# Networks & Crowds: Spreading Bad Information Process

**Robert Horton**
UCCS
1420 Austin Bluffs Pkwy,
Colorado Springs, Colorado 80918

## Abstract

This paper briefy studies an instance of a graph given for the project whee the network is studied along with the influences that a node with a set amount of opinion and information can affect the whole network in different ways through connections to different original existent nodes.

## Introduction

In class, we discussed the Friedkin-Johnsen model of opinion dynamics, where opinions spread on a network but people always remember their initial opinions. To refresh our memory, the model works like this: We're given a society with n individuals connected on a social influence network described by an $n \times n$ row-stochastic, directed, weighted adjacency matrix A. An edge $(i,\ j)$ (arrow pointing from i to j with weight aij means that node i gives a fraction aij of its attention to node j. Each node i has a susceptibility $\lambda i \in [0,\ 1]$, and we collect these susceptibilities into an $n \times n$ diagonal matrix $\Lambda = (\lambda_0,\ \lambda_1,\ .\ .\ .\ ,\ \lambda_{n-1})_{diag}$. Given an initial vector of opinions $x(0) = (x_0(0),\ x_1(0),\ .\ .\ .\ ,\ x_{n-1}(0))^T$ , the opinion at time $t$ is given recursively by

$$x(t) = \Lambda\ A\ x(t-1) + (I - \Lambda)\ x(0),$$

where $I$ is the $n \times n$ identity matrix.

In this project, all questions consider the following weighted directed graph, with susceptibilities of $\lambda = (0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 0.95)$.

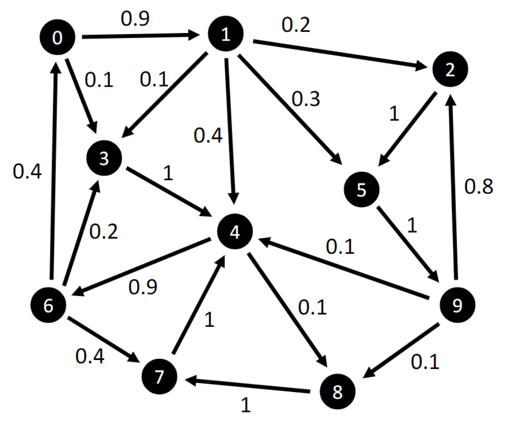


Figure 1.1

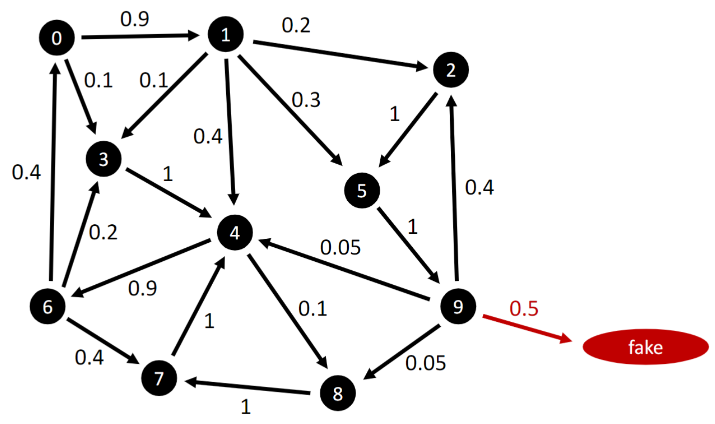


Figure 1.2

Consider the following model of misinformation: let $x(0) = (0,\ 0,\ .\ .\ .\ ,\ 0)^T$ so that everybody starts out at 0. Then, choose a node to influence (let's say our influencing node $i$), and connect i to a new "fake" node whose initial "opinion" is 1. To make this connect, cut the weights of each of $i$'s out-edges in half, and give the new edge a weight of 0.5. So the math works, the new fake node should have a single self-loop with weight 1. See Figures 1.1 and 1.2 to show an example of a edge being added to node 9 to the new "fake node". Because everybody starts out at 0, you can measure the effect of your misinformation simply by adding up everybody's opinion. We'll call this sum the propaganda value of your misinformation scheme, and denote it by $P(t) := \sum_{i=0}^{n-1} x_i(t)$ . An ineffective misinformation scheme would have $P(t)$ close to 0 for all time, but a very effective one would have $P(t)$ close to $n$.

## Implementation

Well to implement this first an object oriented approach was taken which was causing more over head and dependencies so a Jupyter Notebook was create to create the basic foundations for simpler to read and read code. The following is a description of this implemented code. First an adjacency matrix was built and hard codded into a python two dimensional array. This we then proceed to use other code provided and earlier handout for the class to print the network out and to show that the network it represented correctly by the adjacency matrix.

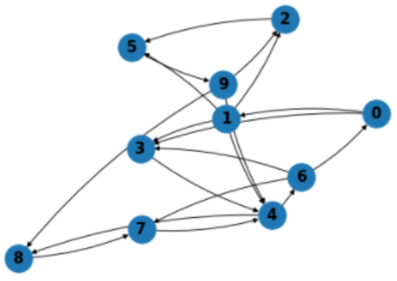


Figure 1.3

Although it might be hard to see the graph shown by Networkx that is used in a function provided to use, we can see that all the same nodes exist with the same existing edges as on the picture of the graph shown in Figure 1.1. Now to demonstrate that we can add nodes to this adjacency matrix we build two more function that handle building the matrix and then will print it you to show that it works in connected a fake node to node 9 as in Figure 1.2.

The first function built I called addNodeToNetwork() which took two parameters which are the network we built an adjacency matrix for and the original opinions of all the existing nodes at time step $t = 0$. This functions take the network, as a two dimensional list and builds new rows with 0's added to the end of them and setting them to a new two dimension list to continue to modify before returning. Next the two dimensional list will have a row of all zeros added at the bottom of the list. finally before returning the new network with an added node we set the new node to have a self loop and we also add a new opinion to the list or original opinions to represent the opinion of the new node which we will call the "fake node". We also add another value to the lambda values which can be any number since it does not listen to any other nodes. The list of opinions, lambda values, and new network are then passed back.

Then the addBadEdgeToNetwork() method was created to be called right after adding a new node. This function simply accepts three parameters to adjust the original network to having a edge between the fake node and what ever node is based as the 'node' argument. To do this, the function multiplies every existing edge by 0.5 and then sets each element do that amount specific to the column in that node's row of edges. Then a new edge of 0.5 is added between the fake node and the specified node. This newly adjust network is then passed back.

After using the draw_from_matrix() function provided to us we can show that the matrix does in fact add a new edge. We also checked the matrix was right by checking the console output of printing the matrix.

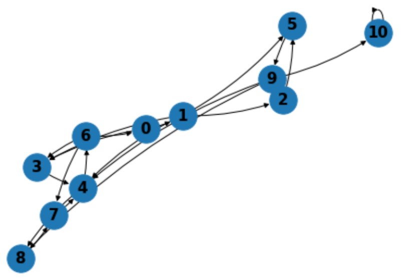


Figure 1.4

As we can see a new node has been added and the only added edge are the self loop to itslef and the new edge between node 9 and node 10, the fake node.

After verifying that this was working we used the friedkin_johnsen() provided to us through class handouts and modified it to print out a little bit more verbose information when setting its keyword argument, show_plot, is set to true. This function does what has already been discussed in class by essentially using the equation as described in the introduction. By preforming these different time steps of Friedkin-Johnsen model we were able to map the rate at which the information, or "opinion", of the "fake" spreads depending on the node that it is connected to. To perform each instance of when a different node could be connected to the "fake" node we make an abstract layer to call of the methods previously discussed so that this method or model can be called upon in a for loop of each possible neighbor that the fake node could have a connection from. Within this model a fresh new network of the original network is made sure to be passed in to this model to represent the only connection between the fake node and any other pre-existing node.
Along with executing every possible connection, a list was made to capture the overall propaganda value of the network at each time step. Another function

was written to show this value as well as the opinion of each node at each iterated time step through the model and were used to display the results through other implemented methods plotAllProagandaValOverTime() & plotAllProagandaValOverTime() to plot them individually between simulation runs and to call at the end of the for loop of all possible simulation runs to map all possible outcomes of $P(t)$ values specific to the node that the fake node was being connected from. Unit test were also written to make sure that basic functionality is maintained through out the completion and development of this project.

## Execution

To help mitigate any dependencies that may cause issues with running the program a requirments.txt file was provided which can be used with the below command to make sure that all the libraries used for this project are pre installed.

```
py -m pip install -r requirments.txt
```

Execution is set up to simply run the missinformationProjecy.py file with the commad

```
py project03.py
```

from the same working directory as this file. When running this file, without modifying it, it should save different graphs and tables to different folders within the included .zip file structure. If the user wishes to run the file without the print out of all the graphs as they run it they must go through and uncomment out the show() method with in the functions timestepsToPTable() and plotAllProagandaValOverTime(). This will still display the tables required of in both deliverables and will be turned in from previous runs along with capbability to execute and generate "new" tables. The overall console output has also been saved to the out tableOutput within the Outputs folder. If there any any other questions on the over head of the project executeable refer to the README.md for futher details and help.

## Analyisis

After personal execution with collected data and observations, I was then able to continue with providing the deliverables for the project. First off, here is a quick overview of each deliverable asked of for this project and what it consists of.

1. Implement the Friedkin-Johnsen model on the given graph.
   (a) Deliverable 1: Either the computer code for your implementation, or a complete description of the maths you used to prove the other deliverables.
2. If you want to influence opinions over the short-term, which node should you influence?
   (a) Deliverable 2: The graph has 10 nodes; create a table which shows $P(1)$ (the total opinion of society after 1 time step) for each node to be influenced. That is, compute $P(1)$ in the case that you influence node 0; then compute $P(1)$ in the case that you influence node 1, and so on. If you care most about $P(1)$ (society's opinions in the very short term), which node is best to influence? Which is worst? Write a few sentences explaining why you think this is the case.
3. If you want to influence opinions over the long-term, which node should you influence?
   (a) Deliverable 3: The graph has 10 nodes; create a table which shows $P^{\infty} := \lim_{tto\infty} P(t)$ (the opinion that society converges to after a long time) for each node to be influenced. That is, compute $P^{\infty}$ in the case that you influence node 0; then compute $P^{\infty}$ in the case that you influence node 1, and so on. If you care most about $P^{\infty}$ (society's opinions in the very long term), which node is best to influence? Which is worst? Write a few sentences explaining why you think this is the case. If the answer is different from what you got in Deliverable 2, try to explain why.

### Deliverable 1

The code for this assignment will be turned in via a .pdf file of the code that is included a .zip file with the .pdf inside the Deliverable1 folder of the .zip. There is also output that can be generated from running the code within the unzipped project .zip folder. The code and the .zip will take care of Deliverable 1 as well as setting up the basic file structure required for running the program within the parent directory of the unzipped folder. For trouble shooting, a README.md is included with helpful information.

### Deliverable 2

For Deliverable 2 we are to evaluate the overall networks opinions in the short term at running one time step. Starting from time step 0, we can see see from Figure 1.5 a graph with the $P(t)$ values for each possible connection that can be made.

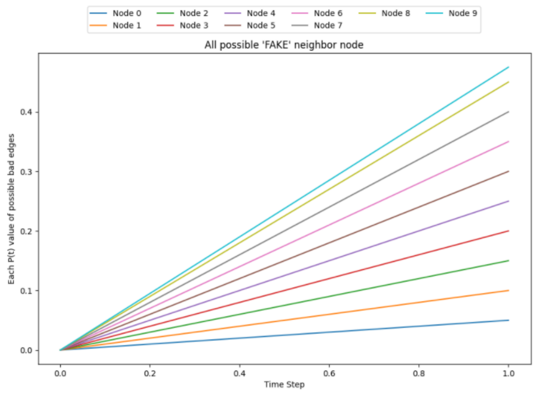
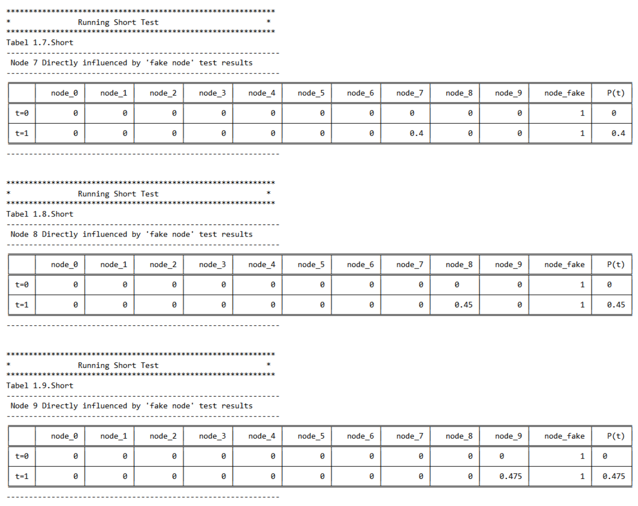


Figure 1.5

The propaganda value is plotted over 1 time step. When lookin at this graph we see that when the fake node is being directly pointed to by nodes 9, 8, and 7, the propaganda values seem to rise to the highest. Lets Node 9 being the highest. Looking at the table in Figure 1.6 we can also see that there values do seem to be the highest. As we can see having a connection from the fake node to node 9 seem to have the highest influence on the network over the first time step. We see it gets as high as 0.475 followed by P-values of fake connections from node 8 and 7 with values 0.45 and 0.4 respectively. This is probably due to the fact that the susceptibility of node 9 has the highest corresponding lambda value, 0.95. This seem to correlate with the other higher influential connections as 8 and 7 have lambda values 0.9 and 0.8.



```
************************************************************
*                Running Short Test                        *
************************************************************
Tabel 1.7.Short
------------------------------------------------------------
 Node 7 Directly influenced by 'fake node' test results
------------------------------------------------------------
```

| | node_0 | node_1 | node_2 | node_3 | node_4 | node_5 | node_6 | node_7 | node_8 | node_9 | node_fake | P(t) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| t=0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| t=1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.4 | 0 | 0 | 1 | 0.4 |

```
------------------------------------------------------------

************************************************************
*                Running Short Test                        *
************************************************************
Tabel 1.8.Short
------------------------------------------------------------
 Node 8 Directly influenced by 'fake node' test results
------------------------------------------------------------
```

| | node_0 | node_1 | node_2 | node_3 | node_4 | node_5 | node_6 | node_7 | node_8 | node_9 | node_fake | P(t) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| t=0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| t=1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.45 | 0 | 1 | 0.45 |

```
------------------------------------------------------------

************************************************************
*                Running Short Test                        *
************************************************************
Tabel 1.9.Short
------------------------------------------------------------
 Node 9 Directly influenced by 'fake node' test results
------------------------------------------------------------
```

| | node_0 | node_1 | node_2 | node_3 | node_4 | node_5 | node_6 | node_7 | node_8 | node_9 | node_fake | P(t) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| t=0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| t=1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.475 | 1 | 0.475 |

```
------------------------------------------------------------
```

Figure 1.6

Referring back to Figure 1.4, we see that the lowest propaganda values when the connection to the fake node are the lowest at values 0, 1, and 2. 0 being the lowest at a value 0.05, followed by nodes 1 and 2 with overall $P(t)$ values 0.1 and 0.15 respectively. We retrieved from a table generated like the one in Figure 1.6. This also tracks with our logic since these nodes have corresponding very low lambda values.

So, we can say that if the fake node wants the influence the network the most then it should have node 9 listen to it since it is the most susceptible. Node 0 being the lest influential node that the fake node would have little influence and low propaganda values.

## Deliverable 3

For Deliverable 3 we are to evaluate the overall networks opinions in the long term. By looking at a graph over a longer period of time we see that the $P(t)$ values seem ton converge failry quick and flatten out over just about 10-15 time steps. With the graph generated in Figure 1.7 we see that the $P(t)$ seem to have different correlation to the first time steps.

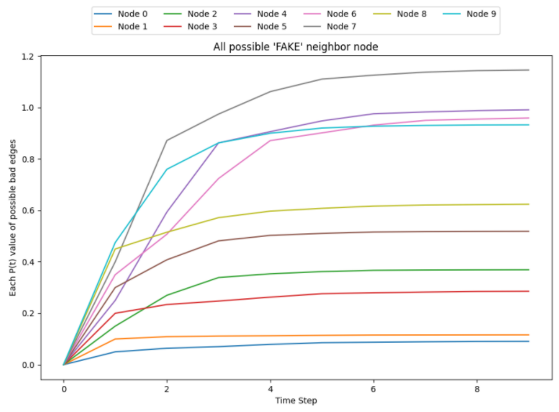
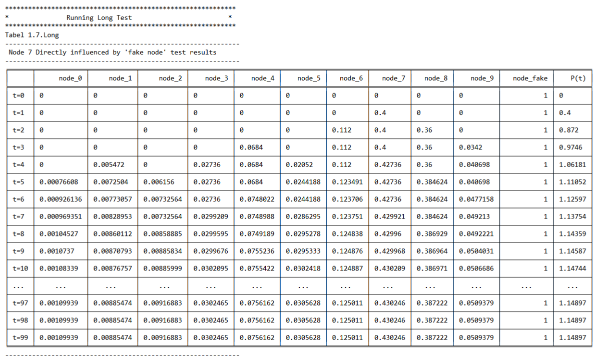


Figure 1.7

Here we can see that after just the first 10 time steps we start to see convergence of most of the nodes in just the first 3-4 time steps. The other values do start to converge but some do have intersection points where the $P(t)$ continued to grow in the network compared to ones that were more rapidly flattening out. For example, even though node 9 seem to have the most influential connection in the first few time steps, nodes 6 and 4 end up having a higher influence on the network than 9. However the node with the most influence over 10 time steps was node 7, in just ten time steps the value rises from 0 to just below 1.2. Looking at the generate table shown in figure 1.8 we see that node 7 gets up to 1.4897 and does not get any higher than that.


```
************************************************************
*                 Running Long Test                        *
************************************************************
Tabel 1.7.Long
------------------------------------------------------------
 Node 7 Directly influenced by 'fake node' test results
------------------------------------------------------------
```

| | node_0 | node_1 | node_2 | node_3 | node_4 | node_5 | node_6 | node_7 | node_8 | node_9 | node_fake | P(t) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| t=0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| t=1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.4 | 0 | 0 | 1 | 0.4 |
| t=2 | 0 | 0 | 0 | 0 | 0 | 0 | 0.112 | 0.4 | 0.36 | 0 | 1 | 0.872 |
| t=3 | 0 | 0 | 0 | 0 | 0.0684 | 0 | 0.112 | 0.4 | 0.36 | 0.0342 | 1 | 0.9746 |
| t=4 | 0 | 0.005472 | 0 | 0.02736 | 0.0684 | 0.02052 | 0.112 | 0.42736 | 0.36 | 0.040698 | 1 | 1.06181 |
| t=5 | 0.00076608 | 0.0072504 | 0.006156 | 0.02736 | 0.0684 | 0.0244188 | 0.123491 | 0.42736 | 0.384624 | 0.040698 | 1 | 1.11052 |
| t=6 | 0.000926136 | 0.00773057 | 0.00732564 | 0.02736 | 0.0748022 | 0.0244188 | 0.123706 | 0.42736 | 0.384624 | 0.0477158 | 1 | 1.12597 |
| t=7 | 0.000969351 | 0.00828953 | 0.00732564 | 0.0299209 | 0.0748988 | 0.0286295 | 0.123751 | 0.429921 | 0.384624 | 0.049213 | 1 | 1.13754 |
| t=8 | 0.00104527 | 0.00860112 | 0.00858885 | 0.0299595 | 0.0749189 | 0.0295278 | 0.124838 | 0.42996 | 0.386929 | 0.0492221 | 1 | 1.14359 |
| t=9 | 0.0010737 | 0.00870793 | 0.00885834 | 0.0299676 | 0.0755236 | 0.0295333 | 0.124876 | 0.429968 | 0.386964 | 0.0504031 | 1 | 1.14587 |
| t=10 | 0.00108339 | 0.00876757 | 0.00885999 | 0.0302095 | 0.0755422 | 0.0302418 | 0.124887 | 0.430209 | 0.386971 | 0.0506686 | 1 | 1.14744 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| t=97 | 0.00109939 | 0.00885474 | 0.00916883 | 0.0302465 | 0.0756162 | 0.0305628 | 0.125011 | 0.430246 | 0.387222 | 0.0509379 | 1 | 1.14897 |
| t=98 | 0.00109939 | 0.00885474 | 0.00916883 | 0.0302465 | 0.0756162 | 0.0305628 | 0.125011 | 0.430246 | 0.387222 | 0.0509379 | 1 | 1.14897 |
| t=99 | 0.00109939 | 0.00885474 | 0.00916883 | 0.0302465 | 0.0756162 | 0.0305628 | 0.125011 | 0.430246 | 0.387222 | 0.0509379 | 1 | 1.14897 |

```
------------------------------------------------------------
```

Figure 1.8

We can probably contribute this to the fact that while node 7 does have a relatively average lambda value and has two nodes listening to it. Both these nodes listening to node 7 have high lambda values meaning higher susceptibility. Node 8 has a 0.9 value and node 7 has a 0.8 lambda value. Node 8 has a very strong connection of 1 and node 6 has a 0.4 weight connection to 7. These two nodes also share a connection with node 4 where node 4 has many other highly susceptible values as well. We can also see loops with in the strongly susceptible nodes. For the lower influential nodes, when the fake node has a connection from node 0, it continues to have the lowest influence on the network and this probably attributes to it low susceptibility as well as the fact that the only node listening to node 1 is node 0 with a even lower susceptibility. So, in the longer runs we can see that node 7 has the highest influence on the network while node 1 seem to continue to have the lowest influence on the network.

## Conclusion

In conclusion we saw how different connection on a newtork with a node with a opnion in contrast to nodes that have none, there are many different ways and rates that the opnion can be spread through the network. When looking as very longer iteration run there were also these weird clefts that would appear a little latter after it seems like that flattening out. In figure 1.9 we see an example where the P value seem to flatten out and then spike up a little bit real quick and flatten out again.

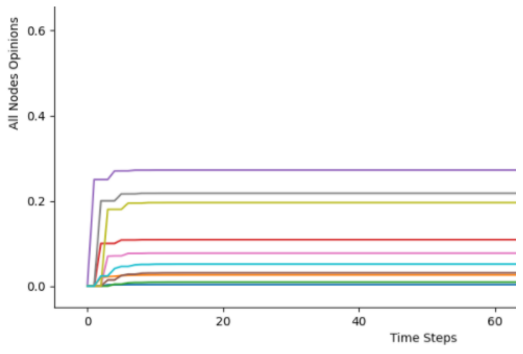


Figure 1.9

While i can't be sure what causes this exactly I have my suspensions there could be possible loops where the opnion can grow quickly when lopping through and back to a highly susceptible node.

## References

Credit to Dr. Phillip Brown's Jupyter note books and other class handouts, lecture and notes.

Networks, Crowds, and Markets: A Book by David Easley and Jon Kleinberg, https://www.cs.cornell.edu/home/kleinber/networks-book/.