Spread of Gossip In a Social Network

Transport Phenomena in Biological Systems
Choose-Focus-Analyse

BT5051

Shreeharsha G Bhat | BE21B037
Department of Biotechnology
Indian Institute of Technology Madras

Table of Contents

1 Abstract **3**
2 Background **3**
2.1 Gossip 3
2.2 Graphs 3
2.2.1 Properties of Graphs: 4
2.2.2 Types of graphs: 4
2.3 Pull-Push Model of Gossip Spread 6
3 Principles **7**
3.1 Equation of continuity 7
3.2 Fick's First Law 7
4 Analysis **8**
The Model 8
Popularity 8
Closeness 8
Level of Trust 9
Flux of Information 9
Continuity Equation for a single node 10
5 Conclusion **12**
6 Future work **12**
7 Acknowledgements **13**
8 References **13**
Appendix **13**
Code 13

1 Abstract

Social circles are built on gossip, at least in my experience. This article aims to model the flow of gossip through different social networks using concepts from transport phenomena and graph theory. The model qualitatively proves that social circles can become more tightly knit as gossip spreads. It models the essence of gossip showing how there is a rapid initial increase and then a gradual drop in the rate of its spread.

2 Background

2.1 Gossip

Gossip mainly involves the informational exchange about behaviours and characteristics of a person not part of the conversation.[1] Gossip has a bad reputation in most friend groups or communities since it is generally attributed to a person trying to ruin the reputation of others. But, surprisingly, gossip is more often than not used to acquire information rather than attempt to ruin another's reputation.[2] Even though gossip gets a bad rap, gossip in evolutionary terms was key to survival of large groups. To early neanderthals, it was not just important for them to know the location of food and water sources, or even the location of threats, but, it was more important for them to discuss who in their group hates whom, who is honest and who is a cheat. Gossip is key to a group's survival. Human beings in the wake of their cognitive revolution were able to form larger and more stable groups majorly due to their ability to gossip. But most groups break around the size of 150 people. They can neither intimately know nor effectively gossip about larger group sizes[3].

2.2 Graphs

Definition: A graph G, can be represented as a set of vertices and a set of edges. $G := (V, E)$. It is a mathematical structure that represents a particular function with a set of points and connections. In a loopless graph, an edge connects two different vertices.
The following are examples of a few graphs:

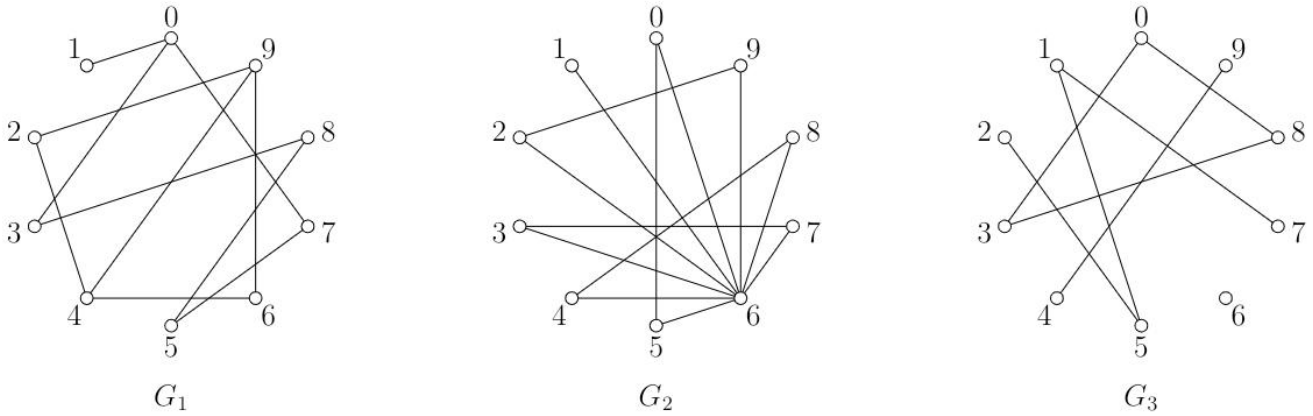

Fig 1: Example graphs

2.2.1 Properties of Graphs:

Degree: The degree of a node is the number of edges incident on that node. For example, the degree of the vertex 4 in G_1 is 3

Edge Weight: The weight of an edge is given by the value assigned to that edge. The value assigned would denote a property of that edge. For example, if we consider cities on a map as nodes and the roads as edges. The amount of traffic on a particular road/edge can be considered as its weight.

Colour: The nodes and edges could also be coloured to denote a property of that node or edge. For example: A person who is travelling from city to city would mark all the visited cities (nodes) in **green** and all the non-visited cities in **red**

2.2.2 Types of graphs:

Depending on the properties of the graph it can be categorised into different types

a) Directed & Undirected :

If a direction is assigned to an edge in the graph i.e, you can traverse only in one direction. The graph becomes a directed graph. For example: A one-way road in a city would be considered as a directed edge, as you can traverse only in one direction.

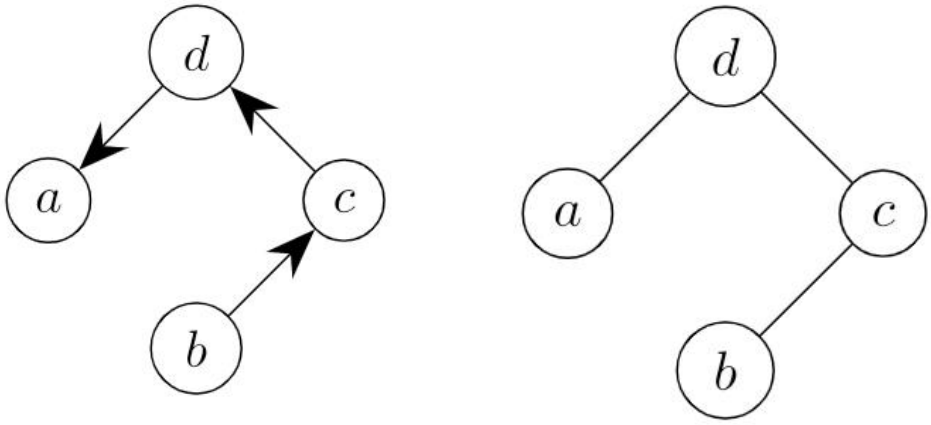

Fig 2: Directed Graph(left) and Undirected Graph(right)

b) Simple Graph: We define a simple graph as a graph that does not contain loop(edge starting and ending on the same vertex) or parallel edges(two or more edges having the same start and end node)

c) k–regular Graph: A graph where each and every node has a degree = k is a k–regular graph.

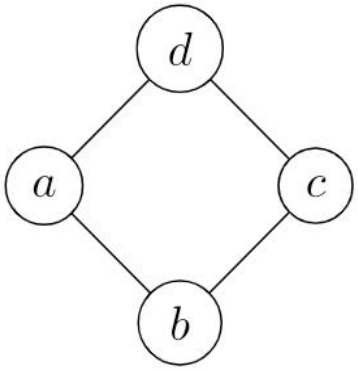

Fig 3: A 2–regular graph on 4 vertices

d) Reachability: Node 1 is reachable from node 2 if there is a sequence of nodes and edges that start at Node 2 and end at Node 1.

e) Connected Components: A graph is considered connected if every node is reachable from every other node. A given graph can have multiple connected components. In a particular connected component of the graph all nodes are reachable to each other. For example

Graph 3 has 4 connected components. The vertex set {2, 5, 1, 7}, {4, 9}, {0, 3, 8}, {6}

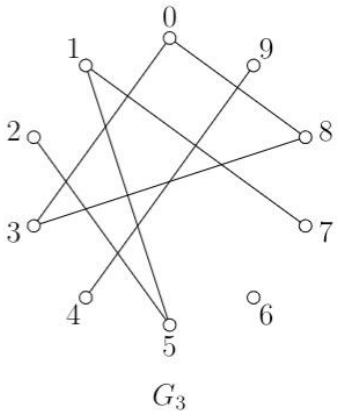

Fig 4: Graph 3 having 4 connected components

f) Complete Graphs: A complete graph is one where there exists an edge between every two nodes

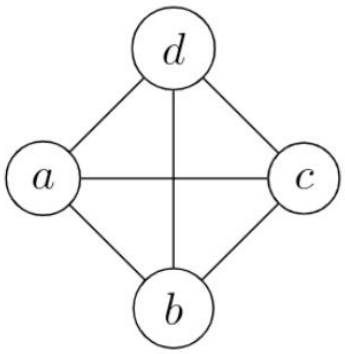

Fig 5: A complete graph on 4 nodes

Note : A complete graph on N nodes is the same as an [N-1]–regular graph on N nodes

2.3 Pull-Push Model of Gossip Spread

A group of friends can be considered as a graph. Where the people represent the nodes and their "friendships" can be considered as an edge. The relationship is **symmetric** (if A is a friend of B then B is a friend of A) but **not transitive** (if A is a friend of B and B is a friend of C, A and C need not be friends necessarily).

Each node/person can be assigned a colour, **green** if he knows the gossip and **red** if he doesn't know the gossip.

We start with one person aware of the gossip and his node is coloured **green**. The information is passed to others based on a simple algorithm. At time 't' if node A is **green**, then at t + Δt all the nodes connected to A that are **red** will be changed to **green**.

In simpler words, once a person is aware of the gossip, he will tell all his friends that are not aware of the gossip.

The information is **pushed from a node** and **pulled by the other**. Hence, the pull–push model.

3 Principles

3.1 Equation of continuity

Given a system, few input streams, few output streams and some components. The following are the five things that can occur to the component of interest.[5]

1) It can enter the system : **I = Input**
2) It can exit the system : **O = Output**
3) It can be generated in the system due to a reaction : **G = generation**
4) It can be consumed in the system as a result of a reaction : **C = Consumption**
5) It can accumulate in the system : **A = Accumulation**

The net equation described is the equation of continuity that describes the fate of a component of interest

$$I - O + G - C = A$$

If we represent the time rates of each of the components with a dot above their respective symbols we get

$$\dot{I} - \dot{O} + \dot{G} - \dot{C} = \frac{dA}{dt}$$

3.2 Fick's First Law

Fick's first law of diffusion helps relate flux with the concentration gradient. It states that diffusion of flux is directly proportional to the concentration gradient[5]

J − diffusion flux : = amount of substance flowing through a unit area per unit time

D − diffusion coefficient

$\frac{\partial C}{\partial x}$ − Gradient of concentration of the diffusing molecules, x is the positional variable

4 Analysis

We shall perform our analysis on the flow of information using a model and different graph networks. We want to analyse the flow of gossip through friend network with variation in

- P – Measure of Popularity
- L – Measure of closeness (a level of friendship, how close they are to each other)
- K – Measure of trust

The component of interest that is flowing or diffusing is information or Gossip[4]. The gossip flows due to the presence of a popularity gradient. This popularity gradient exists due to the difference in popularity of the sharer and the listener of the gossip and is inversely related to how close the sharer and listener are to each other. The closer they are to each other, the more likely they are to share gossip. Hence, smaller the L value, greater chances of sharing the gossip. The measure of trust acts as the diffusion coefficient, since the more trust two nodes have, more will be the flow of gossip.

The Model

We modify the pull–push model to incorporate the above factors into the given graph network.

Popularity

We assign a popularity value based on the degree of each node in the graph network. We scale the degree of each node by some value as a way of measuring their popularity.

Popularity of a node = Degree of the node * 10

Closeness

To assign the closeness of friendship value, we use a random number generator to define lengths from [1,5] . With each time step, where each node that has the gossip shares it with one other node, the closeness is decreased. This is done to mimic the ideas of Yuval Noah in the book Sapiens where he explains how gossip can make a group more stable[3].

$$L(t + \Delta t) = L(t) \cdot e^{-|P_{Diff}|}$$

$L(t + \Delta t)$ – closeness between two nodes after gossip is shared
$L(t)$ – closeness between the two nodes before gossip is shared
P_{Diff} – Difference in popularity between the two nodes

The reason that this is done is that people with higher differences in popularity come closer together then people who have similar popularity differences. Any function that depicts such a relation can be chosen.

Level of Trust

In this model we assume that each person has an equal probability of sharing or not sharing when given the opportunity. K = {1, 0} with equal probability. K is either 1 or 0 at every time step.

Flux of Information

We define the Instantaneous flux of information that is flowing between two consecutive time steps as

$$J_{Inst} = \frac{Change\ in\ number\ of\ people\ who\ know\ the\ gossip}{Total\ number\ of\ people\ in\ the\ network}$$

Here the $\Delta t = 1$

We define the Cumulative flux of information that is flowing between two consecutive time steps as

$$J_{Cumulative} = \frac{Total\ number\ of\ people\ who\ know\ the\ gossip}{Total\ number\ of\ people\ in\ the\ network \cdot TimeStep}$$

The Pull-Push Model is modified as such:

We start with making **one** random node set to **green**. At each time step, every node that is **green** decides to share the gossip with one other person, turning one of its neighbours **green**. This process goes on till everyone in the network is aware of the gossip i.e, all the nodes are **green**.

Every **green** node has a choice to spread gossip to one of its neighbours. The choice is randomised based on their probabilities. This is best explained by an example,

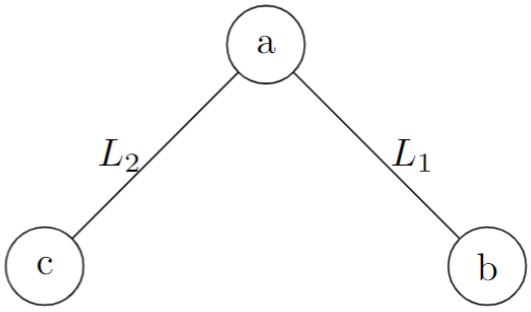

Fig 6: Example

a, b, c represent the popularity. L_1 and L_2 represent the closeness. The probability the information is shared to c & b is given by

$$P_c = \frac{\frac{|a-c+1|}{L_2}}{\frac{|a-c+1|}{L_2} + \frac{|a-b+1|}{L_1}} \qquad P_b = \frac{\frac{|a-b+1|}{L_1}}{\frac{|a-c+1|}{L_2} + \frac{|a-b+1|}{L_1}}$$

Which is just the probabilities calculated for the **popularity gradients.**[1]
To simplify, when gossiping, a person is **more likely** to pick the friend with which he has the **highest popularity gradient**.

Continuity Equation for a single node

We apply the continuity equation for a single person in the network. At a given time step there is either a combination of Input and Accumulation of information (when the node is a listener) or there is a combination of Generation and Output of information (when the node is the sharer).

We now run this model for a complete graph, a 2–regular graph, and a randomised graph, for node size of 100, 125, 150. We analyse the Instantaneous flux, Cumulative Flux & the mean closeness.

[1] The 1 is added to the popularity gradient to work around the case of probability = 0 when the popularity difference is = 0. This is in accordance with Laplace's rule of succession

Complete Graph

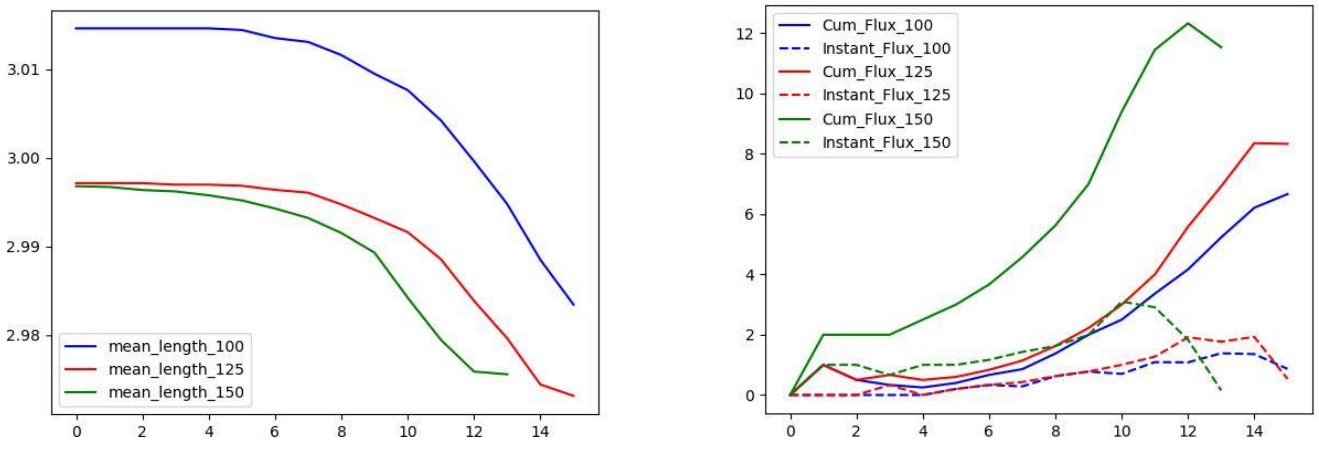

Fig 7: Mean closeness vs time(left), Cumulative & Instant Flux variation vs time (right)

2 – regular Graph

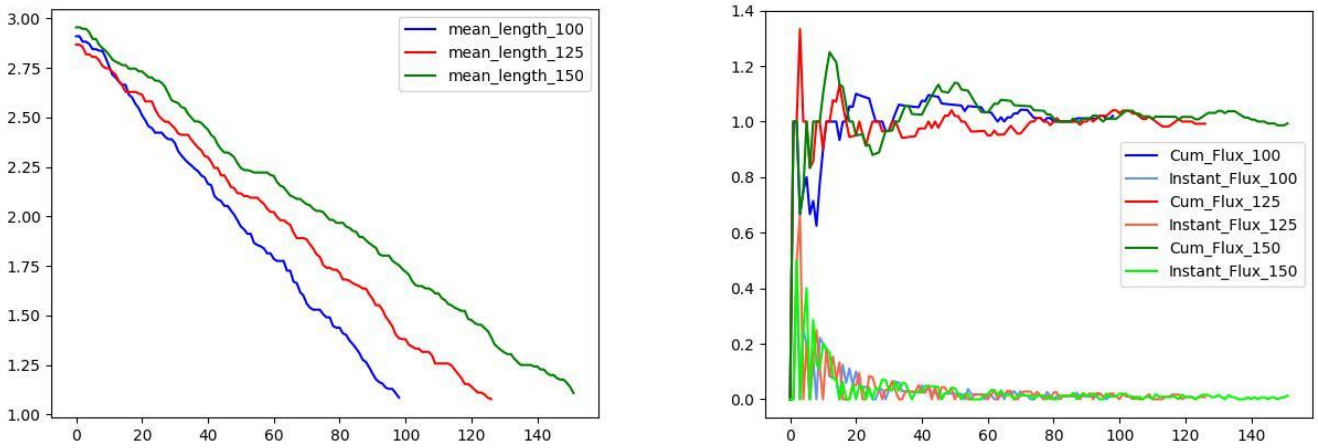

Fig 8: Mean closeness vs time(left), Cumulative & Instant Flux variation vs time (right)

Randomised Graph

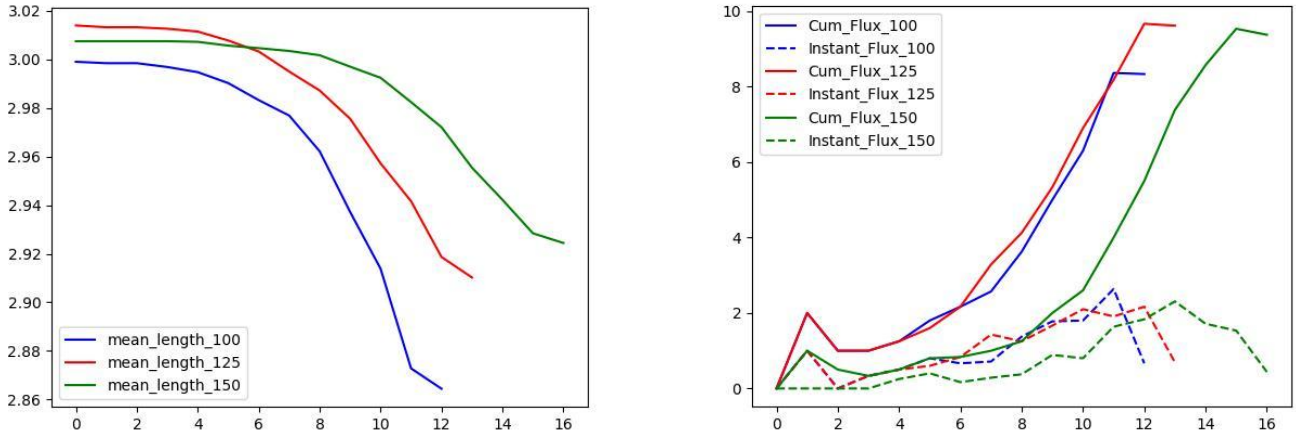

Fig 9: Mean closeness vs time(left), Cumulative & Instant Flux variation vs time (right)

5 Conclusion

We observe the flow of information in the model. The information flux(cumulative) starts slow, continues to grow at a high rate and starts to saturate towards the end. This is expected as initially there are more **red** nodes than **green** nodes, while towards the end, the number of **green** nodes is more than the number of **red** nodes. The model has mimicked real life experiences that we all have seen or experienced. Gossip (sometimes disguised as rumours) spread like wildfire initially and later tend to saturate towards the end. While you may feel haunted by an embarrassing rumour being travelling around, you can live safely with the knowledge that it helped bring the group close together. This is depicted in the mean closeness decreasing with time.

The 2-regular graph is an interesting graph to analyse. If we think about it at each time step only one node will be turned green. Hence there is a linear decrease in mean length and a logarithmic type decrease in the instantaneous group. The 2-regular graph was analysed,

6 Future work

There are few places where this work can be extended further.

1) Analysis could be performed by changing the diffusivity. Here in the model, the person either chooses to tell or not tell the information(1 or 0 trust factor). We can increase the complexity of the model by keeping the trust factor a variable.
2) We can explore the process of isolation occurring due to a single person, or a small group of people refraining from spreading gossip.

3) In the above model the popularity was kept a constant as a number directly proportional to the degree of the node. Real life gossip flow could be mimicked more if the popularity was allowed to change by allowing more connections to be formed during the spread of gossip

7 Acknowledgements

I would like to thank Professor G K Suraishkumar for giving me the opportunity to take part in writing this report as part of his Transport Phenomena in Biological Systems course. His course has been one of the most enjoyable courses I have taken yet.

8 References

1. Dunbar, R. I. (2004). Gossip in evolutionary perspective. *Review of General Psychology*, *8*(2), 100–110. https://doi.org/10.1037/1089-2680.8.2.100
2. Hartung, F.-M., Krohn, C., & Pirschtat, M. (2019). *Better than its reputation? gossip and the reasons why we and individuals with "dark" personalities talk about others*. Frontiers. https://www.frontiersin.org/articles/10.3389/fpsyg.2019.01162/full
3. Harari, Y. N. (2018). *Sapiens: A brief history of humankind*. Harper Perennial.
4. Banerjee, A., Chandrasekhar, A. G., Duflo, E., & Jackson, M. O. (2019). Using gossips to spread information: Theory and evidence from two randomized controlled trials. *The Review of Economic Studies*, *86*(6), 2453–2490. https://doi.org/10.1093/restud/rdz008
5. Suraishkumar, G. K. (2014). Continuum Analysis of Biological Systems. *Biosystems & Biorobotics*. https://doi.org/10.1007/978-3-642-54468-2

Appendix

Code

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

```
random.seed(5051)

number = 10
#############Uncomment and Run for randomized graph##################
# G = nx.Graph()
# for i in range(1,number+1):
```

```
#   G.add_node(i)

# while nx.number_connected_components(G) != 1:
#   edges = []
#   edge_length = {}
#   popularity = {node:0 for node in list(G.nodes)}
#   for i in range((number*(number-1))//4):
#     edge = random.sample(list(G.nodes),k=2)
#     if [edge[1],edge[0]] not in edges and [edge[0],edge[1]] not in edges:
#       edge.sort()
#       edges.append(edge)
#       popularity[edge[0]] += 1
#       popularity[edge[1]] += 1
#   edge_length = {tuple(edges[i]) : round(random.uniform(1, 5),3) for i in
range(len(edges))}
#   edges = list(edge_length.keys())
#   G = nx.Graph()
#   G.add_edges_from(edges)

#############for complete and regular graphs##################
G = nx.random_regular_graph(d = 2, n = 150)
while nx.number_connected_components(G) != 1:
  G = nx.random_regular_graph(d = 2, n = 150)
popularity = {node:2 for node in list(G.nodes)}
edges = []
for a,b in list(G.edges):
  if a > b:
    a,b=b,a
    edges.append((a,b))
  else:
    edges.append((a,b))

edge_length = {tuple(edge) : round(random.uniform(1, 5),3) for edge in edges}

for node in list(G.nodes):
  popularity[node] *= 10
####################For visualizing the network#####################################
# fig, ax = plt.subplots(figsize=(10,10),dpi=50)
# Graph(edges,edge_labels=edge_length,
node_layout='geometric',node_layout_kwargs=dict(edge_length=edge_length),
ax=ax,scale=(1.0, 10.0))
# ax.set_aspect('equal')
```

```
# plt.show()

state = {i:"Red" for i in list(G.nodes)}
start_node = random.choice(list(G.nodes))
state[start_node] = "Green"

def pick_listener(sharer,state,popularity,lengths):
  available = []
  for node in list(G.neighbors(sharer)):
    if state[node] == "Red":
      available.append(node)
  probabilities = []
  for node in available:
    length = None
    if node < sharer:
      length = lengths[(node,sharer)]
    else:
      length = lengths[(sharer,node)]
    p = abs(popularity[sharer]-popularity[node] + 1)/length
    probabilities.append(p)
  probabilities = [p/sum(probabilities) for p in probabilities]

  if len(available) != 0:
    diffusivity = random.choice([0,1])
    if diffusivity == 1:
      listener = random.choices(available,probabilities)[0]
      return listener
    else:
      return None
  else:
    return None

def spread_gossip(local_know,state,popularity,lengths):
  current_state = state.copy()
  current_lengths = lengths.copy()
  for node in list(G.nodes):
    if state[node] == "Green":
      listener = pick_listener(node,state,popularity,lengths)
      if listener != None and current_state[listener] != "Green":
        current_state[listener] = "Green"
        local_know += 1
        edge = None
```

```
        if listener < node:
          edge = tuple([listener,node])
        else:
          edge = tuple([node,listener])
        current_lengths[edge] =
round(current_lengths[edge]*math.exp(-(abs(popularity[node]-popularity[listener]+1))/
1),3)

  return local_know, current_state, current_lengths

global_know = 1
knows = [global_know]
time = 0
mean_lengths = []

print(f"Time = {time}")
print(global_know)
print(state)
print(edge_length)
mean_length = sum(edge_length.values())/len(edge_length)
mean_lengths.append(mean_length)
print()
time += 1


while global_know < len(list(G.nodes)):
  print(f"Time = {time}")
  global_know, state, edge_length =
spread_gossip(global_know,state,popularity,edge_length)
  knows.append(global_know)
  mean_length = sum(edge_length.values())/len(edge_length)
  mean_lengths.append(mean_length)
  print(global_know)
  print(state)
  print(edge_length)
  print()
  time += 1
```

Plot mean lengths

```
plt.plot(range(time),mean_lengths);
```

Plot Instantaneous and Cumulative Flux

```
flux = [0]
instantaneous = [0]
for i in range(1,len(knows)):
  flux.append((knows[i])/i)
  instantaneous.append((knows[i]-knows[i-1])/i)
plt.plot(range(time),flux,label="Cum_Flux_100")
plt.plot(range(time),instantaneous,label="Instant_Flux_100")
plt.legend();
```

****

CFA Report

by Shreeharsha G Bhat

Submission date: 06-Nov-2023 11:43PM (UTC+0530)
Submission ID: 2219605577
File name: Shreeharsha_BE21B037_CFA.pdf (659.74K)
Word count: 3104
Character count: 18376

Spread of Gossip In a Social Network

Transport Phenomena in Biological Systems
Choose-Focus-Analyse

BT5051

Shreeharsha G Bhat | BE21B037
Department of Biotechnology
Indian Institute of Technology Madras

Table of Contents

1 Abstract **3**

2 Background **3**

2.1 Gossip 3

2.2 Graphs 3

2.2.1 Properties of Graphs: 4

2.2.2 Types of graphs: 4

2.3 Pull-Push Model of Gossip Spread 6

3 Principles **7**

3.1 Equation of continuity 7

3.2 Fick's First Law 7

4 Analysis **8**

The Model 8

Popularity 8

Closeness 8

Level of Trust 9

Flux of Information 9

Continuity Equation for a single node 10

5 Conclusion **12**

6 Future work **12**

7 Acknowledgements **13**

8 References **13**

Appendix **13**

Code 13

1 Abstract

Social circles are built on gossip, at least in my experience. This article aims to model the flow of gossip through different social networks using concepts from transport phenomena and graph theory. The model qualitatively proves that social circles can become more tightly knit as gossip spreads. It models the essence of gossip showing how there is a rapid initial increase and then a gradual drop in the rate of its spread.

2 Background

2.1 Gossip

Gossip mainly involves the informational exchange about behaviours and characteristics of a person not part of the conversation.[1] Gossip has a bad reputation in most friend groups or communities since it is generally attributed to a person trying to ruin the reputation of others. But, surprisingly, gossip is more often than not used to acquire information rather than attempt to ruin another's reputation.[2] Even though gossip gets a bad rap, gossip in evolutionary terms was key to survival of large groups. To early neanderthals, it was not just important for them to know the location of food and water sources, or even the location of threats, but, it was more important for them to discuss who in their group hates whom, who is honest and who is a cheat. Gossip is key to a group's survival. Human beings in the wake of their cognitive revolution were able to form larger and more stable groups majorly due to their ability to gossip. But most groups break around the size of 150 people. They can neither intimately know nor effectively gossip about larger group sizes[3].

2.2 Graphs

Definition: A graph G, can be represented as a set of vertices and a set of edges. $G := (V, E)$. It is a mathematical structure that represents a particular function with a set of points and connections. In a loopless graph, an edge connects two different vertices.
The following are examples of a few graphs:

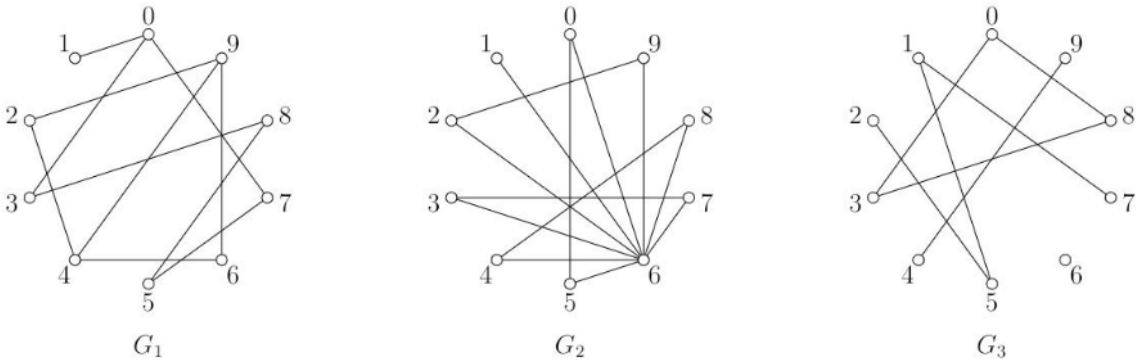

Fig 1: Example graphs

2.2.1 Properties of Graphs:

Degree: The degree of a node is the number of edges incident on that node. For example, the degree of the vertex 4 in G_1 is 3

Edge Weight: The weight of an edge is given by the value assigned to that edge. The value assigned would denote a property of that edge. For example, if we consider cities on a map as nodes and the roads as edges. The amount of traffic on a particular road/edge can be considered as its weight.

Colour: The nodes and edges could also be coloured to denote a property of that node or edge. For example: A person who is travelling from city to city would mark all the visited cities (nodes) in **green** and all the non-visited cities in **red**

2.2.2 Types of graphs:

Depending on the properties of the graph it can be categorised into different types

a) Directed & Undirected :

If a direction is assigned to an edge in the graph i.e, you can traverse only in one direction. The graph becomes a directed graph. For example: A one-way road in a city would be considered as a directed edge, as you can traverse only in one direction.

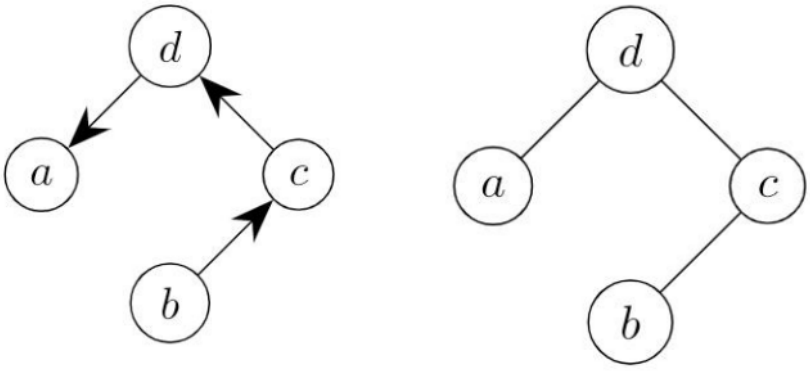

Fig 2: Directed Graph(left) and Undirected Graph(right)

b) Simple Graph: We define a simple graph as a graph that does not contain loop(edge starting and ending on the same vertex) or parallel edges(two or more edges having the same start and end node)

c) k–regular Graph: A graph where each and every node has a degree = k is a k–regular graph.

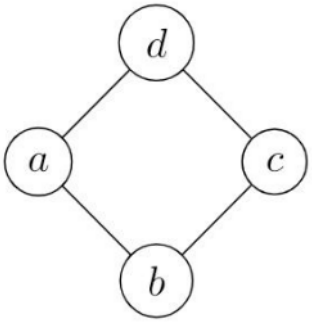

Fig 3: A 2–regular graph on 4 vertices

d) Reachability: Node 1 is reachable from node 2 if there is a sequence of nodes and edges that start at Node 2 and end at Node 1.

e) Connected Components: A graph is considered connected if every node is reachable from every other node. A given graph can have multiple connected components. In a particular connected component of the graph all nodes are reachable to each other. For example
Graph 3 has 4 connected components. The vertex set {2, 5, 1, 7}, {4, 9}, {0, 3, 8}, {6}

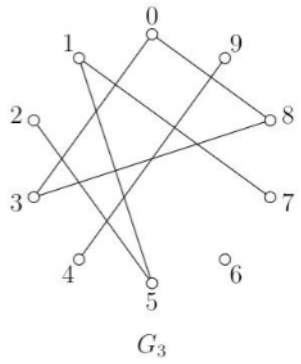

Fig 4: Graph 3 having 4 connected components

f) Complete Graphs: A complete graph is one where there exists an edge between every two nodes

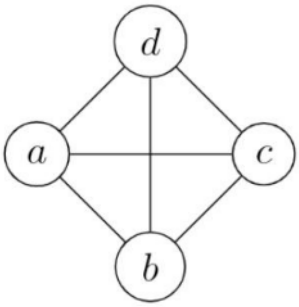

Fig 5: A complete graph on 4 nodes

Note : A complete graph on N nodes is the same as an [N-1]–regular graph on N nodes

2.3 Pull-Push Model of Gossip Spread

A group of friends can be considered as a graph. Where the people represent the nodes and their "friendships" can be considered as an edge. The relationship is **symmetric** (if A is a friend of B then B is a friend of A) but **not transitive** (if A is a friend of B and B is a friend of C, A and C need not be friends necessarily).

Each node/person can be assigned a colour, **green** if he knows the gossip and **red** if he doesn't know the gossip.

We start with one person aware of the gossip and his node is coloured **green**. The information is passed to others based on a simple algorithm. At time 't' if node A is **green**, then at t + Δt all the nodes connected to A that are **red** will be changed to **green**.

In simpler words, once a person is aware of the gossip, he will tell all his friends that are not aware of the gossip.

The information is **pushed from a node** and **pulled by the other**. Hence, the pull–push model.

3 Principles

3.1 Equation of continuity

Given a system, few input streams, few output streams and some components. The following are the five things that can occur to the component of interest.[5]

1) It can enter the system : **I = Input**
2) It can exit the system : **O = Output**
3) It can be generated in the system due to a reaction : **G = generation**
4) It can be consumed in the system as a result of a reaction : **C = Consumption**
5) It can accumulate in the system : **A = Accumulation**

The net equation described is the equation of continuity that describes the fate of a component of interest

$$I - O + G - C = A$$

If we represent the time rates of each of the components with a dot above their respective symbols we get

$$\dot{I} - \dot{O} + \dot{G} - \dot{C} = \frac{dA}{dt}$$

3.2 Fick's First Law

Fick's first law of diffusion helps relate flux with the concentration gradient. It states that diffusion of flux is directly proportional to the concentration gradient[5]

J − diffusion flux : = amount of substance flowing through a unit area per unit time

D – diffusion coefficient

$\frac{\partial C}{\partial x}$ – Gradient of concentration of the diffusing molecules, x is the positional variable

4 Analysis

We shall perform our analysis on the flow of information using a model and different graph networks. We want to analyse the flow of gossip through friend network with variation in

- P – Measure of Popularity
- L – Measure of closeness (a level of friendship, how close they are to each other)
- K – Measure of trust

The component of interest that is flowing or diffusing is information or Gossip[4]. The gossip flows due to the presence of a popularity gradient. This popularity gradient exists due to the difference in popularity of the sharer and the listener of the gossip and is inversely related to how close the sharer and listener are to each other. The closer they are to each other, the more likely they are to share gossip. Hence, smaller the L value, greater chances of sharing the gossip. The measure of trust acts as the diffusion coefficient, since the more trust two nodes have, more will be the flow of gossip.

The Model

We modify the pull–push model to incorporate the above factors into the given graph network.

Popularity

We assign a popularity value based on the degree of each node in the graph network. We scale the degree of each node by some value as a way of measuring their popularity.

Popularity of a node = Degree of the node * 10

Closeness

To assign the closeness of friendship value, we use a random number generator to define lengths from [1,5] . With each time step, where each node that has the gossip shares it with one other node, the closeness is decreased. This is done to mimic the ideas of Yuval Noah in the book Sapiens where he explains how gossip can make a group more stable[3].

$$L(t + \Delta t) = L(t) \cdot e^{-|P_{Diff}|}$$

$L(t + \Delta t)$ – closeness between two nodes after gossip is shared
$L(t)$ – closeness between the two nodes before gossip is shared
P_{Diff} – Difference in popularity between the two nodes

The reason that this is done is that people with higher differences in popularity come closer together then people who have similar popularity differences. Any function that depicts such a relation can be chosen.

Level of Trust

In this model we assume that each person has an equal probability of sharing or not sharing when given the opportunity. K = {1, 0} with equal probability. K is either 1 or 0 at every time step.

Flux of Information

We define the Instantaneous flux of information that is flowing between two consecutive time steps as

$$J_{Inst} = \frac{Change\ in\ number\ of\ people\ who\ know\ the\ gossip}{Total\ number\ of\ people\ in\ the\ network}$$

Here the $\Delta t = 1$

We define the Cumulative flux of information that is flowing between two consecutive time steps as

$$J_{Cumulative} = \frac{Total\ number\ of\ people\ who\ know\ the\ gossip}{Total\ number\ of\ people\ in\ the\ network \cdot TimeStep}$$

The Pull-Push Model is modified as such:

We start with making **one** random node set to **green**. At each time step, every node that is **green** decides to share the gossip with one other person, turning one of its neighbours **green**. This process goes on till everyone in the network is aware of the gossip i.e, all the nodes are **green**.

Every **green** node has a choice to spread gossip to one of its neighbours. The choice is randomised based on their probabilities. This is best explained by an example,

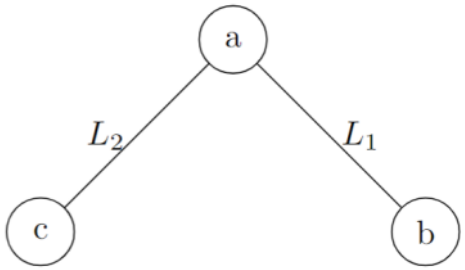

Fig 6: Example

a, b, c represent the popularity. L_1 and L_2 represent the closeness. The probability the information is shared to c & b is given by

$$P_c = \frac{\frac{|a-c+1|}{L_2}}{\frac{|a-c+1|}{L_2} + \frac{|a-b+1|}{L_1}} \qquad P_b = \frac{\frac{|a-b+1|}{L_1}}{\frac{|a-c+1|}{L_2} + \frac{|a-b+1|}{L_1}}$$

Which is just the probabilities calculated for the **popularity gradients.**[1]
To simplify, when gossiping, a person is **more likely** to pick the friend with which he has the **highest popularity gradient**.

Continuity Equation for a single node

We apply the continuity equation for a single person in the network. At a given time step there is either a combination of Input and Accumulation of information (when the node is a listener) or there is a combination of Generation and Output of information (when the node is the sharer).

We now run this model for a complete graph, a 2–regular graph, and a randomised graph, for node size of 100, 125, 150. We analyse the Instantaneous flux, Cumulative Flux & the mean closeness.

[1] The 1 is added to the popularity gradient to work around the case of probability = 0 when the popularity difference is = 0. This is in accordance with Laplace's rule of succession

Complete Graph

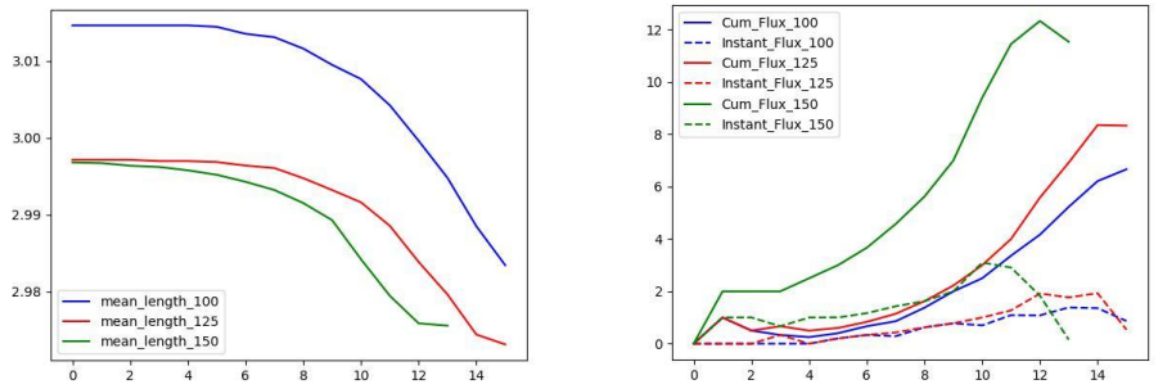

Fig 7: Mean closeness vs time(left), Cumulative & Instant Flux variation vs time (right)

2 – regular Graph

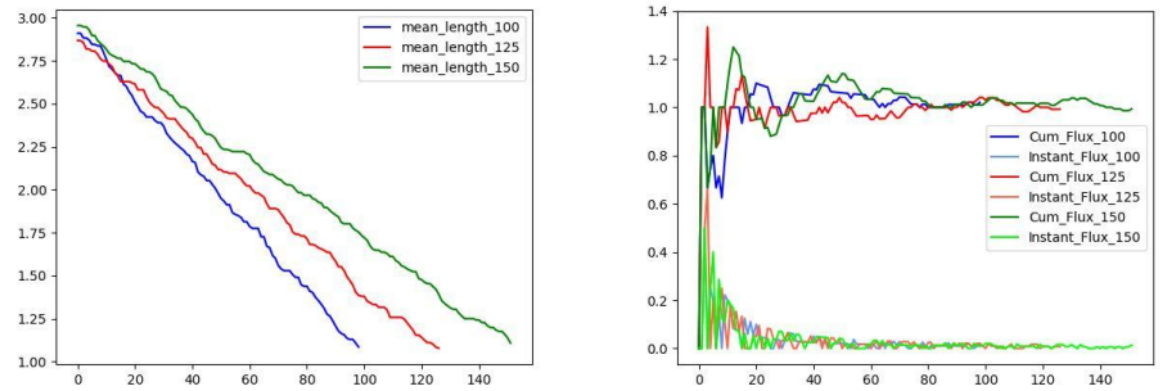

Fig 8: Mean closeness vs time(left), Cumulative & Instant Flux variation vs time (right)

Randomised Graph

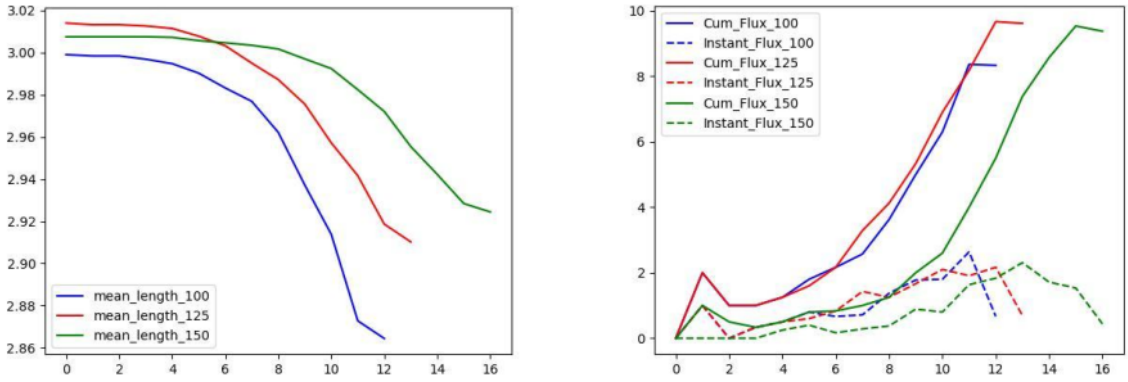

Fig 9: Mean closeness vs time(left), Cumulative & Instant Flux variation vs time (right)

5 Conclusion

We observe the flow of information in the model. The information flux(cumulative) starts slow, continues to grow at a high rate and starts to saturate towards the end. This is expected as initially there are more **red** nodes than **green** nodes, while towards the end, the number of **green** nodes is more than the number of **red** nodes. The model has mimicked real life experiences that we all have seen or experienced. Gossip (sometimes disguised as rumours) spread like wildfire initially and later tend to saturate towards the end. While you may feel haunted by an embarrassing rumour being travelling around, you can live safely with the knowledge that it helped bring the group close together. This is depicted in the mean closeness decreasing with time.

The 2-regular graph is an interesting graph to analyse. If we think about it at each time step only one node will be turned green. Hence there is a linear decrease in mean length and a logarithmic type decrease in the instantaneous group. The 2-regular graph was analysed,

6 Future work

There are few places where this work can be extended further.

1) Analysis could be performed by changing the diffusivity. Here in the model, the person either chooses to tell or not tell the information(1 or 0 trust factor). We can increase the complexity of the model by keeping the trust factor a variable.
2) We can explore the process of isolation occurring due to a single person, or a small group of people refraining from spreading gossip.

3) In the above model the popularity was kept a constant as a number directly proportional to the degree of the node. Real life gossip flow could be mimicked more if the popularity was allowed to change by allowing more connections to be formed during the spread of gossip

7 Acknowledgements

I would like to thank Professor G K Suraishkumar for giving me the opportunity to take part in writing this report as part of his Transport Phenomena in Biological Systems course. His course has been one of the most enjoyable courses I have taken yet.

8 References

1. Dunbar, R. I. (2004). Gossip in evolutionary perspective. *Review of General Psychology*, *8*(2), 100–110. https://doi.org/10.1037/1089-2680.8.2.100
2. Hartung, F.-M., Krohn, C., & Pirschtat, M. (2019). *Better than its reputation? gossip and the reasons why we and individuals with "dark" personalities talk about others*. Frontiers. https://www.frontiersin.org/articles/10.3389/fpsyg.2019.01162/full
3. Harari, Y. N. (2018). *Sapiens: A brief history of humankind*. Harper Perennial.
4. Banerjee, A., Chandrasekhar, A. G., Duflo, E., & Jackson, M. O. (2019). Using gossips to spread information: Theory and evidence from two randomized controlled trials. *The Review of Economic Studies*, *86*(6), 2453–2490. https://doi.org/10.1093/restud/rdz008
5. Suraishkumar, G. K. (2014). Continuum Analysis of Biological Systems. *Biosystems & Biorobotics*. https://doi.org/10.1007/978-3-642-54468-2

Appendix

Code

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

random.seed(5051)

number = 10
##########Uncomment and Run for randomized graph##################
# G = nx.Graph()
# for i in range(1,number+1):
```

```
#   G.add_node(i)

# while nx.number_connected_components(G) != 1:
#   edges = []
#   edge_length = {}
#   popularity = {node:0 for node in list(G.nodes)}
#   for i in range((number*(number-1))//4):
#     edge = random.sample(list(G.nodes),k=2)
#     if [edge[1],edge[0]] not in edges and [edge[0],edge[1]] not in edges:
#       edge.sort()
#       edges.append(edge)
#       popularity[edge[0]] += 1
#       popularity[edge[1]] += 1
#   edge_length = {tuple(edges[i]) : round(random.uniform(1, 5),3) for i in
range(len(edges))}
#   edges = list(edge_length.keys())
#   G = nx.Graph()
#   G.add_edges_from(edges)

#############for complete and regular graphs##################
G = nx.random_regular_graph(d = 2, n = 150)
while nx.number_connected_components(G) != 1:
  G = nx.random_regular_graph(d = 2, n = 150)
popularity = {node:2 for node in list(G.nodes)}
edges = []
for a,b in list(G.edges):
  if a > b:
    a,b=b,a
    edges.append((a,b))
  else:
    edges.append((a,b))

edge_length = {tuple(edge) : round(random.uniform(1, 5),3) for edge in edges}

for node in list(G.nodes):
  popularity[node] *= 10
##################For visualizing the network#################################
# fig, ax = plt.subplots(figsize=(10,10),dpi=50)
# Graph(edges,edge_labels=edge_length,
node_layout='geometric',node_layout_kwargs=dict(edge_length=edge_length),
ax=ax,scale=(1.0, 10.0))
# ax.set_aspect('equal')
```

```
# plt.show()

state = {i:"Red" for i in list(G.nodes)}
start_node = random.choice(list(G.nodes))
state[start_node] = "Green"

def pick_listener(sharer,state,popularity,lengths):
  available = []
  for node in list(G.neighbors(sharer)):
    if state[node] == "Red":
      available.append(node)
  probabilities = []
  for node in available:
    length = None
    if node < sharer:
      length = lengths[(node,sharer)]
    else:
      length = lengths[(sharer,node)]
    p = abs(popularity[sharer]-popularity[node] + 1)/length
    probabilities.append(p)
  probabilities = [p/sum(probabilities) for p in probabilities]

  if len(available) != 0:
    diffusivity = random.choice([0,1])
    if diffusivity == 1:
      listener = random.choices(available,probabilities)[0]
      return listener
    else:
      return None
  else:
    return None

def spread_gossip(local_know,state,popularity,lengths):
  current_state = state.copy()
  current_lengths = lengths.copy()
  for node in list(G.nodes):
    if state[node] == "Green":
      listener = pick_listener(node,state,popularity,lengths)
      if listener != None and current_state[listener] != "Green":
        current_state[listener] = "Green"
        local_know += 1
        edge = None
```

```
        if listener < node:
          edge = tuple([listener,node])
        else:
          edge = tuple([node,listener])
        current_lengths[edge] =
round(current_lengths[edge]*math.exp(-(abs(popularity[node]-popularity[listener]+1))/
1),3)

  return local_know, current_state, current_lengths

global_know = 1
knows = [global_know]
time = 0
mean_lengths = []

print(f"Time = {time}")
print(global_know)
print(state)
print(edge_length)
mean_length = sum(edge_length.values())/len(edge_length)
mean_lengths.append(mean_length)
print()
time += 1


while global_know < len(list(G.nodes)):
  print(f"Time = {time}")
  global_know, state, edge_length =
spread_gossip(global_know,state,popularity,edge_length)
  knows.append(global_know)
  mean_length = sum(edge_length.values())/len(edge_length)
  mean_lengths.append(mean_length)
  print(global_know)
  print(state)
  print(edge_length)
  print()
  time += 1
```

Plot mean lengths

```
plt.plot(range(time),mean_lengths);
```

Plot Instantaneous and Cumulative Flux

```
flux = [0]
instantaneous = [0]
for i in range(1,len(knows)):
  flux.append((knows[i])/i)
  instantaneous.append((knows[i]-knows[i-1])/i)
plt.plot(range(time),flux,label="Cum_Flux_100")
plt.plot(range(time),instantaneous,label="Instant_Flux_100")
plt.legend();
```

****

CFA Report

ORIGINALITY REPORT

5%	4%	3%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Imperial College of Science, Technology and Medicine Student Paper	1%
2	www.jstott.me.uk Internet Source	1%
3	V.S. Patwardhan. "Diffusion and sorption in zeolites—I. A Markov process formulation", Chemical Engineering Science, 1989 Publication	1%
4	www.nesc.ac.uk Internet Source	1%
5	Submitted to University of Southern California Student Paper	<1%
6	Submitted to Heriot-Watt University Student Paper	<1%
7	origin.geeksforgeeks.org Internet Source	<1%
8	Submitted to The University of Manchester Student Paper	<1%

9	Mehran Mesbahi, Magnus Egerstedt. "Graph Theoretic Methods in Multiagent Networks", Walter de Gruyter GmbH, 2010 Publication	<1%
10	Submitted to Coventry University Student Paper	<1%
11	gmuhumandimensions.com Internet Source	<1%
12	stackoverflow.com Internet Source	<1%

Exclude quotes	On	Exclude matches	< 7 words
Exclude bibliography	On		

CFA Report

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17