Analysis of Misinformation Containment Models in Social Networks

Sohom Chatterjee, UIN: 729007535
Rohit Dube, UIN: 930001592

May 2020

1 Scope of Study

Due to the explosion in the use of the internet over the past few years there has been a huge growth in the multitude of the interaction through which people meet, share and spread the ideas. There are models used in the domain of epidemiology which mimic the spread of a disease by direct person-to-person contact [1]. Some notable ones which are used to model social network rumor spreading are the SIS [2] and SIR [3] models. An improved version of SIS model is used to study the person-to-person transfer of information within an online social network.

In this study, we evaluate and simulate rumor-spreading models and check the strategies that can be employed effectively to contain the spread of such rumors. A detailed study of an improved SIS rumor spreading model [4] in the dynamically growing online Social Network is made. It considers the changing levels of user engagement and the rate at which users enter or leave the social network. The study of this improved SIS model in a dynamically growing social network setting has been primarily conducted by Dong and Huang [4]. Classic SIS model is based on a static system, while in reality social networks are a dynamic system where users keep registering and leaving continuously. Thus the impact of these dynamics should be considered within the model and the influence of the user activity should be taken into account.

The rumor diffusion process in such online social network (SN) is here represented by a stochastic system model where in every time step a rumor spreading user tries to pass on the information that they possess to the uninformed user. The nodes in the graph represent such users and the edges represent connection between them. Our network consists of susceptible (healthy) and infected users and there stance might change at each time step due to the infection and recovery process. We have tried to mimic the setting of the social networks like Facebook and Twitter.

2 Our Contribution: Anti-Rumor User Introduction

A rumor is usually recognized after the certain portion of population is infected with it. Thus the efforts to contain rumor starts only after some users are infected.The strategy is to reduce such spread of misinformation by introducing positive information nodes which are immune to the rumor diffusion process described by the improved SIS model. These users will never change their anti-rumor opinion. Once they pass on this anti-rumor information to other users, even they will be immune to rumors. This diffusion process and the introduction of positive information nodes is modeled on a dynamically growing graph which closely represents the real SN process. Convergence is reached when there is no infected node left in the system.

Since there is a cost associated with the introduction of the users in the social network responsible to spread positive information, a cap or an upper limit of such users is maintained. We investigate a few common strategies behind selection of these initial set of anti-rumor nodes, as well as propose a new and better strategy for the same, that reduces the total convergence time of the rumor-spreading dynamics. Two common selection procedures for the anti-rumor nodes are,

- Random Selection
- Max Degree Node Selection

A comparative study is made between these models for introduction of positive information nodes and the one proposed to check which reduces the rumor in SN in minimum time steps.

3 Data Set Description

We have taken a network data set from the DIMACS Facebook Networks Repository. In this network the nodes represent users and an edge between two nodes represent a friendship relation between the corresponding users. There are 962 nodes (users) and 18812 edges (friendship ties) in this network. The network is unweighted and undirected.

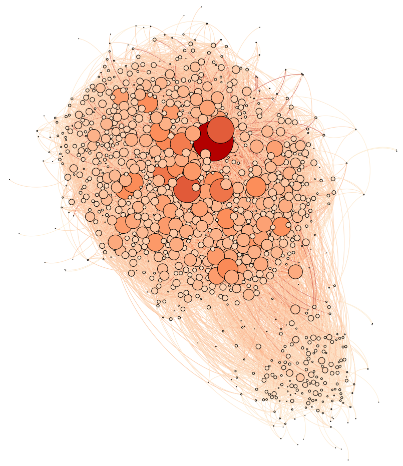

Figure 1: Facebook Friendship Network

As shown in Fig. 2, we generate the degree distribution of our network using *Gephi* software. We see that the average degree of our network is 39.11.

As is commonly seen in social networks and as is evident from the degree distribution, our network has a scale free structure or a power law degree distribution.

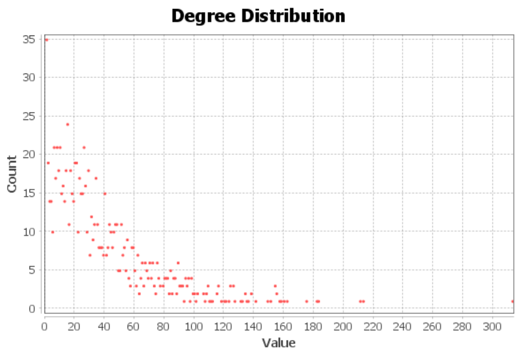

Figure 2: Degree Distribution of Facebook Friendship Network

We can also get an idea about the topology of our network using other parameters. Since this is a social network another parameter of interest is the clustering coefficient which is shown in Fig. 3. We see that that the average clustering coefficient is 0.330 which is not too high.

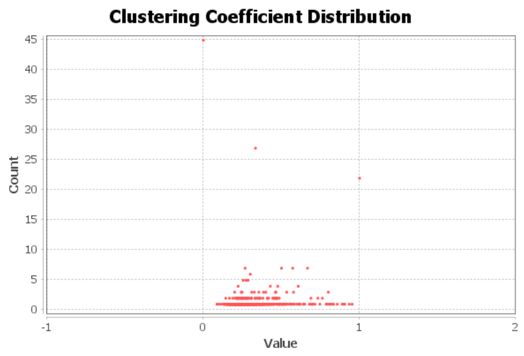

Figure 3: Clustering Coefficient Distribution of Facebook Friendship Network

The Eigenvector Centrality is a centrality measure that assigns the importance of a node to the importance of other nodes to which it is connected. Fig.4 shows the Eigenvector centrality distribution of our network.

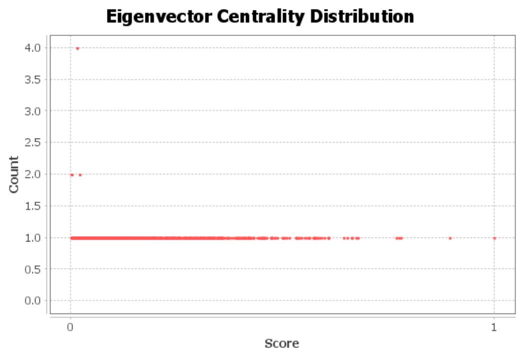

Figure 4: Eigenvector Centrality Distribution of Facebook Friendship Network

The above parameters give us an insight on the topological characteristics of our network. This data will be used to construct our dynamically growing scale-free network and subsequently the rumor evolution map.

4 Model

In the setting of the improved SIS model discussed in [4] we have considered the network node into two categories, namely health nodes S which is the number of users who are not under any influence of the rumor or misinformation, while the transmission nodes I are those users who are under the influence of the rumor and have the ability to spread it to the health nodes, so are involved in the propagation of the message. These two types, $S(t)$ and $I(t)$ form the entire population of the social network $N(t)$ and the individuals can contract disease only from their immediate neighbors, where the t refers to the time step. The users within the social network are thus healthy(susceptible), $x_i(t) = 0$ or infected, $x_i(t) = 1$. We normalize these variables and thus take the proportion of S and T in the total population N.

Thus, $N(t) = S(t) + I(t)$ and $0 < S(t), I(t) < 1$.

Consider b as the net growth rate of the network, β as the infection rate at which the rumor spreads from healthy nodes to the healthy nodes and σ as the cure rate at which the infected nodes recover to being a healthy node.

We expand this network using the algorithm proposed by Barabasi and Laszlo [5], which conserves the scale-free properties of the network. Essentially a new node which is added has the probability of being connected to an existing node i by the probability $p(i) = \frac{degree(i)}{\sum_{j \in V} degree(j)}$, where V is the set of nodes in the network.

Initially we infect only one node with our rumor message. Since the average degree of the network is 39.11, we choose a node with degree 40 as the first node with rumor.

After this, we update the nodes or propagate the rumor based on the ideas of the SIS model. Because the infected nodes would pass on rumor message to the healthy nodes during their interaction at some probability, and the infected nodes also may recover from the influence of rumor message at some time rate, we have the following propagation rule,

$$S + I \xrightarrow{\beta} 2I, \qquad I \xrightarrow{\sigma} S.$$

After a certain portion of population is infected, anti rumor users are activated in the network. These anti rumors node are a part of S for which the probability β is always zero i.e, they will always remain healthy and these nodes will spread the positive information with the rate γ. Since activating these nodes initially would be costly, we assign ourselves a budget which is the number of anti-rumor nodes we can initially activate. Let this budget be D. Having this constant budget, we wish to check which strategy of selection of these nodes lead to the fastest convergence and removal of rumor from the network.

Apart from the usual strategies to select anti-rumor nodes, which is selecting the nodes from healthy nodes randomly, and selecting the high degree nodes, we find a fundamental issue in these strategies. In both these cases, there is a possibility of the selected nodes being connected to each other. In this case, we are losing out on possible spread of the anti-rumor information. So it would be beneficial to us if none of these initially selected nodes are connected to each other. Using this logic, we propose a third strategy for initial node selection.

4.1 Independent Set Maximum Degree (ISMD) Strategy

Since we would be benefited by selection of high degree nodes which are not connected to each other, we select the initial anti-rumor nodes using a two step procedure.

First, we select the set of candidate users to turn into anti-rumor nodes. They are the set of healthy users at that point of time when the ratio of infected people have crossed a certain threshold.

On the induced subgraph of this candidate set of nodes, we solve the maximum independent set problem.
The maximum independent set problem can be formally stated as follows,

The maximum independent set problem: Given a simple undirected graph $G = (V, E)$, a *maximum independent set* is a maximum-cardinality subset of vertices which are pairwise independent, that is, no two nodes share any edges between them.

We solve this maximum independent set problem using the commercial solver Gurobi 9.0 using an Integer Programming Formulation. The integer programming formulation for the problem can be written as:

Let $\mid V \mid = n$. Let x_i be a binary variable, which assumes value 1 if vertex i is a part of the independent S, and 0 otherwise. Our decision variables are these x_i, which decide which vertices are going to be a part of the independent set, where $i \in V$. Hence, the optimization problem is,

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} x_i \\ \text{subject to} \quad & x_u + x_v \leq 1 \qquad \forall (u,v) \in E \\ & x_i \in \{0,1\} \qquad \forall i \in V \end{aligned}$$

This will give us a set of nodes that are possible to be converted into anti-rumor nodes, and are not connected to each other.

Next we sort these nodes according to their degree and select the top few nodes, as per our budget. This will ensure maximum utilization of the edges and maximum anti-rumor spread from these initial nodes.

A comparison is made between the models with no anti rumor nodes and these three models.

Our convergence criteria for this model is when all the nodes in this model are completely healthy and there is no chance of further propagation of infection, ie. $S = 1$ and $I = 0$.

5 Simulation

The following parameter values were used throughout in the analysis.

Parameters	**Values**
birthrate, b	0.3
infection rate, β	0.35
recovery rate, σ	0.2
anti rumor spread rate, γ	0.1
Budget of anti rumor nodes, D	10

Table 1: Parameter Values for model

5.1 With Zero Anti Rumor nodes

In this model we do not introduce any anti rumor nodes and the recovery is based entirely on the improved SIS model [4]. Based on our algorithm, we add new nodes and edges to a simple Facebook network with 962 nodes using the *Networkx* library, and construct an online social network with changing number of nodes.

Fig. 5 shows the time versus Infection proportion curve for the rumor spreading in the Facebook network with $\beta = 0.35$. From Fig. 5 we can see that, in Facebook network, the trend change of the infection rates match with the simulations done in [4]. The first peak of the curve of the infection node density is around $t = 20$.

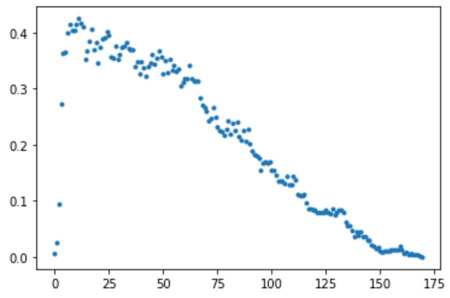

Figure 5: Infection Node Proportion for Dynamically Growing Network, β=0.35

In order to understand the impact of this infection rate on rumor-propagation we plot infection proportion plots for varying infection rates. Fig. 6 shows the curves for 3 different values of β. Lowering the infection rate leads to a smaller peak and faster convergence. All of these simulations are completely consistent with previously established results.

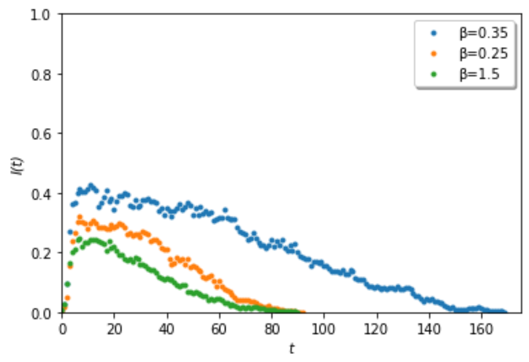

Figure 6: Infection Node Proportion for Varying Infection Rates

5.2 Anti Rumor nodes introduced Randomly

Introdcution of Anti-Rumor nodes drastically changed the rumor growth dynamics. From Fig. 7, we see that for the same infection rate 0.35, the curve converges in 36 time steps as compared to 175 time steps required in the absence of Anti-Rumor nodes. This is considering an anti-rumor spread rate as low as 0.1. Thus we see that even the introduction of anti-rumor nodes randomly really affect the dynamics.

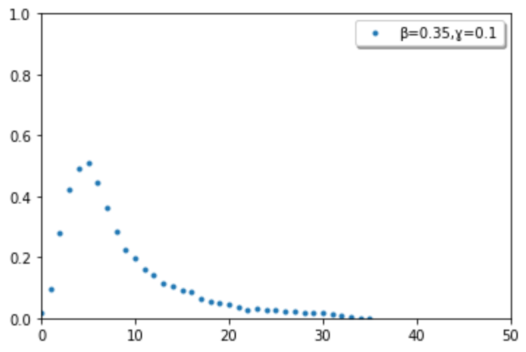

Figure 7: Introduction of Anti-Rumor Nodes, Random Selection

5.3 Anti Rumor nodes introduced at Maximum degree nodes

Next we use a different and more pragmatic choice of selecting anti-rumor nodes. At the point of introduction of anti-rumor nodes, ie. when the infection proportion of the network crosses a particular threshold, we extract the candidate set which is just the set of all healthy nodes at that point of time and then we sort them according to their degree. We then select the highest number of nodes as per our budget and select them to be the anti-rumor nodes. From Fig. 8, we see that for the same infection rate 0.35 and the same anti-rumor spread rate 0.1, the Max Degree Selection clearly outperforms the random selection. Convergence is reached much earlier, at around 25 time steps. Hence this seems to be a better selection scheme.

5.4 Anti Rumor nodes introduced Using ISMD Algorithm

In the third instance we use the proposed algorithm ISMD. This has a clear advantage over the others as there is no chance of the selected anti-rumor nodes to be connected to each other, which ensures maximum utilization of their edges and thereby anti-rumor spreading. We can see from Fig. 9 that the dynamics converge even faster and has a steeper descent rate than the previous two. This essentially validates the better selection of intial nodes.

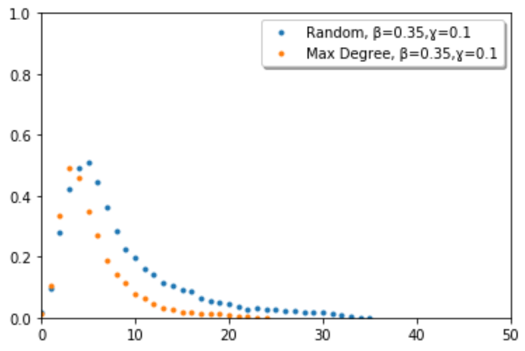

Figure 8: Introduction of Anti-Rumor Nodes, Random and Max Degree Selection

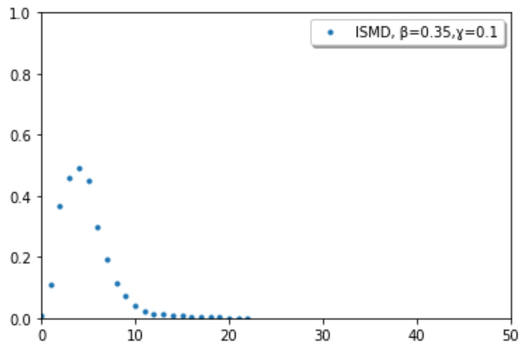

Figure 9: Introduction of Anti-Rumor Nodes, ISMD Selection

6 Conclusion

Fig. 10 provides a comparison between the 3 selection schemes and it can be clearly seen that the ISMD Selection scheme dominates the other two consistently. The average convergence time is much lower than the other two heuristics and the fall in rumor nodes is also steeper as can be seen from the curve.

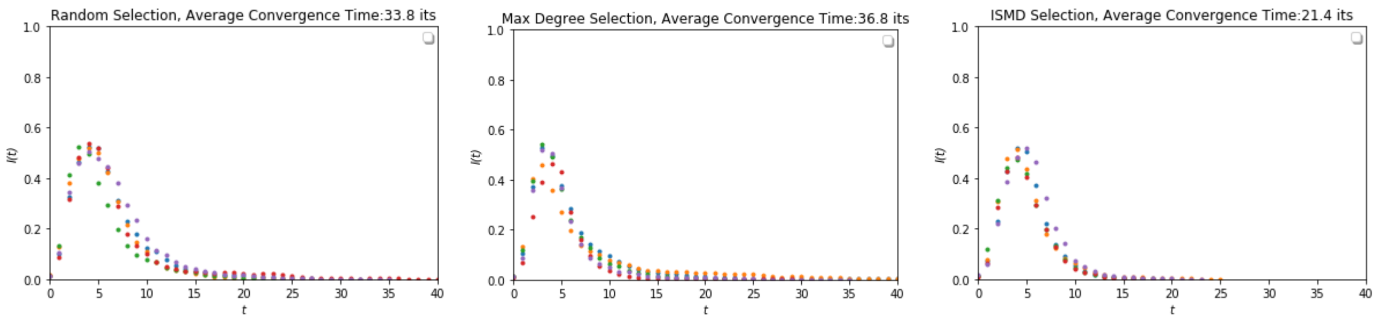

Figure 10: Comparison of Selection Schemes

Our study here provides a comparison between three anti-rumor propagation models on a dynamically growing social network, on top of the SIS rumor spread model. The results, especially for the proposed ISMD scheme, provide interesting insights into how selection of appropriate nodes can drastically change the convergence time of the network dynamics. Computing similar simulations on larger networks might provide even better reductions in convergence time.

References

[1] Herbert W Hethcote. The mathematics of infectious diseases. *SIAM review*, 42(4):599–653, 2000.

[2] William Ogilvy Kermack and Anderson G McKendrick. Contributions to the mathematical theory of epidemics. ii.—the problem of endemicity. *Proceedings of the Royal Society of London. Series A, containing papers of a mathematical and physical character*, 138(834):55–83, 1932.

[3] William Ogilvy Kermack and Anderson G McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character*, 115(772):700–721, 1927.

[4] Suyalatu Dong and Yong-Chang Huang. A class of rumor spreading models with population dynamics. *Communications in Theoretical Physics*, 70(6):795, 2018.

[5] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

Appendices

```
 # -*- coding: utf-8 -*-
"""
Created on April '20'

@author: Sohom Chatterjee
"""
#%% 1. IMPORTING LIBRARIES
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
from gurobipy import *
from scipy.io import mmread
import os, glob, re
import sys
import string
import math
import random
from tqdm import tqdm
from scipy import optimize
#%% 2. FILE READ (FACEBOOK NETWORK)
##Read .mtx file as nxGraph
if __name__ == "__main__":
    # load mtx files
    path ='C:/Users/Sohom/Desktop/ISEN 689/'
    # os.path.dirname(os.path.realpath(__file__)) + '/'''
    data_path = path + 'Data/'
    # data_path = '/Users/rw422/Documents/Twitter/AV Events Keywords Only (csv)''

    filename = 'socfb-Reed98'
    # glob.glob(data_path + '*.csv')[:1]''

    df = mmread(data_path + filename+'.mtx')

Gr = nx.from_scipy_sparse_matrix(df)
#%% 3. GEPHI VISUALIZATION
if __name__ == "__main__":

    gephi_path = path + 'Gephi/'

    nx.write_gexf(Gr, gephi_path + filename + '.gexf')
#%% 4. FUNCTIONS
```

```

def ind_set(netw):
    #G is the network
    G=nx.complement(netw)
    f=Model("Max Clique")
    f.Params.OutputFlag=0
    x=[]
    for node in G.nodes():
        x.append(f.addVar(name="x"+str(node), vtype=GRB.INTEGER))

    # define objective
    f.setObjective(sum(x),sense=GRB.MAXIMIZE)

    #define constraints
    G_prime=nx.complement(G)
    for edge in G_prime.edges():
        u=edge[0]
        v=edge[1]
        f.addConstr(x[u]+x[v]<=1)

    # add valid inequalities

    # optimise
    f.optimize()
    l=[]
    # general clique set
    for v in f.getVars():
        if (v.x==1):
            l.append(int(v.varName[1:len(v.varName)]))
    return l

def avg_degree(G):
     a=0
     for node in G.nodes():
         a = a+ G.degree[node]
     return(a/len(G.nodes()))

def total_degree(G):
    td=0
    for node in  G.nodes():
        td = td + G.degree(node)
    return td

def randflip():
    return np.random.uniform(low=0.0, high=1.0, size=None)

def expand_scale_free(G):
    if randflip()<=birth_rate:
        G.add_node(len(G.nodes())) #ONLY CREATING NODE
        for node in range(0,len(G.nodes)-1): #CREATING EDGES
            if randflip()<=G.degree(node)/total_degree(G):
                G.add_edge(node,len(G.nodes())-1)

def assign_initial_node(G):
    for node in G.nodes():
        if G.degree[node] ==math.ceil(avg_degree(G)):
            status[node]=1
            break

def RandomSelect(cand):
    nodes=random.sample(cand,budget)
    return nodes

def MaxDSelect(cand):
    Graph=G.subgraph(cand)
    a=sorted(Graph.degree, key=lambda x: x[1], reverse=True)
    #return first "budget" elements
    list=[]
    for ele in a[0:budget]:
        list.append(ele[0])
    return list


def ISMaxDSelect(cand):
    Grap=G.subgraph(cand)
    i=0
    tupl=[]
    SGr=nx.Graph()
    for node in Grap.nodes():
        tupl.append((node,i))
        SGr.add_node(i)
        i=i+1
    for edge in Grap.edges():
        u=edge[0]
        v=edge[1]
        for c in tupl:
            if u==c[0]:
               u_pr=c[1]
               break
        for d in tupl:
            if v==d[0]:
                v_pr=d[1]
                break
        SGr.add_edge(u_pr,v_pr)
    ind= ind_set(SGr)
    lst=[]
    for z in ind:
        for y in tupl:
            if z==y[1]:
                lst.append(y[0])
                break
    return MaxDSelect(lst)

def select_antirumor_nodes(cand):
    # return RandomSelect(cand)
    # return MaxDSelect(cand)
    return ISMaxDSelect(cand)
#%% 5. CONSTANTS
birth_rate=0.3
inf_rate=0.38
rec_rate=0.2
ar_rate=0.1
budget=10
#%% 6. LISTS AND STARTING CONDITIONS
G=Gr.copy() #We keep a copy of the graph to keep the original undistorted

#Status List (1:Infected, 0:Healthy)
status= np.zeros((len(G.nodes()),), dtype=int)
#Select initial infected node=
assign_initial_node(G)
# #infect intial node
# status[assign_initial_node(G)]=1
```

```

#status[3]=1
#Lists
infC=[1] #Count of Total Infected People, per time stage
infR=[] #Ratio of Total Infected People, per time stage
susC=[len(G.nodes())-1] #Count of Total Healthy People, per time stage
susR=[] #Ratio of Total Healthy People, per time stage
#%%  7. BASIC ALGORITHM
count=0
flag=0
inf=1
cand=[]
while inf>0 and count<=500:
    ##Update system as per logic
    InfectedList=[]
    ARList=[]
    #Update ill nodes
    for edge in G.edges():
        u=edge[0]
        v=edge[1]
        if status[u]==1 and status[v]==0 and randflip() <= inf_rate:
            InfectedList.append(v)
    for ele in range(0,len(InfectedList)):
        status[InfectedList[ele]]=1

    #Update healthy nodes
    for ele in range(0,len(status)):
        if status[ele]==1 and randflip()<=rec_rate:
           status[ele]=0

    #if infected ratio is some proportion of total population,
    if (inf/len(G.nodes()))>=0.4 and flag==0:
        cand=[]
        for node in G.nodes():
            if status[node]==0:
                cand.append(node)
        act_nodes=select_antirumor_nodes(cand)
        for node in act_nodes:
            status[node]=2
        flag=1
  #select some nodes using some logic as anti rumor nodes
           #1. Randomly
           #2. Max Degree
           #3. greedyAlgorithm



    #Update Anti Rumor Nodes
    for edge in G.edges():
        u=edge[0]
        v=edge[1]
        if status[u]==2 and randflip() <= ar_rate:
            ARList.append(v)
    for ele in range(0,len(ARList)):
        status[ARList[ele]]=2

    #Count and add to
    inf=0
    for ele in range(0,len(status)):
        if status[ele]==1:
            inf=inf+1
    # i=sum(status)
    infC.append(inf)
    susC.append(len(G.nodes()) - inf)
    infR.append(inf/len(G.nodes()))
    # #Code to expand network (scale-free)
    expand_scale_free(G)
    # #status append for new node
    status=np.append(status,0)
    count=count+1
    print('Iteration:'+ str(count))
    print('Infected Count:' + str(inf))
    print('Infected Ratio:' + str(inf/len(G.nodes())))
    print('-----------------------------------------------------')
    if inf==0 or count==500:
        break
```

Listing 1: Simulation Code