MH2401 - Algorithms and Computing III

Group 204: Ashiq Muhammad, Luo XinYi, Teo Shi Min Jasmine, Edmund Chew, Chua Swee Kiat (Cai Ruijie), Ryan Soh Yong Kian

November 21, 2018

Abstract

This project aims to simulate the spread of information over the online social network, Facebook. In this report, we first look at the various models, such as the Erdös-Rényi Model, Watts–Strogatz model, Barabási-Albert Model, Triangle Friends Model and the Mediation-Driven Attachment Model.

We then analysed in detail and compared the properties of these models to find one that resembles Facebook the most. We then concluded that the Triangle Friends Model models after Facebook the best, therefore using it to simulate the spread of information in the Susceptible - Infectious - Recovered (SIR) model over the network. Next, using the data from Google trend for '377A', we find the optimal parameters of the model by using the Particle Swarm Optimization (PSO), where we minimised the difference between data from our SIR model and that from Google Trend. It was computed that the optimal probability of infection is 0.998 and the optimal probability of recovery is 0.136. Lastly, we also looked at the Susceptible - Infectious - Recovered - Susceptible (SIRS) model to see if it will be able to simulate the spread of information more closely.

Contents

Chapter 1

Random Networks

1.1 Overview

We will be comparing the 5 following models to determine which will be the best model of Facebook:

- Erdös-Rényi Model

- Watts-Strogatz Model

- Barabási-Albert Model

- Mediation-driven Attachment (MDA) model

- Triangle Friends Model (TFM)

All models generate graphs with differing properties. In this report, we examine 3 main properties: clustering coefficient, average degree, average path length.

1.1.1 Erdös-Rényi Model

The Erdős–Rényi (ER) Model takes the parameters $G(n, p)$, where n is the number of nodes and p is the probability that each edge is connected. This means that the ER model starts with an initial amount of nodes n, and between any 2 nodes, it is given a probability p that an edge is formed.

Clustering Coefficient

For the ER model, the probability that an edge between any two nodes is present is independent of the fact that two randomly selected nodes connected to a particular node, are connected. Hence, the clustering coefficient of the ER random network is equal to p. The model results in clustering coefficients that are too small and thus, the ER model is a poor predictor of clustering compared to real networks, such as Facebook.

Algorithm

A Erdös-Rényi graph is simple to generate. By using the built in networkx function, where n is the number of nodes while p is the probability of each edge creation.

The graph has a size of $n = 100$, and probability of an undirected edge forming between any two nodes $p = 0.24$.

An example of an Erdös-Rényi graph is as follows:

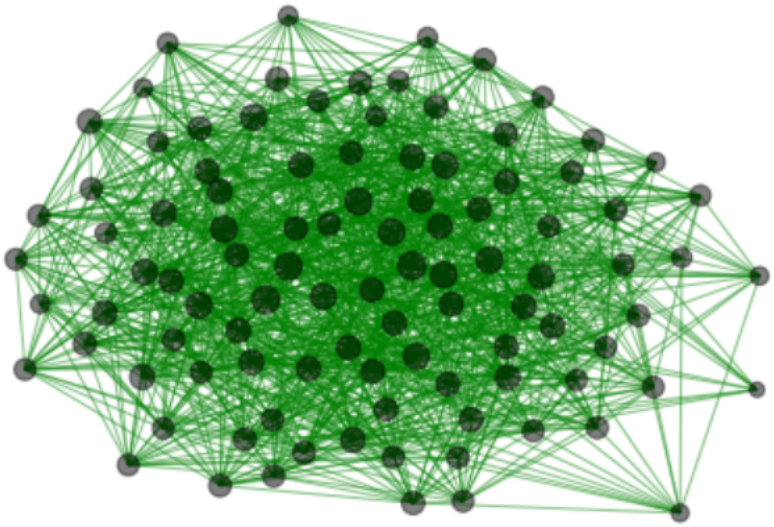


Figure 1.1: An example of an Erdös-Rényi graph with 100 nodes

1.1.2 Watts-Strogatz Model

The Watts–Strogatz (WS) model is a random graph generator that produces undirected graphs with small-world properties, represented by short average path lengths and high clustering coefficient[2]. The WS Model takes the parameters $G(n, k, p)$, where n is the number of vertices, k is the mean degree of a vertex with an even value, since each vertex connects to equal number of vertices on the left and right sides, and p is the probability that the vertex is being rewired by replacing (u, v) with (u, v'), where v' is chosen among all available vertices, and outputs a random graph G.

Degree Distribution

The WS model is beneficial for its ability to generate networks with short average path length between nodes, however it yields unrealistic degree distribution. In real life, relationships between neighbours will not follow the distribution WS model displays but rather has the preferential attachment rule which we will later observe when analysing the Facebook data.

Clustering Coefficient

For a WS model, a higher p will result in a lower clustering coefficient as well as a lower average path[2]. On the contrary, with a lower p, the structure of the network will resemble the original ring lattice closely, having a higher clustering coefficient and higher average path. Hence, using the WS model, the clustering coefficient can be modified to obtain the required amount without affecting the number of edges formed between the fixed number of nodes.

Algorithm

By using the built in networkx function for WS, where n is the number of nodes, k is the number of neighbours each node connects to in the ring lattice and p is the probability of each edge being rewired, we are able to generate the following graph:

By using the built in networkx function for BA, where n is the number of nodes, k is the number of neighbours each node it joint to in the ring lattice and p is the probability of each edge being rewired, we are able to generate the following graph:

From Fig 1.2, we see a example of a Watts-Strogatz Model with $n = 100$, $k = 44$, $p = 0.24$

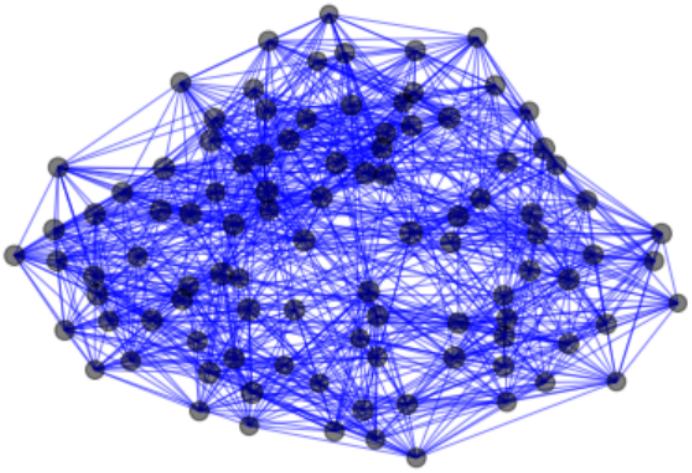


Figure 1.2: An example of a Watts-Strogatz graph with 100 nodes

1.1.3 Barabási-Albert Model

The Barabási-Albert (BA) model is an algorithm meant to produce scale-free networks using a preferential attachment mechanism. The model gives an undirected graph. The BA Model takes the parameters $G(n, k)$, where n is the number of nodes and k is the number of edges to between the new node and existing nodes.

Scale-free Network

Scale-free network is a network with degree distribution that follows the power law. The fraction of $P(k)$ nodes in the network having k connections to other nodes goes for large value of k as $P(k) \sim k^{-\gamma}$ where $2 < \gamma < 3$. [3]

Preferential Attachment

Preferential attachment is a probabilistic mechanism: a new node is free to connect to any node in the network. In each time step, a new node is created. It is then connected to a maximum of m existing connected nodes with probability

$$p(k_i) = \frac{k_i}{\sum_j kj}$$

where k_i is the degree of node i over the sum of all pre-existing nodes j. The following probability is theoretically similar to that executed with the built in BA functions. Due to preferential attachment, if a new node has to choose between a degree-two and a degree-four node, it is two times more likely that it connects to the degree-four node. This results in the phenomenon known as "the rich get richer".
This means that the new nodes establish links to the well connected existing ones preferentially.

Properties of the Barabási-Albert Model

1) Growth: The number of nodes are not stagnant and increases over time.

2) Preferential attachment: The higher the degree of the node, the higher the likelihood that a new edge will be connected to the node.

Algorithm

By using the networkx function, nx.barabasi_albert_graph(n, m), where n is the number of nodes and m is the number of edges to attach from a new node to existing nodes.
The Fig 1.3 has a size of $n = 100$, initial seed size of 22 nodes and maximum degree of each node $m = 44$.
An example of a Barabási-Albert graph is as follows in figure 1.3:

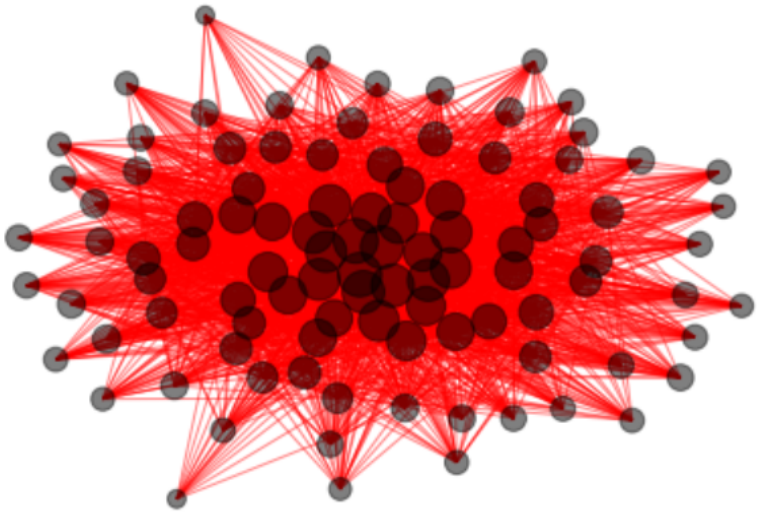


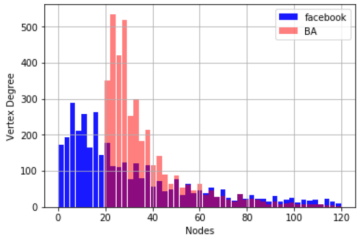
Figure 1.3: A Barabási-Albert Graph with 100 nodes

1.1.4 Triangle Friends Model

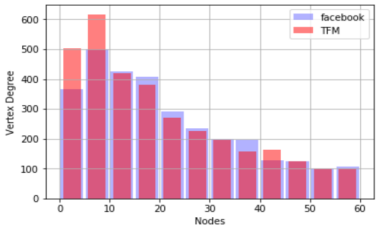
The Triangle Friend Model is a modification of the Barabási-Albert Model. The Triangle Friend model takes the parameter $G(n, m, p)$, where n is the desired network size; the total number of nodes, m is the number of edges connected to the new node that is added in every iteration and p is the maximum probability allowed of forming connection with new source, target and the targets friend. To fully under the Triangle friend model, we first have to understand the Holme and Kim's model.

Literature Review: Holme and Kim's Algorithm

The average clustering has a hard time getting above a certain cutoff that depends on 'm'. This cutoff is often quite low. The transitivity (fraction of triangles to possible triangles) seems to decrease with network size. It is essentially the Barabási–Albert (BA) growth model with an extra step that each random edge is followed by a chance of making an edge to one of its neighbors too (and thus a triangle). This algorithm improves on BA in the sense that it enables a higher average clustering to be attained if desired. ref : [1]



(a) Facebook data's and Barabási-Albert degree distribution



(b) Facebook data's and Triangle Friends model degree distribution

Figure 1.4: Comparison of Degree Distributions

Triangle Friends Model

From the understanding of HK model and BA model we try to improve on them. First we compare the difference between the BA and Facebook model [Fig 1.4].

Comparing the degree distributions of Facebook and the Barabási-Albert Model, we see that the end tails of the Barabási-Albert Model does not fit well with Facebook's. This observation drives us to find another model which fits Facebook's degree distribution well, yet exhibits small world and scale-free properties as in the Barabási-Albert Model.

We can see that the Triangle Friend model indeed compares preferably to the Facebook data's Model in comparison to the Barabási-Albert model.

Algorithm

1. First we initialise the complete graph with m nodes and each node with $m + 1$ edges.
2. Identify the target randomly from the existing nodes in the model.
3. After which we list the neighbours of the respective target.
4. Using the list of neighbours, each of the neighbour of the target node is connected to the new node with probability $p = 0.48$.
5. The steps 2 - 4 are repeated until the total number of nodes reaches n.

Evaluation

The TFM model tries to reduce the effects of the preferential attachment rule which are present in the above mentioned MDA and BA models by randomising the probability of forming new edges. We introduce this random-ness as sometimes in Facebook the connections formed are indeed random. As the number of nodes introduced increases there are more existing nodes present in the model, hence the probability connecting to one specific node is randomised due to the random choice function allocated in the algorithm.

p-value: the maximum probability of forming connections with the target's friends. After much calibration, we have decided to let $p = 0.48$. As this is the optimum probability to form successful edges so that after 4039 nodes are introduced we do indeed have a total edge equivalent to the Facebook model; approximately 88,000.

Below we can see an example of Triangle Friends Model with $n = 100$, $m = 44$, $p = 0.48$. From the model itself we can see the formation of triad, i.e. a connection between the new node, the target and the target's friend.

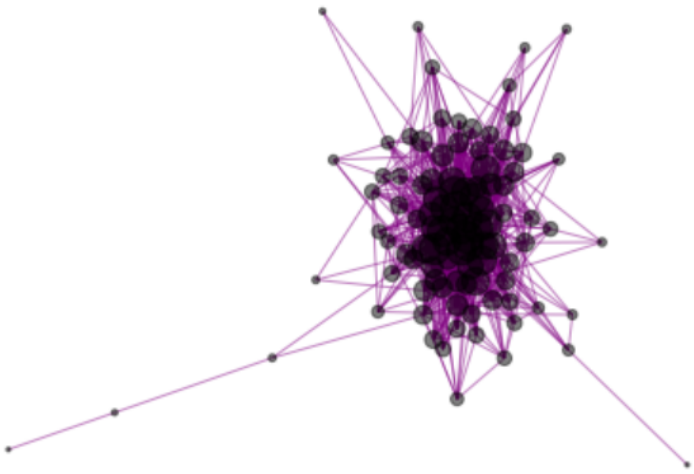


Figure 1.5: Example of the Triangle Friends Model

1.1.5 Mediation-Driven Attachment Model

The Mediation-Driven Attachment model (MDA Model) seems to follow the preferential attachment rule implicitly, instead of blatantly expressing this rule. MDA Model takes in two parameters, n and m. The parameter n refers to the desired network size, m refers to number of neighbours of mediator to be connected to the new node.

Preferential Attachment

Similar to the BA model, the MDA model appears to follow the preferential attachment mechanism, which is essentially the rich get richer mechanism.

Evaluation

Similarly, the MDA model follows the preferential attachment rule, where for small m is rather super-preferential as it gives rise to winner-takes-all (WTA) effect, where the hubs are regarded as the winners.[5]

However, as m increases, the MDA growth rule will exhibit the simple PA rule as the WTA effect transits to winner-takes-some (WTS) effect[5]. Hence, for MDA to better model Facebook, we decided to use $m = 22$. However, by observing Figure 1.6, the MDA model only displays similar trends in the degree distribution as the Facebook model at the tail, while the front holds figures that are too small to be depicted here (approximately 22 to 40 nodes). Hence, we can deduce that the MDA model is not as complete in modeling after the Facebook model, which motivates us to seek for a better model.
An example of a Mediated Driven Attachment graph is shown in Fig 1.6:
The graph has a size of $n = 100$, number of neighbouring nodes of the randomly choosen node $m = 22$.

Refering to Fig 1.7, can see that the BA and the MDA are very similar in degree distribution. From, the following graph showing their resemblance. The both clearly follow similar distribution.

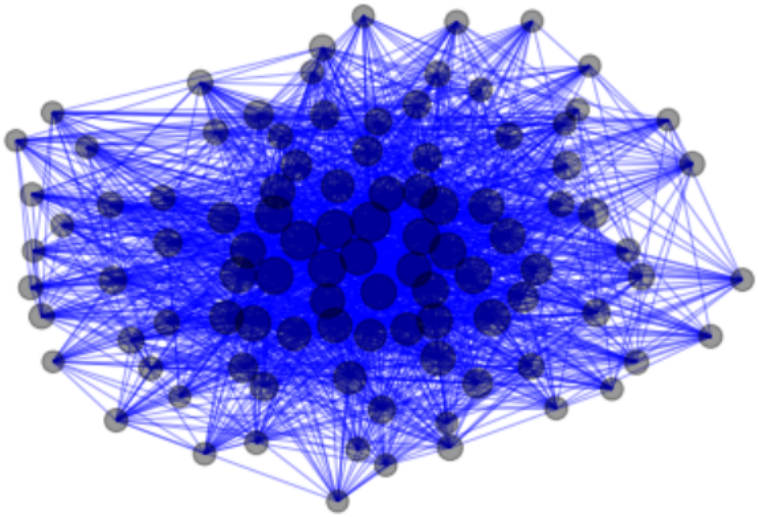


Figure 1.6: An example of a MDA graph with 100 nodes

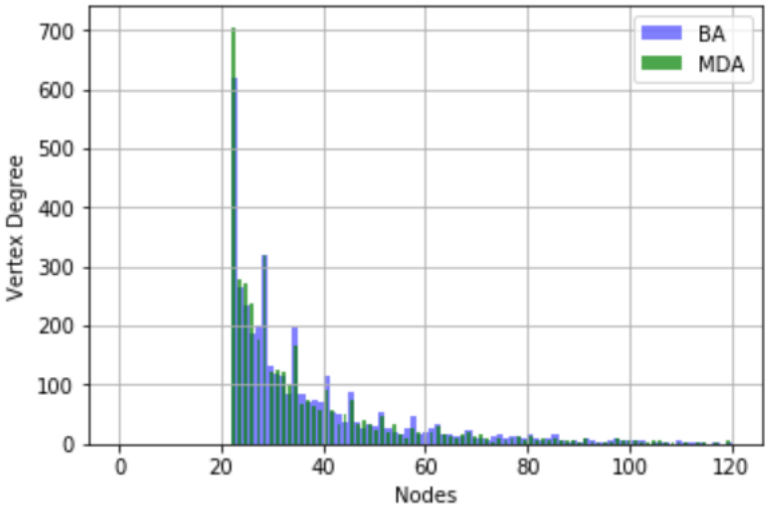


Figure 1.7: Histograms comparing the Degree Distribution of MDA and BA model

Chapter 2

Modelling Facebook

For us to model Facebook using the random networks, we must first understand the characteristics of a Real-Network.

2.1 Parameters used in Analysis of a Real-Networks [7]

2.1.1 Path Length

A Real network has to be small. First we measure typical separation between two generic nodes in a graph G, the characteristic path length L coefficient :

$$L(G) = \frac{1}{N * (N-1)} \sum d_{ij}, i, j \subseteq G$$

The efficiency between node i and j, ϵ_{ij} , is assumed to be inversely proportional to the shortest path length, i.e $\epsilon_{ij} = 1/d_{ij}$. When there is no path linking i and j it is assumed $d_{ij} = +\infty$ and, consistently, ϵ_{ij}

2.1.2 Clustering

For instance, in a social system, there is a high probability that two individuals linked by acquaintance have a third acquaintance in common: people show their inclination for self-organization in small communities

within the system. Hence a Real-Network has a high clustering, i.e Facebook.

2.1.3 Low cost

In quantitative terms, the cost for a graph G, composed by N vertices and K edges, can be defined as the normalized number of edges.

$$Cost(G) = \frac{2 * K}{N * (N - 1)}$$

Note: The cost will be similar for the models compared later due to the similar number of nodes,N, and number of edges,K.

These 3 propertises are consistant in a Real network as mentioned by P.Crucitti.

For the Facebook data provided we were able to derive number of vertices, number of edges and the average degree of each node, given by 4039, 88234 and 43.6910 respectively. We will model Facebook's data by plotting the random network graphs with number of edges as close to that of Facebook's(≈ 88000).

2.2 Parameters of some Random Networks

2.2.1 Modelling with Erdös-Rényi Model

The parameters of the model are given by the number of nodes $n = 4039$ and the probability of an undirected edge forming between any two nodes $p = 0.0108$, where p is derived using the following formula.

$$p^M (1 - p)^{nC2-M}$$

The Erdös-Rényi model was a stepping stone to many other models to come. [8]

2.2.2 Modelling with Watts-Strogatz Model

The parameters of the model are given by the number of nodes $n = 4039$, the mean even degree $k = 44$, and the probability of rewiring each vertex $p = 0.0614980$. The Watts-Strogatz model contains n desired nodes, with mean even degree k and $\frac{nk}{2}$ edges.

2.2.3 Modelling with Barabási-Albert Model

The parameters of the model are given by the number of nodes $n = 4039$ and number of edges to attach from a new node to existing nodes $m = 22$. The Barabási-Albert Model coupled with the Holm and Kim's Model set the foundation for the Triangle Model explained below. [1]

2.2.4 Modelling with Triangle Friends Model

The parameters of this model are given by the number of nodes $n = 4039$, average degree $m = 44$ and probability of adding a triangle $p = 0.48$.

2.2.5 Modelling with Mediation Driven Attachment Model

The parameters of the model are given by the number of nodes $n = 4039$, number of neighbouring nodes of the randomly chosen node $m = 22$.

2.3 Analysis of Random Networks using Facebook data

Graphs	Number of Nodes	Number of Edges	Average Edges	Average Path Length	Clustering Coefficient
Facebook	4039	88234	22.845	3.693	0.609
Erdös-Rényi	4039	88072	21.812	2.607	0.010974
Watts-Strogatz	4039	88858	22.000	3.222	0.622
Barabási-Albert	4039	88374	21.88	2.509	0.031
MDA	4039	88605	21.93	2.499	0.0476
Triangle Friends Model	4039	86772	21.483	2.898	0.369

Firstly, when looking at the Erdös-Rényi model, we can see that even though the average Edges (22.845) of the graph is very close to that of Facebook Models, However it lacks the 2 main criteria as mentioned in 2.1 to be considered comparable to a real network such as Facebook. It neither has a comparable Average Path length nor a high clustering coefficient. Therefore we can rule it out from our consideration to which model represents Facebook best.

Next, we look towards the Watts-Strogatz model. This model excels in comparison when it comes to a high clustering coefficient. However, the unrealistic degree distribution and its inability to reflect the variance of the vertex degrees of a real social network accurately shifts our interest from this model. This can clearly be seen in Fig 2.1.

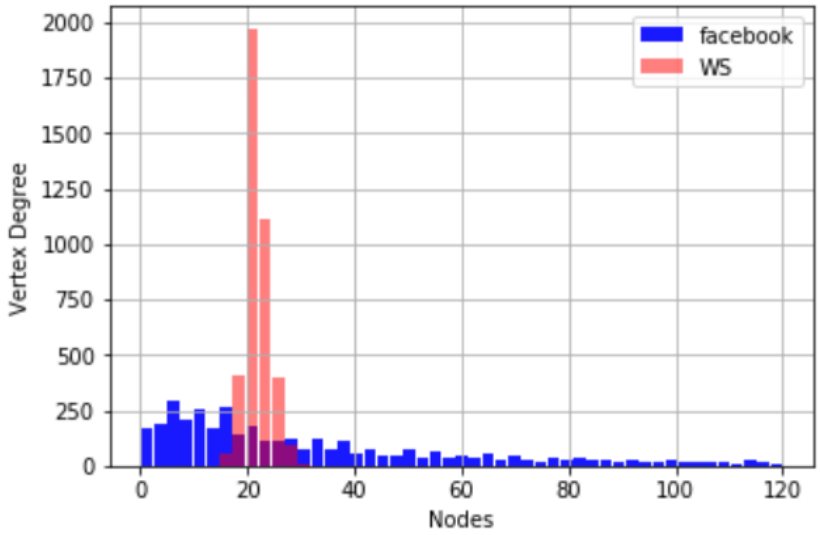


Figure 2.1: Histogram of Facebook model compared to WS model

The models which exhibit a growth functions with a scale-free network distribution similar to the Facebook model would be the Barabási-Albert model and the Mediation Driven Attachment (MDA) model. This allows the Barabási-Albert model to imitate hubs found in the Facebook data. Presence of these hubs in the Barabási-Albert model gives rise to the small-world property as seen from the low average path length of 2.509, which is a relatively close average path length of Facebook. The small-world property is a regular occurrence in real world social networks, thus making the Barabási-Albert model a suitable candidate in modeling Facebook.

Similarly, the MDA follows a rich gets richer mechanism (preferential attachment). Hence, notice that MDA and BA are very similar. These mechanism does somewhat follow a similar ideology to that of Facebook as it is reasonable to assume that people with more friends is likely to make more friends due to a larger number of Mutual friends present.

In spite of both the possible reasons these two models resemble the Facebook model they both fail in one aspect, which is the clustering coefficient. Both the BA and MDA models have a relatively small clustering coefficient of 0.031 and 0.0476 respectively which contradicts the characteristics of a Real Network.

Hence, we look towards our new model the Triangle Friends Model which portrays a growth model which

results in a small average path length and a high clustering coefficient. With the Triangle Friends Model we seek to form a vertex degree distribution similar to that of the Facebook model. From the Fig 2.2, we can see that the Triangle Friends Model fits the head and tail of the Degree distribution to that of Facebook model.

Therefore, in summary, we believe that the Triangle Friends model is best suitable to mimic that of the Facebook model as the methodology of making new friends is more reasonable in explaining how Facebook works.

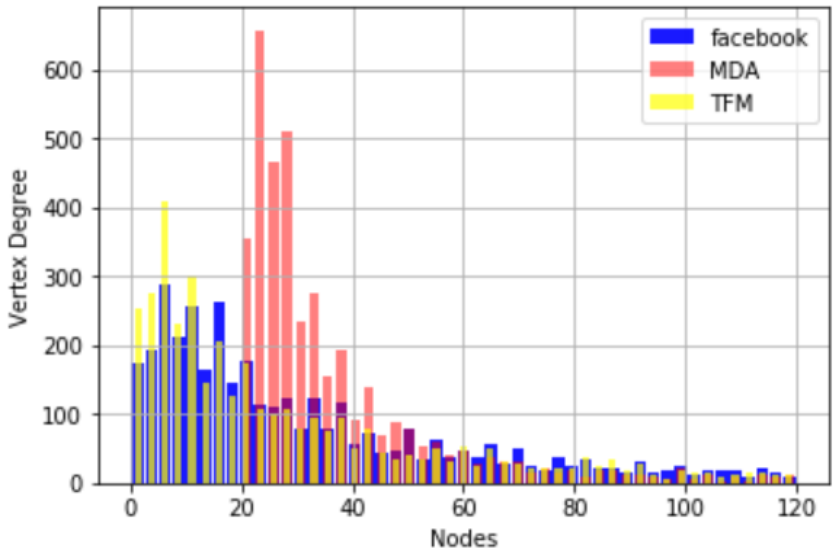


Figure 2.2: Histogram of comapring MDA, TFM and FB

Chapter 3

Spread of information over a network

3.1 SIR Model

Converting the SIR model to model the spread of information over Facebook, we let there be a network whose nodes represent people.

We will thus have the following 3 states:

1. Succeptible - does not know about the event
2. Infected - knows about the event and manifests her interest by doing Google search
3. Recovered - knows about the event but does not care any more

In this model,the time is discrete and some nodes are intially infected and the rest are susceptible. At each moment, susceptible nodes may get infected from their neighbours with probability p and infected nodes may recover and become immune with probability q.

3.1.1 Algorithm

1. We initialise a number of people who are infected (e.g. 4 out of 400)

2. Using a variable probability found by PSO, there is a chance for uninfected people to turn infected

3. Using another variable probability found by PSO, there is a chance for the currently infected to be permanently cured.

4. Both steps take place once each day, repeated for a total of 31 days for our sample code.

5. Interest would be calculated each day by the total number of infected people.

3.2 Particle Swarm Optimisation (PSO)

We then use the Particle Swarm Optimization to find the optimal parameters for the SIR model so that we can get a best fit curve to the Google trend data, using the objective function $= \sum_0^{31}(N(t) - S(t))^2$.
In this equation, S(t) is the ratio of number of Google searches about the event that we are interested in, while N(t) refers to the array obtained from the SIR model. This is the objective function that will be used in the particle swarm optimisation. When the objective function is minimized, we can then find the optimum value of p and q.

3.3 The Event

We will be looking at the search result "Penal Code 377A".

Penal Code 377A is the legislation that criminalises sex between 2 consenting adult men [4]. As PinkDotSG, a nonprofit organisation that supports LGBT rights in Singapore started petitioning for the repeal of the penal code 377A on 9th September 2018, we have decided to look at the interest in 377A from 7th September 2018 to 7th October 2018 to capture the spike in interest towards 377A and the gradual decrease as well.

Then using the PSO function as mentioned above, we will thus be able to find values for the parameters of our SIR and TFM models that best fit the profile of the Google Search which are $p = 0.998249173979348$ and $q = 0.136586503988429$.

Plotting the results, we have the following graph:

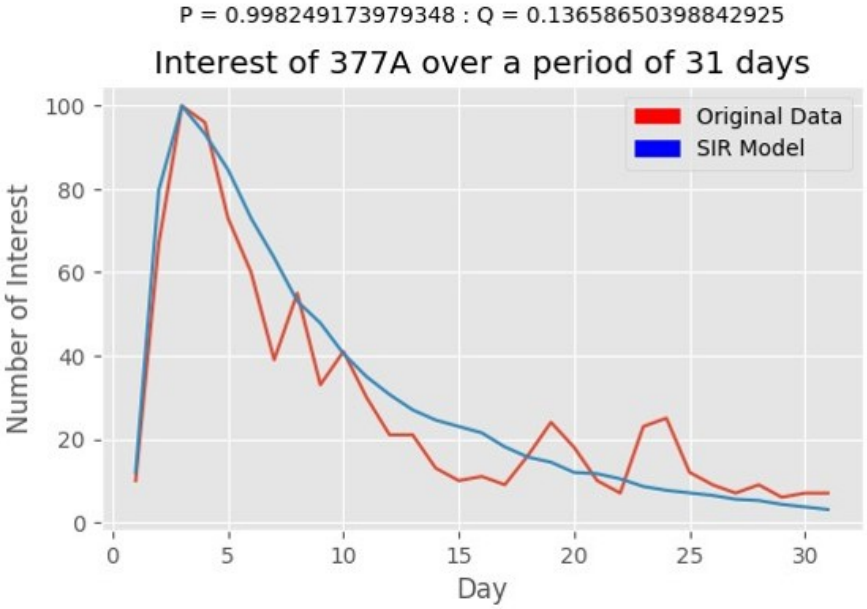


Figure 3.1: SIR model graph against Facebook data graph

Chapter 4

Extras

Below are the extra variations of the SIR model, the SIRS model and the SEIRS model, to try to see if we are able to get our graph to follow the original data more closely, at the local peaks towards the end of the above graph.

4.1 SIRS Model

This is a variation of the SIR model, where it allows recovered individuals return to a susceptible state. Hence, the SIR model becomes an SIRS model (Susceptible - Infectious - Recovered - Susceptible) model, where recovery does not confer lifelong immunity and individuals may become susceptible again.

Thus the algorithm required is similar to the SIR model, where $Permanently\_Cured$ is replaced by $Temporarily\_Immune$, an similar array with all ones initially, and Y_chance representing the probability of immunity waning, or rather, the cured population becoming susceptible again.

4.2 SEIRS Model

This is another variation of the SIR and SIRS model, where individuals experience incubation period such that the individual is infected but not yet infectious. Hence, the SIRS model becomes an SEIRS model

(Susceptible - Exposed - Infectious - Recovered - Susceptible) model, where the incubation rate, σ, is the rate of latent individuals becoming infectious.

Thus the algorithm required is similar to the SIR and SIRS model, with addition of an initial all zero array $Total\_Exposed$, and E_chance representing the probability of the exposed patient becoming infectious.

4.3 Evaluation

Even though the SIRS and SEIRS model does follow the graph more closely as compared to the SIR model, it still fails to be a perfect fit. The reasoning behind the failure of perfect fit is due to the fact we assume infection only happens via neighbours on Facebook.

The peaks shown in the original data indicate rather unusual and sharp peaks. As the SIRS models after disease spread, there should not be sudden spikes in viral infections but rather gradual rises/dips.

In the epidemiology equivalent, this is similar to introducing a large group of newly infected people into the population at Day 16 and Day 21, or in a more realistic context, a few highly infectious people traveling into a densely populated city filled with susceptible people. susceptible people. Moreover, the SIRS model is only accurate when its assumptions hold.

One of the assumption is that the members of the population mix homogeneously; where everyone has the same interactions with one another to the same degree[8]. As this assumption fails to hold true in real world context, in both epidemiology as well as Interest of 377A, the SIRS model therefore does not produce a perfect fit graph.

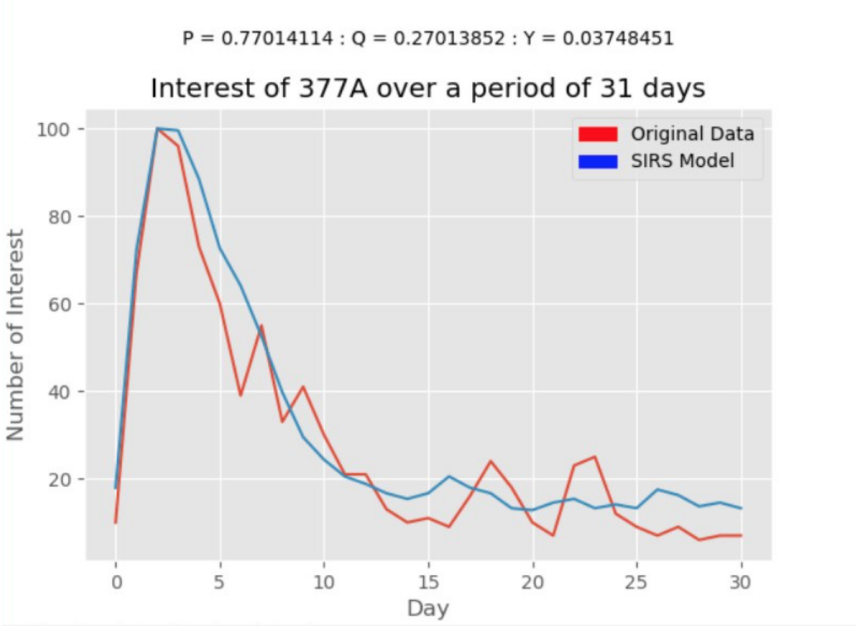


Figure 4.1: SIRS model graph against Facebook data graph

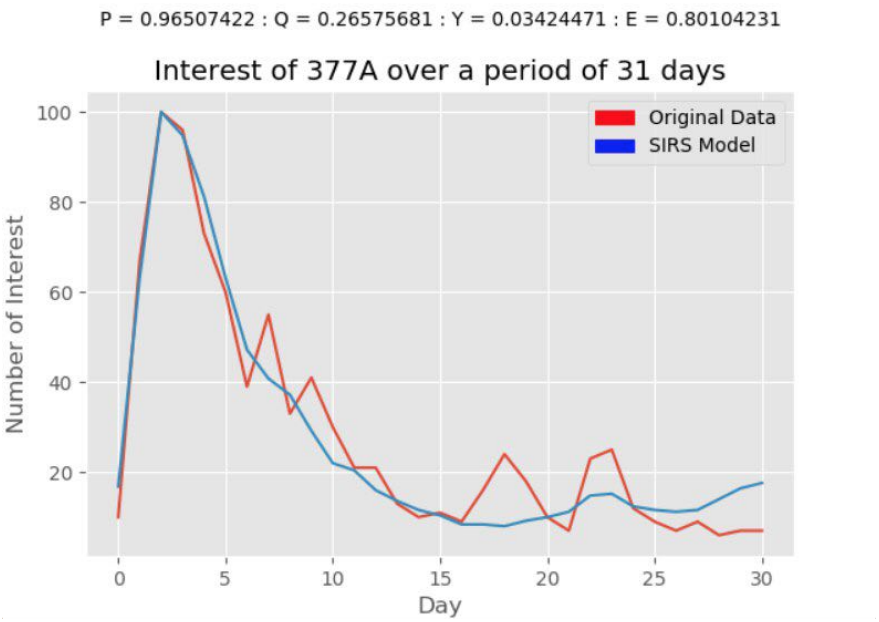


Figure 4.2: SEIRS model graph against Facebook data graph

Chapter 5

References

[1Qiang Guoa, T. Z.-G.-J. (2006). Growing scale-free small-world networks with tunable assortative coefficient. Science Direct.

[2]Watts, D., and Strogatz, S. (1998). Collective dynamics of 'small-world' networks.

[3]Onnela, J. -P., Saramaki, J., Hyvonen, J., Szabo, G., Lazer, D., Kaski, K., . . . Barabasi, A. -L. (2007). Structure and tie strengths in mobile communication networks. Proceedings of the National Academy of Sciences

[4] (2007). Singapore. Parliament. Parliamentary debates. Parliament of Singapore website: https://sprs.parl.gov.sg/search/report.jsp?currentPubID=00004744-WA.

[5] Md. Kamrul Hassan, L. I. (n.d.). Degree Distribution, Rank-size Distribution, and Leadership in Persistance in Mediation-Driven Attachment Networks. Department of Physics, University of Dhaka, Dhaka-1000,

[6] CRUCITTI, P. (2011). COMPLEX SYSTEMS: ANALYSIS AND MODELS OF REAL-WORLD NETWORKS

[7] Bollobás, B. (n.d.).(1984) The evolution of random graphs. Trans. Amer. Math. Soc. 286 , 257-274 .

[8] Teri Johnson. (2009). Mathematical Modelling of Diseases: Susceptible-Infected-Recovered (SIR) Model. University of Minnesota, Moris.

[9] TFM Source Code, stated below with the code.

Chapter 6

Appendix

6.1 Code for Random Networks

```
import numpy as np
import networkx as nx
import random
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
#Nice Plot Network of G, where G is any of the random networks analysed


def nice_plot_network(G):
    v_degrees = np.array(list(dict(nx.degree(G)).values()))
    nx.draw(G, node_size = 2.5 * v_degrees,
             node_color = range(len(v_degrees)),
         alpha = 0.2, cmap = plt.cm.winter)
    return None
```

```
#Plotting graph of G, where G is any of the random networks analysed


def print_network_chars(G):
    vertex_degrees = list(dict(nx.degree(G)).values())
    plt.hist(vertex_degrees,
         bins = np.linspace(np.min(vertex_degrees),
                            1 + np.max(vertex_degrees), 20),
         facecolor='blue', alpha=0.75, rwidth = 0.9)
    plt.title("Vertex degree distribution of graph")
    plt.grid(True)
    plt.show()
    n = nx.number_of_nodes(G)
    m = nx.number_of_edges(G)
    CC = nx.average_clustering(G)
    print("|V| =", n)
    print("|E| =", m)
    print("Average degree is", 2 * m / n)
    print("Average path length=", nx.average_shortest_path_length(G))
    print("CC =", CC)
    return None
```

6.1.1 Code for Erdös-Rényi model

```
def ER(n,p):
    nx.erdos_renyi_graph(n,p)
    return
```

```
ER = ER(n,p)
nice_plot_network(ER)
```

6.1.2 Code for Watts-Strogatz Model

```
def WS(n,m,beta):
    nx.watts_strogatz_graph(n,m,beta)
    return


WS = WS(n,m,beta)
nice_plot_network(WS)
```

6.1.3 Code for Barabási-Albert model

```
def BA(n,k):
    nx.barabasi_albert_graph(n,k)
    return


BA = BA(n,k)
nice_plot_network(BA)
```

6.1.4 Code for Triangle Friends Model

```
def TFM_graph(n, m, p, seed=None, k = 2):
    """Makes a small world graph with a lognormal degree distribution."""
```

```
if m < 1 or  m+1 >= n:
    raise nx.NetworkXError(\
          "FOF network must have m>=1 and m+1<n, m=%d,n=%d"%(m,n))
if seed is not None:
    random.seed(seed)

# start with a completely connected core
G = nx.complete_graph(m+1)



G.name="TFM_graph(%s,%s)"%(n,m)

for source in range(len(G), n):
    # choose a random node
    target = np.random.choice(G.nodes())
    # enumerate neighbors of target and add triangles
    friends = list(G.neighbors(target))
    k = len(friends)
    numberoftriangles = np.random.binomial(k, p, size=None)
    # calculate number of triangles based on how many edges
    # the nodes has, the more connected the node,
    # the more triangles to form.
    if numberoftriangles != 0:
        G.add_edges_from(zip([source]*numberoftriangles,
                             random.sample(friends,numberoftriangles)))
        # form triangles based with preferential
```

```
            # attachment to nodes with more edges
        # connect source and target
        G.add_edge(source, target)

    return G

G = TFM_graph(4039,44, p=0.48, seed=21)

# original source of code can be found (https://github.com/AllenDowney/
# ThinkComplexity2/blob/master/code/fof_model.ipynb)
```

6.1.5 Code for Mediation-Driven Attachment Model

```
def MDA(size,m):
    seed = (np.zeros([size,size]))
    #start with a seed of m+1 nodes that each has m edges
    start = (np.ones([m+1,m+1], dtype= int) - np.diag(np.ones(m+1 , dtype = int)))
    seed[0:m+1,0:m+1] = start
    #initialise the iteration of each additional node
    current = m
    check = (current == (size-1))
    while check == False:
        #pick the mediator
        sum_seed = np.where(np.sum(seed, 0) >= m)
        #the coordinate of the mediator node
        stem = np.random.choice(sum_seed[0])
```

```
        #Finding all neighbouring nodes to mediator
        leaf = np.where(seed[stem]==1)[0]
        #Choosing m neighbour nodes to mediator
        n_leaf = np.random.choice(leaf,size = m , replace=False)
        #update the seed for the new node's index
        seed[current,n_leaf] = 1
        seed[n_leaf,current] = 1
        current += 1
        check = (current == (size-1))
    return seed

MDA = MDA(4039, 22)
MDA_Graph = nx.Graph(MDA)
MDA_graph = nx.Graph(MDA)
```

6.1.6 SIR Model

Code for Facebook Graph

```
google=np.loadtxt('data.txt',delimiter = ",", dtype = int, usecols=([1]))
```

Code for SIR model

```
G = TFM_graph(400,4, p=0.24, seed=21)
len(G), len(G.edges())
B = nx.adjacency_matrix(G).todense()

def SIR(beta,gamma):
```

```
# using the best model we found
A = np.array(B)
infect = np.random.choice([1,0], len(A), p=[(google[0]/100),
                                            1-(google[0]/100)])
# create all zero vector for the interest vector
interest = np.zeros(31)

PermanentlyCured = np.ones(len(A))

# Next, it will create a for loop, each loop will first save
# the sum of infected nodes multiplied by the google trend
# into the interest vector.
# The code will loop for 31 times because the data captures 31 days.
for t in range(0,31):
    interest[t] = np.sum(infect)
    if interest[t] == 0:
        # the loop will exit here. This is because if there is no
        # more infected, there can no longer by any further infection
        break

    R = np.random.choice([1,0], len(A), p=[gamma,1-gamma])
    # R is for cured patients.
    # 1 means the patient has been newly cured today, 0 means not cured.
    Pchance = np.random.choice([1,0], len(A), p=[beta,1-beta])
    # Pchance is for infected patients.
    # 1 means the patient has been newly infected today, 0 means not
```

```
    # infected. Infection probability is not accumulative

    PossibleInfectedAdjacentNodes = np.array(np.matmul(A,infect),
                                             dtype=bool)

    CuredInfected = infect * R
    # CuredInfected is the array for curing only infected people
    PermanentlyCured = np.array(PermanentlyCured - CuredInfected,
                                dtype=bool)
    # PermamentlyCured array are people that will never get infected
    # again
    NewlyInfected = PossibleInfectedAdjacentNodes * Pchance
    # Multiply Pchance with PossibleInfectedAdjacentNodes as the
    # only way to get infected is when an adjacent node is infected
    # NewlyInfected will then be the array for infecting only possible
    # adjacent nodes
    TotalInfected = (NewlyInfected + infect)

    infect = np.array(TotalInfected*PermanentlyCured, dtype = bool)
    # Multiplied by PermanentlyCured, as the permanently cured people
    # will never be infected again


# Lastly, the code will find the maximum interest which will
# be used to divided the interest, then multiply by 100 to get
# the percentage interest.
```

```
    interest = interest/np.max(interest) * 100

    return interest
```

6.1.7 Code for Particle Swarm Optimization

```
from pyswarm import pso

def objective_function(X):
#error function
    a = np.sum((SIR(X[0],X[1])-google)**2)
    return a

lb = [0,0] # lower bounds for x and y
ub = [1,1] # upper bounds for x and y

xopt, fopt = pso(objective_function, lb, ub)
print("Minimum point = ", xopt)
print("Minimum value = ", fopt)

pSIR = xopt[0]
# Optimal P value from PSO
qSIR = xopt[1]
# Optimal Q value from PSO

interestsearch = (SIR(pSIR,qSIR))
```

```
style.use('ggplot')
x,y = np.loadtxt('data.txt',
                 unpack = True,
                 delimiter = ',')
#Making space for double header with subtitle
line1 = plt.plot(x,y)
# Original Data Line 1
line2 = plt.plot(x, interestsearch)
# SIR Model Line 2
red_patch = mpatches.Patch(color='red', label='Original Data')
blue_patch = mpatches.Patch(color='blue', label='SIR Model')
# Labels for the lines
plt.legend(handles=[red_patch, blue_patch])
plt.title('Interest of 377A over a period of 31 days')
plt.figtext(.5,.9,'P = {} : Q = {}'.format(pSIR,qSIR), fontsize=10,
            ha='center')
plt.xlabel('Day')
plt.ylabel('Number of Interest')
plt.show()
```

6.1.8 Code for SIRS model

```
def SIRS(beta,gamma,y):
    A = np.array(B)
    infect = np.random.choice([1,0], len(A), p=[(google[0]/100),
                                                1-(google[0]/100)])
```

```
interest = np.zeros(31)


TemporarilyImmune = np.ones(len(A))


for t in range(0,31):
    interest[t] = np.sum(infect)
    if interest[t] == 0:
        break


    R = np.random.choice([1,0], len(A), p=[gamma,1-gamma])


    Pchance = np.random.choice([1,0], len(A), p=[beta,1-beta])


    Ychance = np.random.choice([1,0], len(A), p=[y,1-y])
    # Ychance represent the probability of immunity waning, or rather,
    # the cured population becoming susceptible again.
    # This is mainly for the SIRS model, where the Recovered can become
    # Susceptible again (R -> S)
    # We use probability as immunity waning is a probabilistic event,
    # for both vaccine and recovery from disease.


    TemporarilyImmune = np.array(TemporarilyImmune + Ychance,
                                 dtype=bool)
    # In the SIRS model, there are no permanently cured.
    # This code represent the immunity disappearing from the
    # cured population. Since 1 represents susceptibility,
```

```
        # addition of Ychance will make some the of
        # cured population susceptible again.

        PossibleInfectedAdjacentNodes = np.array(np.matmul(A,infect),
                                                 dtype=bool)

        CuredInfected = infect * R

        TemporarilyImmune = np.array(TemporarilyImmune - CuredInfected,
                                     dtype=bool)
        # TemporarilyImmune array are people that are currently immune
        # but may get infected again

        NewlyInfected = PossibleInfectedAdjacentNodes * Pchance

        TotalInfected = (NewlyInfected + infect)

        infect = np.array(TotalInfected*TemporarilyImmune, dtype = bool)

    interest = interest/np.max(interest) * 100

    return interest
```

6.1.9 Code for SEIRS model

```
def SIR(beta,gamma,y,e):
```

```
A = np.array(B)


interest = np.zeros(31)


PermanentlyCured = np.ones(len(A))


TotalExposed = np.zeros(len(A))


for t in range(0,31):
    interest[t] = np.sum(infect)
    if np.sum(infect) == 0:
        break


    R = np.random.choice([1,0], len(A), p=[gamma,1-gamma])


    Pchance = np.random.choice([1,0], len(A), p=[beta,1-beta])


    Ychance = np.random.choice([1,0], len(A), p=[y,1-y])


    Echance = np.random.choice([1,0], len(A), p=[e,1-e])
    # e represents the probability of the exposed patient
    # becoming infectious. 1 means the exposed patient has
    # successfully incubated the virus and became infectious
    # in the SEIRS model, this represents E -> I
```

```
PermanentlyCured = np.array(PermanentlyCured + Ychance,
                            dtype=bool)
# In the SIRS model, there are no permanently cured.
# This is legacy code from SIR model. This code represent
# the immunity disappearing from the cured population.
# Since 1 represents susceptibility, addition of
# Ychance will make some the cured population susceptible
# again.

PossibleInfectedAdjacentNodes = np.array(np.matmul(A,infect),
                                         dtype=bool)

CuredInfected = infect * R

PermanentlyCured = np.array(PermanentlyCured - CuredInfected,
                            dtype=bool)
# PermamentlyCured array are people that will never get
# infected again

NewlyExposed = (PossibleInfectedAdjacentNodes + TotalExposed)
* Pchance
# NewlyExposed will then be the array for infecting only
# possible adjacent nodes. To be exposed to the virus,
# they are not infectious yet

NewlyInfected = NewlyExposed * Echance
```

```
        # Chance to become infected
        TotalExposed = np.array((NewlyExposed - NewlyInfected)*
                                PermanentlyCured,dtype=bool)

        TotalInfected = (NewlyInfected + infect)

        infect = np.array(TotalInfected*PermanentlyCured,
                          dtype = bool)

interest = interest/np.max(interest) * 100
return interest
```