
        break

for j in range(len(epidemic_size_normalized)):
    if (epidemic_size_normalized[j] >= 0.99):
        #print(probabilities[j])
        p_crit_2 = probabilities[j]
        break

for j in range(len(epidemic_length)):
    if (epidemic_length[j] >= 1.3):
        p_crit_3 = probabilities[j]
        break


#Plotting <s> vs. p
plt.scatter(probabilities,epidemic_size_normalized, alpha=0.5, color = 'b')
plt.xlabel('Probabilities p')
plt.ylabel('Average Epidemic size <s>')
#plt.ylim((0.9,3.1))
yy_1 = np.linspace(0.0, 1.0, num=100)
xx_1 = []
for i in yy_1:
    xx_1.append(p_crit_1)
plt.plot(xx_1,yy_1,linestyle='-', color='k', linewidth=1)
plt.savefig('s_versus_p_p-100_runs-100_3', dpi = 300)
plt.show()

#Plotting <l> vs. p
plt.scatter(probabilities,epidemic_length, alpha=0.5, color = 'b')
plt.xlabel('Probabilities p')
plt.ylabel('Average Epidemic length <l>')
#plt.ylim((0.9,3.1))
yy_2 = np.linspace(1.0, max(epidemic_length), num=100)
xx_2 = []
for i in yy_2:
    xx_2.append(p_crit_3)
plt.plot(xx_2,yy_2,linestyle='-', color='k', linewidth=1)
xx_3 = np.linspace(0.0, 1.0, num=100)
yy_3 = []
for i in xx_3:
    yy_3.append(math.log(n))
plt.plot(xx_3,yy_3,linestyle='-.', color='k', linewidth=1)
plt.savefig('l_versus_p_p-100_runs-100_3', dpi = 300)
plt.show()

# Record elapsed time
elapsed_time = time.time() - start_time

print(elapsed_time)
```

The code for Problem [1] (c) has been shown below.

```
import networkx as nx
import random
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import time
import math
from mpl_toolkits.mplot3d import Axes3D

# Count time
start_time = time.time()

# Initialize all variables
# e = 0
q = 2
n = int(200)
k = int(n/q)
c = int(8)
eps = np.arange(0.0, 2*c + 0.01, 0.1)

array_epsilon = []
array_probability = []
array_epi_size = []
array_epi_len = []

for e in eps:

    # Define edge drawing probabilities
    p_in = (2*c + e)/(2*n)
    p_out = (2*c - e)/(2*n)

    # Initialize arrays for storing epidemic parameters, probabilities
    # and number of runs for each probability
    probabilities = np.arange(0.0, 1.001, 0.01)
    epidemic_size = []
    epidemic_length = []
    runs = np.arange(0.0, 1.01, 0.01)


    for pr in probabilities:

        total_infected_size = 0
        total_infected_length = 0

        for run in runs:

            graph = nx.planted_partition_graph(q, k, p_in, p_out, seed=42
                                               , directed=False)
            #nx.draw(graph)
            # Append all nodes to the list nodes
            nodes = []
            for i in graph.nodes():
                nodes.append(i)

            #print(random.randint(0,n))
            # Initialize variables
            infected = []
            inf = random.randint(0,n-1)
            infected.append(inf)
```

```
            to_be_infected = [n-1]
            count = 0

            # while new_infected is not equal to infected
            # or no new nodes are infected
            while (to_be_infected != []):
                to_be_infected = []

                for i in infected:
                    #print('Already infected node:',i)
                    for j in graph.neighbors(i):
                        #print('Neighbors of infected node' ,i, 'is node' ,j)
                        if (j not in infected):
                            prob = random.uniform(0,1)
                            if (prob <= pr):
                                to_be_infected.append(j)
                                #print('New nodes added')
                            infected = list(set(infected) | set(to_be_infected))

                if (to_be_infected is not None):
                    count += 1

            total_infected_size += len(infected)
            total_infected_length += count

        array_epsilon.append(e) # Array for the x-axis
        array_probability.append(pr) # Array for the y-axis
        array_epi_size.append(total_infected_size/len(runs)) # Array for the z(1) axis
        array_epi_len.append(total_infected_length/len(runs)) # Array for the z(2) axis


# Modifications for plotting
array_epi_size_norm = []

for i in array_epi_size:
    array_epi_size_norm.append(i/(n))


mpl.rcParams['legend.fontsize'] = 10
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(array_epsilon, array_probability, array_epi_size_norm)
ax.set_xlabel('Epsilon e')
ax.set_ylabel('Probability p')
ax.set_zlabel('Average Epidemic Size <s>')
ax.legend()
plt.savefig('s_vs_p_vs_e_p-100_e-160_runs-100', dpi = 300)
plt.show()


mpl.rcParams['legend.fontsize'] = 10
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(array_epsilon, array_probability, array_epi_len)
ax.set_xlabel('Epsilon e')
ax.set_ylabel('Probability p')
ax.set_zlabel('Average Epidemic Length <l>')
ax.legend()
plt.savefig('l_vs_p_vs_e_p-100_e-160_runs-100', dpi = 300)
plt.show()
```

```
mpl.rcParams['legend.fontsize'] = 10
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(array_probability, array_epsilon, array_epi_size_norm)
ax.set_xlabel('Probability p')
ax.set_ylabel('Epsilon e')
ax.set_zlabel('Average Epidemic Size <s>')
ax.legend()
plt.savefig('s_vs_e_vs_p_p-100_e-160_runs-100', dpi = 300)
plt.show()


mpl.rcParams['legend.fontsize'] = 10
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(array_probability, array_epsilon, array_epi_len)
ax.set_xlabel('Probability p')
ax.set_ylabel('Epsilon e')
ax.set_zlabel('Average Epidemic Length <l>')
ax.legend()
plt.savefig('l_vs_e_vs_p_p-100_e-160_runs-100', dpi = 300)
plt.show()


# Record elapsed time
elapsed_time = time.time() - start_time

print((elapsed_time/3600), 'hours' )
```

The code for Problem [2] (a) has been shown below.

```
import networkx as nx
import random
import numpy as np
import matplotlib.pyplot as plt
import time
import math

# Count time
start_time = time.time()

# Initialize all variables
n = int(50)
m = int(50)

# Initialize arrays for storing epidemic parameters, probabilities
# and number of runs for each probability
probabilities = np.arange(0.0, 1.001, 0.01)
epidemic_size = []
epidemic_length = []
runs = np.arange(1.0, 50.1, 1.0)

for pr in probabilities:

    total_infected_size = 0
    total_infected_length = 0

    for run in runs:

        graph = nx.grid_2d_graph(m, n, periodic=False, create_using=None)
```

```
        # Append all nodes to the list nodes
        nodes = []
        for i in graph.nodes():
            nodes.append(i)

        # Initialize variables
        infected = []
        inf1 = random.randint(0,n-1)
        inf2 = random.randint(0,m-1)
        infected.append((inf1, inf2))
        to_be_infected = [(n-1, m-1)]
        count = 0

        # while new_infected is not equal to infected
        # or no new nodes are infected
        while (to_be_infected != []):
            to_be_infected = []

            for i in infected:
                #print('Already infected node:',i)
                for j in graph.neighbors(i):
                    #print('Neighbors of infected node' ,i, 'is node' ,j)
                    if (j not in infected):
                        prob = random.uniform(0,1)
                        if (prob <= pr):
                            to_be_infected.append(j)
                            #print('New nodes added')
                        infected = list(set(infected) | set(to_be_infected))

            if (to_be_infected is not None):
                count += 1

        total_infected_size += len(infected)
        total_infected_length += count

    # Find number of infected nodes and update array
    epidemic_size.append(total_infected_size/len(runs))
    # Find number of steps and update array
    epidemic_length.append(total_infected_length/len(runs))



# Modifications for plotting
epidemic_size_normalized = []

for i in epidemic_size:
    epidemic_size_normalized.append(i/(m*n))

for j in range(len(epidemic_size_normalized)):
    if (epidemic_size_normalized[j] >= 0.02):
        p_crit_1 = probabilities[j]
        break

for j in range(len(epidemic_size_normalized)):
    if (epidemic_size_normalized[j] >= 0.99):
        p_crit_2 = probabilities[j]
        break

for j in range(len(epidemic_length)):
```

```
    if (epidemic_length[j] >= 2.00):
        p_crit_3 = probabilities[j]
        break

#Plotting <s> vs. p
plt.scatter(probabilities,epidemic_size_normalized, linestyle='-', color='b', linewidth=1)
plt.xlabel('Probabilities p')
plt.ylabel('Average Epidemic size <s>')
#plt.ylim((0.9,3.1))
yy_1 = np.linspace(0.0, 1.0, num=100)
xx_1 = []
for i in yy_1:
    xx_1.append(p_crit_1)
plt.plot(xx_1,yy_1,linestyle='-', color='k', linewidth=1)
plt.savefig('grid_s_versus_p_p-100_runs-50', dpi = 300)
plt.show()

#Plotting <l> vs. p
plt.scatter(probabilities,epidemic_length, linestyle='-', color='b', linewidth=1)
plt.xlabel('Probabilities p')
plt.ylabel('Average Epidemic length <l>')
#plt.ylim((0.9,3.1))
yy_2 = np.linspace(0.0, 91.0, num=100)
xx_2 = []
for i in yy_2:
    xx_2.append(p_crit_3)
plt.plot(xx_2,yy_2,linestyle='-', color='k', linewidth=1)
xx_3 = np.linspace(0.0, 1.0, num=100)
yy_3 = []
for i in xx_3:
    yy_3.append(math.log(n*m))
plt.plot(xx_3,yy_3,linestyle='-.', color='k', linewidth=1)
plt.savefig('grid_l_versus_p_p-100_runs-50', dpi = 300)
plt.show()

# Record elapsed time
elapsed_time = time.time() - start_time

print((elapsed_time/3600), 'hours' )
```

The code for [2] (b) has been shown below.

```
import networkx as nx
import random
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
import time
from mpl_toolkits.mplot3d import Axes3D

# Count time
start_time = time.time()

# Initialize all variables
n = int(50)
m = int(50)

array_long_range_probablitiy = []
array_transmission_probability = []
array_epi_size = []
array_epi_len = []
```

```
long_range_prob = np.arange(0.0, 1.001, 0.1)

for q in long_range_prob:

    # Initialize arrays for storing epidemic parameters, probabilities
    # and number of runs for each probability
    probabilities = np.arange(0.0, 1.001, 0.1)
    epidemic_size = []
    epidemic_length = []
    runs = np.arange(1.0, 20.01, 1.0)

    #graph = nx.grid_2d_graph(m, n, periodic=False, create_using=None)
    #nx.draw(graph)

    for pr in probabilities:

        total_infected_size = 0
        total_infected_length = 0

        for run in runs:

            graph = nx.grid_2d_graph(m, n, periodic=False, create_using=None)
            #graph = nx.planted_partition_graph(q, k, p_in, p_out, seed=42
            #                                   , directed=False)
            #nx.draw(graph)

            # Adding a random long range edge
            count = 0
            while (count != 1):
                # generate four random numbers between [0,m-1] and [0,n-1]
                rg_x1 = random.randint(0,n-1)
                rg_y1 = random.randint(0,m-1)
                rg_x2 = random.randint(0,n-1)
                rg_y2 = random.randint(0,m-1)
                # Check if a self-loop is being added
                if (rg_x1 != rg_x2 and rg_y1 != rg_y2):
                    # Check if an edge exists between these two numbers
                    if (graph.has_edge((rg_x1, rg_y1),
                                       (rg_x2, rg_y2)) == False):
                        count = 1
                        prob = random.uniform(0,1)
                        if (prob <= q):
                            graph.add_edge((rg_x1, rg_y1),
                                           (rg_x2, rg_y2))

            # Append all nodes to the list nodes
            nodes = []
            for i in graph.nodes():
                nodes.append(i)

            #print(random.randint(0,n))
            # Initialize variables
            infected = []
            inf1 = random.randint(0,n-1)
            inf2 = random.randint(0,m-1)
            infected.append((inf1, inf2))
            to_be_infected = [(n-1, m-1)]
            count = 0

            # while new_infected is not equal to infected
            # or no new nodes are infected
```

```
            while (to_be_infected != []):
                to_be_infected = []
                # Generate a random number between 0 and 1
                #prob = random.uniform(0,1)
                # Check if the generated number is less than or equal to p
                #if (prob <= p_in and prob <= p_out):
                    # Add all of the current node in 'infected' list's neighbours
                    # to 'infected' that are not in 'infected'
                    #print(infected)
                for i in infected:
                    #print('Already infected node:',i)
                    for j in graph.neighbors(i):
                        #print('Neighbors of infected node' ,i, 'is node' ,j)
                        if (j not in infected):
                            prob = random.uniform(0,1)
                            if (prob <= pr):
                                to_be_infected.append(j)
                                #print('New nodes added')
                            infected = list(set(infected) | set(to_be_infected))
                            #print('The nodes to be infected are',to_be_infected)
                            #print('The     new infected nodes are' ,infected)
                    #to_be_infected = []
                if (to_be_infected is not None):
                    count += 1

            total_infected_size += len(infected)
            total_infected_length += count

        # Find number of infected nodes and update array
        #epidemic_size.append(total_infected_size/len(runs))
        # Find number of steps and update array
        #epidemic_length.append(total_infected_length/len(runs))

        array_long_range_probablitiy.append(q) # Array for the x-axis
        array_transmission_probability.append(pr) # Array for the y-axis
        array_epi_size.append(total_infected_size/len(runs)) # Array for the z(1) axis
        array_epi_len.append(total_infected_length/len(runs)) # Array for the z(2) axis



# Modifications for plotting
array_epi_size_norm = []

for i in array_epi_size:
    array_epi_size_norm.append(i/(n*m))


# Generating the 3d Plots
mpl.rcParams['legend.fontsize'] = 10
fig = plt.figure()
#ax = fig.gca(projection='3d')
ax = fig.add_subplot(111, projection='3d')
#ax.plot(array_epsilon, array_probability, array_epi_size,
#        label='Epidemic Size')
ax.scatter(array_long_range_probablitiy, array_transmission_probability, array_epi_size_nor
ax.set_xlabel('Long range edge Prob. q')
ax.set_ylabel('Transmission Prob. p')
ax.set_zlabel('Average Epidemic Size <s>')
ax.legend()
plt.savefig('lre_s_vs_p_vs_q_p-10_q-10_runs-20', dpi = 300)
plt.show()
```

```
mpl.rcParams['legend.fontsize'] = 10
fig = plt.figure()
#ax = fig.gca(projection='3d')
ax = fig.add_subplot(111, projection='3d')
#ax.plot(array_epsilon, array_probability, array_epi_len,
#        label='Epidemic Length')
ax.scatter(array_long_range_probablitiy, array_transmission_probability, array_epi_len)
ax.set_xlabel('Long range edge Prob. q')
ax.set_ylabel('Transmission Prob. p')
ax.set_zlabel('Average Epidemic Length <l>')
ax.legend()
plt.savefig('lre_l_vs_p_vs_q_p-10_q-10_runs-20', dpi = 300)
plt.show()


mpl.rcParams['legend.fontsize'] = 10
fig = plt.figure()
#ax = fig.gca(projection='3d')
ax = fig.add_subplot(111, projection='3d')
#ax.plot(array_epsilon, array_probability, array_epi_size,
#        label='Epidemic Size')
ax.scatter(array_transmission_probability, array_long_range_probablitiy, array_epi_size_nor
ax.set_xlabel('Transmission Prob. p')
ax.set_ylabel('Long range edge Prob. q')
ax.set_zlabel('Average Epidemic Size <s>')
ax.legend()
plt.savefig('lre_s_vs_q_vs_p_p-10_q-10_runs-20', dpi = 300)
plt.show()


mpl.rcParams['legend.fontsize'] = 10
fig = plt.figure()
#ax = fig.gca(projection='3d')
ax = fig.add_subplot(111, projection='3d')
#ax.plot(array_epsilon, array_probability, array_epi_len,
#        label='Epidemic Length')
ax.scatter(array_transmission_probability, array_long_range_probablitiy, array_epi_len)
ax.set_xlabel('Transmission Prob. p')
ax.set_ylabel('Long range edge Prob. q')
ax.set_zlabel('Average Epidemic Length <l>')
ax.legend()
plt.savefig('lre_l_vs_q_vs_p_p-10_q-10_runs-20', dpi = 300)
plt.show()


# Record elapsed time
elapsed_time = time.time() - start_time

print((elapsed_time/3600), 'hours' )
```