Donovan Guelde
CSCI 5352
PS4

1. a. Given: $\epsilon = c_{in} - c_{out},\ 2c = c_{in} + c_{out},\ p_{out} = \frac{c_{out}}{n},\ p_{in} = \frac{c_{in}}{n}$

$c_{in} = \epsilon + c_{out} = 2c - c_{out}$

$c_{out} = \frac{2c-\epsilon}{2}$

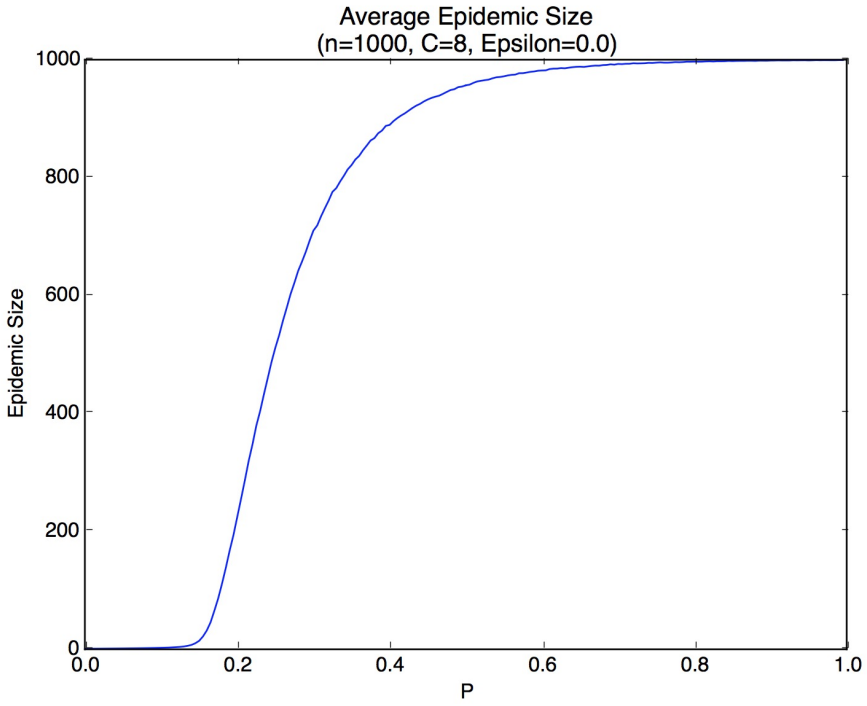
$c_{out} = c_{in} - \epsilon = 2c - c_{in}$

$c_{in} = \frac{2c+\epsilon}{2}$

$p_{in} = \frac{2c+\epsilon}{2n}$, and $p_{out} = \frac{2c-\epsilon}{2n}$

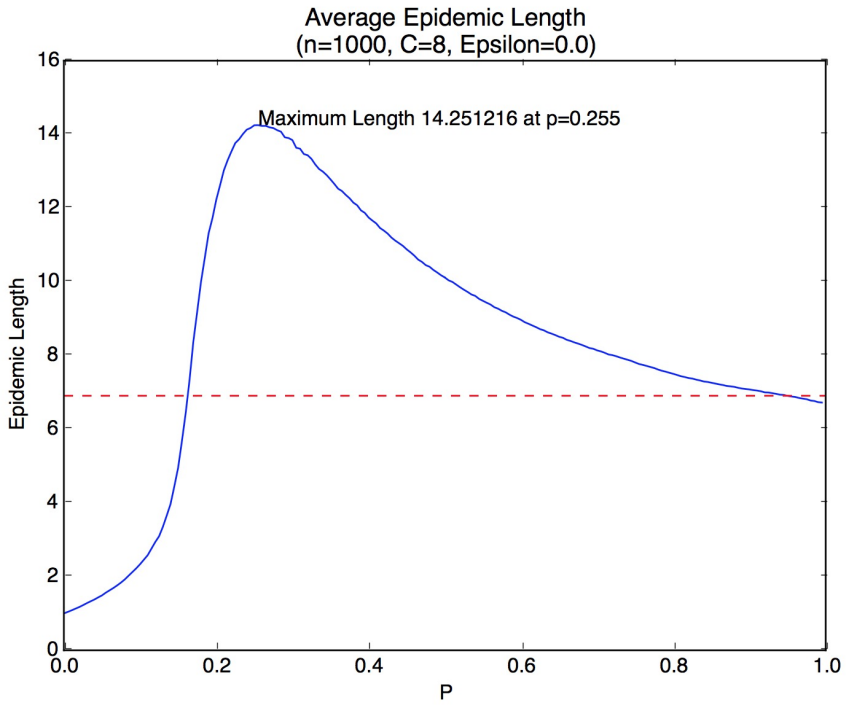
1. b. To study the simple SI model as discussed in class, the networkx library function planted_partition_graph() was used to generate planted partition networks, where p_{in} and p_{out} were derived from ϵ. To determine average length and size of epidemics, ϵ was fixed at 0, and p was raised from 0.0 to 1.0 in increments of 0.01. For each value of p, 200 graphs were generated via the planted_partition_graph() function, and 200 epidemic simulations were performed on each graph. This resulted in relatively smooth plots for average length and average size.

Plotting average epidemic size as a function of p gives the following results:



The epidemic size plot shows three distinct regions. First, there is a nearly flat area

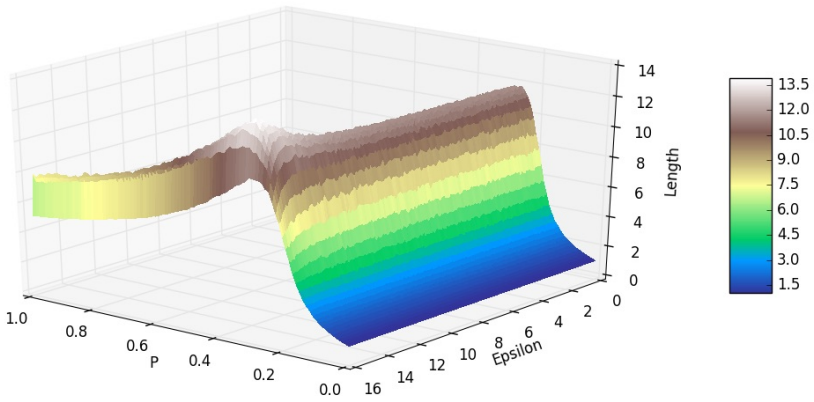
where epidemic size is very close to 1. This correlates to low p values, from 0.0 to approximately 0.17. This is expected, as at p=0.0, it is impossible for the epidemic to spread past the 'patient zero' node. As p increases up to 0.17, the low probability of infection makes the spread of the epidemic very unlikely past a few additional nodes past patient zero. As each node has mean degree 8, the expected number of infections caused by patient zero is $8p = 1.36$ at p = 0.17. The infection will spread by just a few nodes at such low p values. As p increases past 0.18, a pattern analogous to the random graph phase change can be seen. As the number of expected infections per contagious node grows larger, the epidemic size increases very rapidly. In this region of the plot, each infected node infects, on average, more than one neighboring nodes (ignoring nodes already infected). Due to the specified value of ϵ, the number of a node's edges that are internal to its group is, on average, the same as the number of edges connecting it to the other group. This allows the infection to spread easily between the groups, and makes it possible for the infection to reach all nodes without structural hinderance.



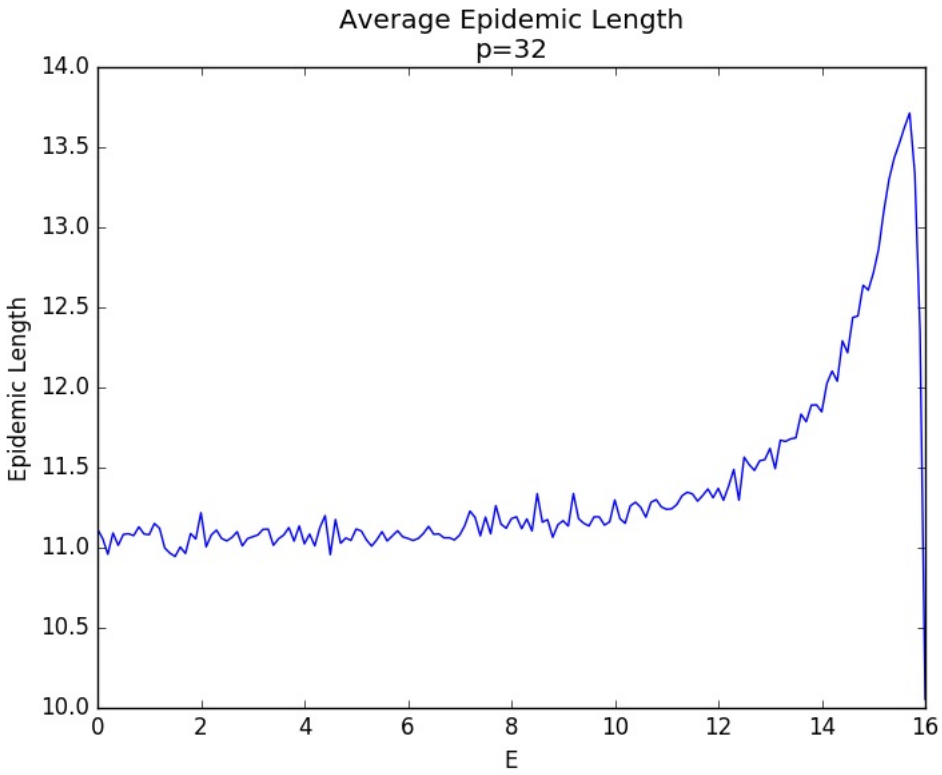
Epidemic length also grows slowly at low p values, a reflection of the low epidemic size. Since we have set up the epidemic simulation in such a way that a node is only contagious for a single time t, there is no opportunity to infect other nodes at time $t + 1$ and beyond. This limited window of infection, combined with a low probability of infection, means that patient zero may occasionally infect a neighbor node, which, with low probability, may infect a neighbor in turn. In other words, at low probability of infection, the epidemic

quickly dies out. Average epidemic length does not surpass 2 until p = 0.17, where average size was found to be 2.02. Once p reaches the critical point seen in the size plot, epidemic length quickly increases. When the number of expected infections caused by each newly infected node is low, but approaches 2, epidemic length increases dramatically. Interestingly, p=0.255 gives the maximum epidemic length, while this corresponds to a relatively low epidemic size of approximately 400. At this p value, the expected number of new infections per node infected at time (t-1) is 7p = 1.785 (8 edges - 1 edge for incoming infection, ignoring the possibility that another neighbor is already infected). This means that the epidemic is spreading in a mostly linear, rather than tree-like, fashion, allowing length to increase while size is still well below n. As p grows and each infected node is expected to produce more infections among its neighbors, the length decreases, approaching log(n), reflecting a branching epidemic spread. As p approaches 1, nearly every neighbor of an infected node will become infected itself, and the epidemic length approaches the diameter of the graph.

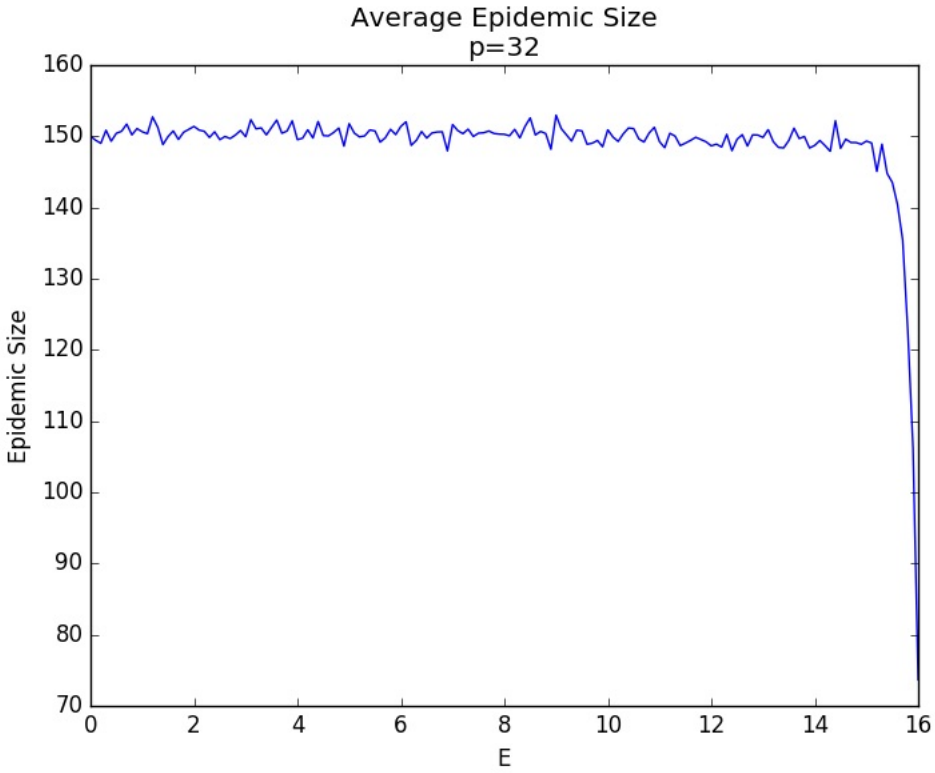
1. c. To examine how the behavior of the epidemic is affected by the large-scale structure of the network, planted_partition_graph() was used to create planned partition graphs, and these graphs were used in simulations for ϵ values of 0 to 16, in steps of 0.1, and p values of 0 to 1, in steps of 0.01. 100 graphs were generated for each ϵ/p value pair, and 100 simulations ere conducted per graph. Plotting ϵ,p, and epidemic length as a surface plot gives very clear indication of the effect of community structure on epidemic behavior. There is a high peak in length when ϵ is between 15 and 16, and p is approximately 0.3. Focussing specifically in this area, maximum epidemic length was found at p=.32, as ϵ approaches 16.



Holding p fixed at 0.32, and varying ϵ from 0 to 16 gives the following results for epidemic length:



At such high ϵ values, there are few edges connecting the two groups. As ϵ gets larger, edges between groups decreases. With just a few connecting edges, yet a high p value, the group where the epidemic did not originate acts to extend the effective network diameter, as far as the epidemic is concerned. Rather than a diameter of log(n), the epidemic experiences a diameter of 2 log (n/2). The effect of this diameter extension would vary randomly depending on what time t the epidemic spread reaches one of the few edges connecting the groups.



For this same p value, epidemic size shows a steep drop after the maximum point observed in the length graph above. This is caused by the separation of the two groups as p_{in} nears 0.0. With no way to reach the $\frac{n}{2}$ nodes in the other component, epidemic size is limited to $\frac{n}{2}$. The relatively low p value of .32 acts to keep maximum infection size below this level, however.

```
#Donovan Guelde
#csci 5352 PS4



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

"""
main() consists of 3 nested loops for the 'easter egg hunt',
outer loop is an iteration over a range of epsilon values
middle loop is an iteration over p values
inner loop is a loop over a specific graph instance
"""
def infectThePopulation(neighborMatrix,p,N):
        #print "\n"
        random.seed(time.time())
        susceptible = np.ones(N) #all susceptible
        contageous=np.zeros(N) #no contageous
        infected=np.zeros(N) #no infected
        patientZero=int(N*random.random())
        susceptible[patientZero]=0
        infected[patientZero]=1
        contageous[patientZero]=1
        newInfection=1
        t=0
        while (newInfection==1): #while disease spreads (if doesn't spread, then no new contageous nodes to check)
                newInfection=0
                newInfections=[]
                spreaders = np.where(contageous==1)
                for person in spreaders[0]:
                        for victim in neighborMatrix[person]:
                                if susceptible[victim]==1:
                                        immunity=random.random()
                                        if immunity<p:
                                                newInfection=1
                                                newInfections.append(victim)
                        contageous[person]=0 #not contageous any more
                        for victim2 in newInfections:
                                infected[victim2]=1
                                contageous[victim2]=1
                                susceptible[victim2]=0
                                newInfections.remove(victim2)
                t=t+1
        size = np.count_nonzero(infected)
        return (size,t)

```

```
52  def plotResults(epidemicSize,epidemicLength,pValues,N,E,C):
53          plt.plot(pValues,epidemicSize)
54          plt.title('Average Epidemic Size\n(n={}, C={}, Epsilon={})'.format(N,int(C),E))
55          plt.xlabel('P')
56          plt.ylabel('Epidemic Size')
57          plt.savefig('./q1c/E{}size.png'.format(E))
58          plt.close()
59          tMax=0
60          maxIndex=0
61          counter=0.
62          for item in epidemicLength:
63                  if item>tMax:
64                          tMax=item
65                          maxIndex=counter
66                  counter+=1
67          maxIndex = float(maxIndex)/100.
68          plt.plot(pValues,epidemicLength)
69          plt.annotate('Maximum Length {} at p={}'.format(tMax,maxIndex),xy=(maxIndex,tMax))
70          plt.title('Average Epidemic Length\n(n={}, C={}, Epsilon={})'.format(N,int(C),E))
71          plt.xlabel('P')
72          plt.ylabel('Epidemic Length')
73
74          plt.axhline(y=math.log(N),xmin=0,xmax=1,color='r',ls='dashed')
75          plt.savefig('./q1c/{}Elength.png'.format(E))
76          plt.close()
77
78  def main():
79          #important variables
80          ITERATIONSONP=250 # number of iterations for each p value
81          ITERATIONSPERGRAPH=250 #iterations on each graph
82          Emin=15.0 #range of epsilon values to consider
83          Emax=16.0
84          ESTEP=.1
85          PMIN=0
86          PMAX=1#range of p values to consider
87          PSTEP=.1
88          C=8
89          N=200
90          L=2
91          size=((PMAX-PMIN)/PSTEP)+1
92          epidemicSize=np.zeros((size)) #hold results from outer loop
93          epidemicLength=np.zeros((size))
94          pValues=np.zeros((size))
95          E=Emin
96          while (E < Emax): #iterate on a range of epsilon values
97                  c=float(C)
98                  k=int(N/L) #k=vertices per group
99                  c_in=2*C+E
100                 c_out=2*C-E
101                 p_in=(.5*c_in)/N
102                 p_out=(.5*c_out)/N
103                 p=PMIN
104                 counter=0
105                 while (p<PMAX): #next inner loop, over p values
106                         start = time.time()
107                         pValues[counter]=p #use this p on multiple generated graphs (multiple times)
108                         sizeArray=np.zeros((ITERATIONSONP)) #store size results for runs on multiple graphs
109                         lengthArray=np.zeros((ITERATIONSONP)) #store length results for runs on multiple graphs
```

```
110                         for index in range(0,ITERATIONSONP):
111                                 #store size results for multiple infections on one graph
112                                 graphInfectionSize=np.zeros((ITERATIONSPERGRAPH))
113                                 #store length results for multiple infections on one graph
114                                 graphInfectionLength=np.zeros((ITERATIONSPERGRAPH))
115                                 g = nx.planted_partition_graph(L,k,p_in,p_out) #generate planted partition graph
116                                 AssociationMatrix = nx.to_numpy_matrix(g)
117                                 neighbors=[]
118                                 counter2=0
119                                 for item in g:
120                                         neighbors.append(nx.neighbors(g,counter2))
121                                         counter2+=1
122                                 neighborMatrix = np.asarray(neighbors)
123                                 for index2 in range(0,ITERATIONSPERGRAPH): #iterate on a graph
124                                         graphInfectionSize[index2],graphInfectionLength[index2]=
125                                                 infectThePopulation(neighborMatrix,p,N) #run scenario on graph
126                                 sizeArray[index]=np.sum(graphInfectionSize)/ITERATIONSPERGRAPH #average size of infection
127
128                                 #average length from given graph
129                                 lengthArray[index]=(np.sum(graphInfectionLength))/ITERATIONSPERGRAPH
130                         #average infection length from multiple graphs for given value of p
131                         epidemicLength[counter]=(np.sum(lengthArray))/ITERATIONSONP
132                         #average infection size from multiple graphs for given value of p
133                         epidemicSize[counter]=np.sum(sizeArray)/ITERATIONSONP
134                         p+=PSTEP #end of p-loop
135                         counter+=1
136                         print "E",E,"p",p,time.time()-start
137                 np.savetxt('./q1c/E{}length.txt'.format(E),epidemicLength)
138                 np.savetxt('./q1c/E{}size.txt'.format(E),epidemicSize)
139                 plotResults(epidemicSize,epidemicLength,pValues,N,E,C)
140                 E+=ESTEP #end of e-loop
141
142 main()
```