# Exercise one: Analyzing an offline and online social networks

Question 1 (3 points):

* Find out the node ID of
  + a) highest degree *Answer: S54*
  + b) highest betweenness *Answer:* S37
  + c) highest closeness *Answer:* S37
  + d) highest eigenvector in the Highschool network *Answer:* S110
* Highlight the above nodes in the Highschool network;

|  |  |
| --- | --- |
| Diagram  Description automatically generated | Chart, diagram  Description automatically generated |
|  |  |
| Chart, radar chart  Description automatically generated  **Figure 1.** Highlighted nodes | |

* Explain why these metrics identify the same node or different nodes as the most central one.

Answer: So, the explanation may be given based on two arguments. First: the difference in their computations, and it provides the information about the difference between degree, and eigenvector centralities. With regard to similar “most-central” nodes for closeness and betweenness it is the fact that computations of both involve the shortest distances between the nodes.

What is more, we can think in terms of the interpretations of those centralities for degree, betweenness, closeness and eigenvectors which can be popularity of the node, brokerage of the node, the possibility to reach every other node, and the amount of influence of the node, respectively. We can see that in the networks of the real world the biggest popularity, for instance, does not imply the most influence, brokerage, or accessibility. Influence, on the other hand does not bring other concepts. However, the degree of brokerage is highly related to the ability to reach others, because a “node” would be chosen as broker only if the path through him/her is the shortest one.

Question 2 (5 points):

* Study the correlations between a) degree and betweenness, b) degree and closeness, c) degree and eigenvector *for all the nodes* in the Highschool network;

*Answer:* In Highschool data (see Fig. 2) there is a strong correlation above 0,8 between degree and every other centrality. What should be noted is that for every pair of centralities with degree there is a threshold of the number of connected nodes at which the overall linear trend changes: around 5 nodes in the pair with closeness, at 10 nodes in the pair with betweenness, and around 12 in the pair with eigen centrality.

Chart, line chart

Description automatically generated

**Figure 2.** Centralities for Highschool data

* Study the correlations between a) degree and betweenness, b) degree and closeness, c) degree and eigenvector *for all the nodes* in the Facebook network;

Answer: for this data which is 40 times bigger than the previous one, the correlation between degree centrality and others do not rise above 0.52 for the pair with betweenness, but gets higher above 0.7 for both eigen and closeness centralities. The metioned threshold is also present in that data: Around 250 nodes in the pair with closeness and in the pair with betweenness, and around 175 in the pair with eigen centrality.

Diagram

Description automatically generated

**Figure 2.** Centralities for Facebook data

* From the above results, how well do different metrics correlate with each other? Which centrality metric will you use and why?

Answer: In the first data set the smallest correlations is between degree and eigen, but in the second case it is with betweeness. What is more, from the plots it is evident that closeness and degree receive of the biggest coefficient in both cases.

With regard to the question about what would we prefer to use, we would argue that it is hard to pick one because the most importand informaition is in their relations not in themselves. It also depends on which properties of the network and concepts related ot them we would like to study: popularity, brokerage, saturation, or the “influence ”. As we can see the correlations are different for both datasets, it could be because of the size or the real and virtual origins of the data, in our case is not that important, the important thing is that such conclusions as "your friends have more friends than you" (J. Ugander et al.) can be derived only if we inspect those metrics together.

Question 3 (5 points):

|  |  |
| --- | --- |
| Chart, bar chart  Description automatically generated | Chart, bar chart  Description automatically generated |
| Chart, scatter chart  Description automatically generated | Chart, histogram  Description automatically generated |

**Figure 3.** Degree and shortest path distributions

* For both the Highschool and Facebook networks, calculate the shortest path lengths between every pair of two nodes. How many percentage of nodes can be reached within 6 path lengths? Does “six degree of separation” apply to each network?

Answer: While in the dataset of High school there are more shorter paths (e.g. 1), the Facebook has an outstanding percentage of degree separation of 4; however for both dataset the six degree of separation works for the 90% of nodes. That shows that in this comparison the size of the network does not play that big role, only if we look at the number of other degree separations, but quantiles are similar.

* Study the degree distribution of these two networks, are they similar? Then use degree distribution to explain the degree of separation you answered above.

Answer: The degree distribution in case of Facebook is more skewed, but the prevalence of low connected nodes and an exponential reduction trend coming from less degree to higher degree are the same for both datasets. The six-separation degree phenomenon implies that even if some nodes have small number of connections, they can still be connected to anyone in the world through the other nodes to which they are connected. What is more the already mentioned paper of J. Ugander et al. is also related to these distributions: prevailing majority of the nodes populations is in the left part, and thus the overall probability of a random node to have more connections than other is really small, but the structure of networks and the presence of hubs(e.g. nodes from the first questions) make the “rule” of six degree possible by their brokerage properties.

**Sources:**

J. Ugander, B. Karrer, L. Backstrom, C. Marlow. The anatomy of the Facebook social graph.

Test the above hypothesis by the following steps (Question 4, 4 points):

1. Visualize the network and color the nodes by gender and residential hall, respectively.

Chart, radar chart

Description automatically generated

**Figure 4.** Highschool network coloured by gender

1. Build 8 subgraphs of the original network according to gender and residential hall: 1 subgraph for female student, 1 subgraph for male student, 1 subgraph for students with unknown gender, and 5 subgraphs for students living in residential hall from 1501 to 1505, respectively.

For example, to build a subgraph of all female students, you should keep all the nodes of female students and the edges between them. Other nodes and edges are removed

|  |
| --- |
| Chart, scatter chart  Description automatically generated |
| A picture containing diagram  Description automatically generated |

**Figure 5.** Highschool subgraphs

1. Study the edge density of all the subgraph and compare them to the edge density of the original network. What is your conclusion for the hypothesis?

*Answer:* Here is densities computed persubgraphs

• Density for female friends is 0.0551786521935776

• Density for male friends is 0.0514285714285714

• Density for unknown friends is 0.1

• Density for 1501 friends is 0.12987012987013

• Density for 1502 friends is 0.0980392156862745

• Density for 1503 friends is 0.152046783625731

• Density for 1504 friends is 0.0758620689655173

• Density for 1505 friends is 0.0965909090909091

• Density for the whole network is 0.0536512667660209

Density computation is similar to the computation of the probability of a random graph: we divide the number of experienced outcomes by the number of the possible outcomes, which is completely the same with density formula (the number of present edges by the number of possible edges), so by comparing densities we also can compare the probability of a tie formation. Comparing the network density with all the subgraphs partly confirms the hypothesis for the study halls, as their densities are considerably different. However, for gender this is not the case only for the unknown gender, but these are small in size.

What is more, the simple comparison may be not sufficient for a statistical claim, so we suggest one of the possible methods (ERGM) and its output, which confirm our argument about the hypothesis: the significance of hall impact and insignificance of gender impact.

Call:

ergm(formula = net\_Highschool ~ edges + nodematch("gender") +

nodematch("hall"))

Maximum Likelihood Results:

Estimate Std. Error MCMC % z value Pr(>|z|)

edges -3.16066 0.08244 0 -38.337 <1e-04 \*\*\*

nodematch.gender 0.02325 0.10428 0 0.223 0.824

nodematch.hall 0.96893 0.10751 0 9.012 <1e-04 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Null Deviance: 10232 on 7381 degrees of freedom

Residual Deviance: 3007 on 7378 degrees of freedom

AIC: 3013 BIC: 3034 (Smaller is better. MC Std. Err. = 0)

Question 5 (4 points):

1. Calculate the modularity of the Highschool network if community is merely identified by a) gender and b) residential hall, respectively.

*Answer*:

* gender modularity 4.78267523722059e-05
* hall modularity 0.17559113865932

1. Search the Louvain Community Detection and explain the algorithm in your own words.

The Louvain algorithm is an unsupervised community detection algorithmdivided in 2 phases: Modularity Optimization and Community Aggregation.

1. The algorithm will start by randomly ordering all the nodes in the network in the modularity optimization phase. Then it will optimize modularity by merging communities of nodes as on the Fig. 6 until no significant increase in modularity is reached.
2. After this phase all nodes belonging to the same community are merged in one big node to build a new network. In this network nodes represent communities from the previous phase and edges represent the sum of the weights of the edges between nodes in those communities. (Rita, 2020)

Chart, radar chart

Description automatically generated

**Figure 6.** Louvain Algorithm (Blondel et. al)

1. Use the Louvain Community Detection to identify communities in the Highschool network. Compare the modularity value produced by the Louvain algorithm to those in 1) and explain the reasons for the differences.

*Answer*: Modularity after the Louvain algorithm: 0.701644

|  |  |  |
| --- | --- | --- |
| Chart, radar chart  Description automatically generated | Chart  Description automatically generated | Chart, diagram  Description automatically generated |

**Figure 7.** Highschool network coloured by gender, hall, modularity

The modularity of the Louvain algorithm is way higher compared to the networks identified by gender or residential hall. This is because the Louvain algorithm tries to create communities which maximizes the modularity. As maximum modularity implies such configuration and number of clusters that the density inside communities is the maximum, and the density of links between communities is the smallest (Blondel et al. 2008).

If we would look at the right graph as the ideal type with the maximum modularity, and compare others to it, it is evident that, for example, the purple cluster in the modularity picture consist of different gender and hall members. That means if we divide this purple modularity cluster based on gender and hall, we will separate already dense cluster, and other dense clusters as well.

So overall when we separate the network by either gender or residence, we separate the subgraphs which are already dense, thereby reducing the modularity. Thus, gender and residence separately are not sufficient criteria to detect communities in the graph. We hypnotize that those should be considered as combination together, and maybe with some other feature which is not covered in attributes, maybe it is better to look at the structure of the network, but this goes beyond the range of the current task.

**Sources:**

Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, *2008*(10), P10008.

Rita, L. (2020, April 9). *Louvain Algorithm.* Opgehaald van towardsdatascience: [https://towardsdatascience.com/louvain-algorithm-93fde589f58c\](https://towardsdatascience.com/louvain-algorithm-93fde589f58c\\)

J. Ugander, B. Karrer, L. Backstrom, C. Marlow. The anatomy of the Facebook social graph.

# Exercise two: Network formation models

## Question 6 (3 points):

1. Develop three networks with the same number of vertices (n), but different probability (p); Name them as ER1, ER2, and ER3. Develop the plots of ER1, ER2 and ER3, describe how these three graphs look differently as p increase and explain why.

*Answer:* The value for P indicates how likely a node is connected another node (i.e. its probability). So as the value of P gets higher more connections/edges appear. This also increases the density of the graph, because, as we already said, that probability of the new tie formation in a random graph reflects its density.

|  |  |  |
| --- | --- | --- |
| Shape, circle, polygon  Description automatically generated | Shape, circle  Description automatically generated | Shape, circle, polygon  Description automatically generated |

**Figure 8.** ER1(p = 0.05), ER2 (0.1), and ER3 (p= 0.2) respectively

1. For a large n (e.g., n=1000), study the relation between clustering coefficient of the network and p, and explain the reason for such a relation. (You can use the function of transitivity (graph.object) to calculate clustering coefficient).

*Answer:* Transitivity: 0.2001041 and the chosen P value was 0.2.

As the probability of connection p increases, the transitivity of the network also increases. This is because as more edges are added to the network, nodes become more likely to form triangles, and therefore the transitivity increases. The relationship was already described; thus, we will leave the quote from previous part where it was:

*“Density computation is similar to the computation of the probability of a random graph: we divide the number of experienced outcomes by the number of the possible outcomes, which is completely the same with density formula (the number of present edges by the number of possible edges), so by comparing densities we also can compare the probability of a tie formation.”*

Question 7 (2 points):

Check the clustering coefficient and average path length of the Regular, SW1, SW2 and SW3. Describe the trend of clustering coefficient and average path length as *p* increase. Which graph does mimic the desirable attributes of a small world network?

*Answer*

SW1\_clustering\_coef: 0.6777076

SW1\_avg\_path\_length: 8.394292

SW2\_clustering\_coef: 0.6387643

SW2\_avg\_path\_length: 4.26903

SW3\_clustering\_coef: 0.3706598

SW3\_avg\_path\_length: 2.919309

The clustering coefficient and average path length decreases as P gets higher. Small world networks are a type of network that have both local clustering and short average path lengths between nodes. They are characterized by a few highly connected hubs that are interconnected to many less connected nodes. So, in this case the SW2 graph represents the small world network the best since the average path length is way shorter than in SW1 and the clustering coefficient did not decrease that much.

Question 8 (5 points):

You might realize not every value of p can return you a small-world network that you are looking for. Then a question arises as how can one find the range of p. In the Figure 2 of Watts and Strogatz (1998) (<https://www-nature-com.proxy.library.uu.nl/articles/30918>), it explains how can one decide the range of p by looking at the dynamics between path length and clustering coefficient.

1. Start with a regular network of size=300, nei=6, first reproduce the Figure 2 of Watts and Strogatz (1998). Then provide the range of p which can turn this regular network (size=300, nei=6) into a small-world network.

*Answer:* the range for rewiring probability P is approximately [0.0004, 0.012] see Fig\_ for the picture of the range. As we see on the Fig. \_, if the rewiring probability P goes beyond this range it starts to look like a random graph.

1. Do you need to rewire significant amount of connections to make the network smallworld-like?

*Answer:* No, when we increase the rewiring probability at some point, we see that the clustering coefficient starts to decrease, which starts to make the graph resemble the random graph. So, the desired range is when the clustering coefficient does not decrease, but the diameter drops, which can be seen on the Fig \_.

1. In the paper of Watts and Strogatz (1998), they pointed out that the value of p has two important implications:

*“The idealized construction above reveals the key role of short cuts. It suggests that the small-world phenomenon might be common in sparse networks with many vertices, as even a tiny fraction of short cuts would suffice.”*

*Answer:* When we worked on the “six-degree separation”, we saw that no matter how big the degree centrality of a node is, it is likely with probability of 90% that it will have the maximum shortest path no bigger than 6. That also is connected to the essence of the small world models, meaning that for a particular node it is not important to have a lot of connections to reach any of other nodes in a network, however, it is crucial to have a connection to a node with a shortcut, or a node which has a connection to a node with a shortcut.

Therefore, in order to make a small world network from a one-dimensional lattice we do not need to introduce that many of shortcuts. We just need to put that many that we balance in the tradeoff between tiring to minimize the diameter and not to reduce the clustering coefficient.

*“Thus, infectious diseases are predicted to spread much more easily and quickly in a small world; the alarming and less obvious point is how few short cuts are needed to make the world small.”*

*Use your own words to explain these two implications. For the second implication, connect it with the spread of COVID.*

*Answer:* Here, in case of the virus or information spread, Wattz and Strogatz claim that in the small world network they spread faster than in the regular networks (i.e. with small amount of shortcuts). For the disease to spread two properties are important: how quickly a node falls out of the network(dies) and how infectious it is.

In case the node dies form the disease too fast before it spreads out itself, then the whole network will not be affected. Here comes the importance of the shortcuts and the average diameter of the network. If the diameter is too big, there is a big chance that disease will wipe out the infected nodes or they will recover before it spreads. However, in case of the small world networks, due to the sufficient number of shortcuts, the disease spreads faster and can capture the majority of the nodes.

There is second point, which is the number of shortcuts to be removed for prevention of spreading. In the example of Bearman et. al (See Fig \_) it is enough to remove the bridge between the coloured infected nodes and blank healthy nodes, but such structure is not present in the small world networks. In case of COVID, the ideal and most efficient solution would be to separate everyone and remove all the connections, but it is impossible. The optimal solution from the perspective of the small world models, was to reduce shortcuts to a number, which will restrict the time needed for a virus to spread close to the time after which the disease becomes inactive of not contagious. What the challenge is, how Wattz and Strogatz said it, to find such a number of shortcut reduction.

**Figure 9.** Networks with various rewiring probability

|  |  |
| --- | --- |
| Shape, circle  Description automatically generated | Shape, circle  Description automatically generated |
|  |  |
| Shape, circle  Description automatically generated | Shape, circle  Description automatically generated |

|  |  |
| --- | --- |
| **Figure 10** The distribution of the clustering coefficient and diameter per rewiring probability | **Figure 11** The bridge structure  (Bearman et al 2004) |
| A picture containing text, sky  Description automatically generated | Chart  Description automatically generated |

Question 9 (3 points):

1. What does the power in the above function mean? How can it govern the structure of the network? (Hint: Change the value of power from 0.05, 0.5, 1, 1.5; See how the plot evolves; if you still fail to see the difference, visualize the vertex size according to the edge number, you can consider the code below.)

*Answer:* Power argument in this function means the alpha in the formula of the scale-free model, which is , where is the probability of the of a new node to connect to node i. The interpretation of this parameter is the preferential attachment mechanism, meaning that if alpha is more or equal than one, then the more degree the node has, the more probable that a new node will connect to it. That is called the super linear probability dependency, and in case of alpha being less than one, it is called sublinear probability dependency; in such case we do not have preferential attachment, but rather get a randomized network.

1. For two networks with a power of 0.5 and 1.5, respectively, what will be their resilience for 1) random attack, and 2) targeted attack? (the meanings of ‘random attack’ and ‘targeted attack’ are the same as what is mentioned in Lecture 6, scale-free network)

**Figure 10.** Networks with various scale (i.e. alpha parameter)

|  |  |
| --- | --- |
| Chart, scatter chart  Description automatically generated  Diameter = 12 | Chart, scatter chart  Description automatically generated  Diameter = 12 |
| Chart, scatter chart  Description automatically generated  Diameter = 8 | Chart  Description automatically generated  Diameter = 4 |
| Chart, scatter chart  Description automatically generated | Chart, scatter chart  Description automatically generated |
| Diameter = 12, N components = 1 | Diameter = 10, N components = 8 |
| Chart  Description automatically generated | Chart, scatter chart  Description automatically generated |
| Diameter = 8, N components = 2 | Diameter = 5, N components = 27 |
|  |  |

So, when we apply Random Failure, a node is removed randomly, when we apply targeted Attack, the most connected node is removed. For both the Scale free network (i.e. the one with alpha = 1.5) and the Randomized sublinear the diameter have not changed after random attack. However, after target removal of the most connected nodes, the diameter of Scale free graph dropped by 37,5% from 8 to 5, while the diameter of the Random network the decrease was only 16.5% from 12 to 10.

In contrast to the results of the same procedures from the work of Réka Albert, Hawoong Jeong and Albert-László Barabási (see Fig. 11), our models showed decrease of the diameter in case of the target attack, which happened because our networks were split into many components – into 8 and into 27 in random and scale-free networks, respectively (2000). That could be explained with the size and density of the networks from their paper that is much larger than ours. None the less, the number of resulting components is much larger in scale–free networks, which also provides the same idea as the Réka et al.: The scale–free networks are less resilient with respect to target removals.

Chart

Description automatically generated with medium confidence

**Figure 11.** Diameter after random and target attacks (Réka et al.)

**Sources:**

Réka Albert, Hawoong Jeong & Albert-László Barabási., 2000. Error and attack tolerance of complex networks. Nature, 406, pages378–382

Bearman, P. S., Moody, J., & Stovel, K. (2004). Chains of affection: The structure of adolescent romantic and sexual networks. American journal of sociology, 110(1), 44-91.

# Exercise three: Simulation of simple and complex contagion

(Question 10, 2 points)

For the Highschool network, identify five edges which after deletion, there will be significant gain of the average path lengths of the network. In other words, if such five edges did not exist, the average path length of the network would increase significant. Provide your answer in the format of A-B, in which A and B are the node ID. Are they weak ties or strong ties?

*Answer:*

By calculating the top 5 edges using betweenness average path length and deleting a edge the following edges would increase the average path length the most, resulting in the 3.956645 average path length which increased from 3.69:

* S4 --S37
* S4 --S77
* S17--S70
* S24--S72
* S37--S90

*S4-S37*: This edge has a highest betweenness centrality, meaning it acts as important bridge or connector between different parts of the network. If this edge is removed, it disrupts the network's overall connectivity and increase the average path length. This edge connects two nodes that are part of different clusters or communities in the network. This edge is a weak tie.

*S4 --S77:* This edge connects two communities that are separate from each other except for one node S77. This is the other edge S4 --S77 that would increase the average path length significantly. This is the bridge between a small community and 2 larger ones. This is a weak tie.

*S17--S70:* This is also a weak tie.

*S24--S72* : This a strong tie, but could be considered as a wide bridge between two communities

*S37--S90*: This is a week tie.

This demonstrates the strength of weak ties with which works best in case of simple contagion mechanism (Granovetter, 1973).

(Question 11, 6 points):

Simulate the spread of simple contagion in the Highschool network (

stopifnot(require(data.table))

stopifnot(require(Matrix))

calculate\_value <- function(node, each\_neighbors,Pprob){

return(each\_neighbors[[node]][ which(runif(length(each\_neighbors[[node]]), 0, 1)<=Pprob)])

#'runif' is a function to generate random number in R

}

#This function:

#1) searches the neighbors of contagious node;

#2) To those who are connected to a contagious node, generates a random number and compare to the

#probability of p, if random number<p, this node will be infected and return the value of 1

IC<-function(node\_seed,network,Pprob){

#prepare input for the 'calculate\_value' function#

adj\_matrix <- igraph::as\_adjacency\_matrix(network, type = 'both')

each\_neighbors <- which(adj\_matrix > 0, arr.ind = TRUE)

each\_neighbors <- split(each\_neighbors[, 2], each\_neighbors[, 1]) #get the neigbhour list of each node

nNode<-vcount(network)

node\_status <- rep.int(0, nNode) #start from a healthy population

day\_infected<-vector()#Total number of infected population

new\_infected <- list() # Record the ID of person getting infected at each time step

day<-1

node\_status[as.numeric(node\_seed)] <- 1 # infected(value=1) health(value=0)

day\_infected[day] <- sum(node\_status )

new\_infected[[day]]<-node\_seed #The ID of the person infected in Day 1 (Patient Zero)

#simulate the spread of virus within 4 weeks##

for (day in c(2:28)){

ContagiousID<-which(node\_status == 1)

infectedID<-unlist(lapply(ContagiousID,calculate\_value,each\_neighbors,Pprob))

newinfectedID<- setdiff(infectedID, which(node\_status == 1))

#Update the node status and other variables

node\_status[newinfectedID] <- 1

day\_infected[day] <- length(newinfectedID)

new\_infected[[day]]<-newinfectedID

day=day+1

}

return(day\_infected) #return the number of newly infected people by day

#return(list(day\_infected,new\_infected)) #if you want to see the ID of infected ppl in each day,use this command instead

tel = 0

for (i in 0:99){

tel = tel + IC("5", Highschool, 0.15)

}

as.integer(tel/100)

number of newly infected people by day: 1 0 1 1 2 3 4 5 5 6 7 8 9 9 9 8 7 6 5 4 3 2 1 1 1 0 0 0

(Question 12, 6 points):

Now you are going to test the “strength of weak ties” in the simple contagion:

1. Delete the 5 edges that you have identified in Q11 from the Highschool network and form a new network (Highschool 2);

The following edges are deleted edge: S4 - S37, S24 - S49, S28 - S97, S36 - S88 and S44 - S49.

1. Delete 5 strong ties from the Highschool network and form a new network (Highschool 3);

*Answer:* To determine the strongest ties an algorithm is used called k-core decomposition, which identifies the most densely connected subgraphs in the network. A k-core of a graph is a maximal connected subgraph in which every vertex is connected to at least k vertices in the subgraph.

This algorithm takes a range of values for k and returns the vertex set that constitutes the k-core with the highest possible value of k within the range. To obtain the k-core of a graph, the algorithm first deletes the vertices whose outdegree is less than k. It then updates the outdegree of the neighbors of the deleted vertices, and if that causes a vertex’s outdegree to fall below k, it will also delete that vertex.

The algorithm repeats this operation until every vertex left in the subgraph has an outdegree of at least k. (k-Core Decomposition, 2023). When running this algorithm on the high school network it returned nodes of only one subgraph so some strong ties will be manually examined and removed.

*Answer:* Removed nodes with k-core decomposition S20-S112, S112-S1, S110-S52 (see Figure 13-15)

|  |  |  |
| --- | --- | --- |
| Afbeelding met grafiek  Automatisch gegenereerde beschrijving | **Afbeelding met grafiek  Automatisch gegenereerde beschrijving** |  |
| **Figure 13** S20 | **Figure 14** S112 |  |

Afbeelding met grafiek

Automatisch gegenereerde beschrijving

**Figure 15** S102

Manually removed nodes: S107-S28, S64-S58 (see Figure 16-17)

|  |  |
| --- | --- |
| Afbeelding met grafiek  Automatisch gegenereerde beschrijving | Afbeelding met grafiek  Automatisch gegenereerde beschrijving |
| **Figure 16** S107 | **Figure 17** S64 |

1. Apply the IC models you developed in Q12 on the original Highschool network, Highschool2 and Highschool3. Record the number of newly infected people by day.

*Answer:* (1 0 1 1 2 3 4 4 5 6 8 8 8 9 9 8 8 6 5 4 3 2 1 1 0 0 0 0)

1. Generate a plot (with x-axis as Day, y-axis as the number of newly infected people by day) to compare the results from Step 3.

Afbeelding met grafiek

Automatisch gegenereerde beschrijving

**Figure 18** IC model

1. Recall the “strength of weak ties” from the lecture, do the results in Step 3&4 support such a claim and why?

*Answer*: The strength of weak ties is a concept that suggests that relationships with weak ties, can be more valuable in terms of providing opportunities and information than relationships with strong ties like close friends (Granovetter, 1973). That is also a reason why in case of simple contagion the spread preforms better on the Small - World than on Clustered models (Centola, pp. 28-29 2018).

This is also displayed in the Figure 18 by looking at the difference between the graphs highschool and highschool2. The weak ties act as bridges between different subgraphs, increasing the likelihood more people get infected every day.

When comparing the network highschool2 and highschool3 there are not many differences after removing strong ties. Strong ties won’t increase the amount of new people to be infected that much since losing one connection of many does not prevent the person to be infected. An example is a social group of 5 friends, if one person cuts of his relationship with one of the friends he can still infect that person through the other 3 friends.

Question 13 (8 points):

In the above exercises, the “strength of weak ties” are tested in a simplified IC model with a specific probability p. Do you think your observation in Q13 holds regardless of the contagiousness of the virus? To find out,

* 1. Play around the probability p in the IC model. Change the value of p to high and low ends, run the IC model again on Highschool, Highschool 2 and Highschool 3, and see if you will observe different things (2 points).

*Answer:* The same observation made in Q12 is seen in both graphs, however the graphs have a higher peak if the disease is less infectious and vice versa for a more infectious disease. Also, the peak amount of people being infected happens earlier if the disease is more infectious and later if the disease is not that infectious.

Afbeelding met grafiek

Automatisch gegenereerde beschrijving

**Figure 19** IC model with P of 0.9

Afbeelding met grafiek

Automatisch gegenereerde beschrijving

**Figure 20** IC model with P value of 0.1

* 1. The above IC model is a simplified version of the SIR model. In the SIR model, node have three status: S, I, or R, (Susceptible, Infectious, or Recovered). Modify the IC model to a SIR model with the following characteristics:
     + - Each node in the network has three statuses: Susceptible, Infectious, or Recovered.
       - At Day 0, all the nodes in the network are Susceptible;
       - At Day 1, an infectious node (N0, node ID= S5) is introduced to the network;
       - At the following days, all the nodes connecting to the infectious node will have a chance of 0.15 (p=0.15) being infected.
       - Every infected node will remain infectious for 3 days, i.e., only the infected nodes activated from the past 3 days can transmit the virus to their neighbours. After that, their status becomes Recovered, which cannot be either Infectious or Susceptible again.
       - Model the contagion process for 4 weeks.

(Question 14, 5 points):

An arbitrary assumption about the thresholds of each node in the Highschool network has been made, which can be found in the “Highschool\_network\_att.csv”. Build a threshold model according to the above model description and the predefined thresholds of each node, answer the following questions

1. By seeding 5 nodes (ID=59,63,91,92,99), how many people in the network can be activated? In total 52 people can be activated.

*Answer:*

Code:

threshold\_model <- function(network, node\_seed, threshold, n\_day){

nNode <- vcount(network)

node\_status <- rep.int(0, nNode) # start from a healthy population

adj\_matrix <- igraph::as\_adjacency\_matrix(network, type = 'both')

each\_neighbors <- which(adj\_matrix > 0, arr.ind = TRUE)

each\_neighbors <- split(each\_neighbors[, 2], each\_neighbors[, 1]) # get the neighbour list of each node

day <- 1

for (seed in node\_seed){

node\_status[as.numeric(node\_seed)] <- 1 # adopt(value=1) dont adopt(value=0)

}

for (day in 2:n\_day) {

for (node in 1:nNode) {

if (node\_status[node] == 0) {

neighbours <- each\_neighbors[[node]]

n\_neighbors <- length(neighbours)

n\_adopters <- sum(node\_status[neighbours] == 1)

if(n\_adopters > n\_neighbors \* threshold[node]){

node\_status[node] <- 1

}

}

}

}

return(sum(node\_status==1))

}

threshold\_model(Highschool, list('59', '63', '91', '92', '99'), highschool\_att$Threshold, 10000)

52

1. Use the “width of a bridge” from the lecture to explain why the contagion fails to reach the following two communities:

* the one consisted of Node 55, 107, 93, 109, 80, 28;
* the one consisted of Node 110, 39, 10, 1, 50, 106.

In a social network, a bridge is a tie between two nodes (or individuals) that belong to different clusters or communities. The width of a bridge between two communities is defined as the number of overlapping ties between them (Centola & Macy, 2007). So, a bridge with a low width has fewer ties connecting them. The contagion fails for these two communities because the majority of the neighbors needs to be adopting the campaign. This is not possible if the connections to these communities are connect by small bridges with weak ties.

Question 15 (6 points) :

Apply the empirical distribution of the threshold of students in this class to the Highschool network, and answer the following questions:

1. How are you going to do it? Explain your method into steps.

We used the distribution in the csv file, but if I would do it myself I would create a distribution that follows a sigmoid function and put more values on the lower and higher end of the sigmoid function. It would be similar to figure 14, but with more of a curve.

Afbeelding met grafiek

Automatisch gegenereerde beschrijving

**Figure 21** Empirical distribution

Afbeelding met grafiek

Automatisch gegenereerde beschrijving

**Figure 22** Distrubtion of the original threshold

After achieving this distribution, it needs to be assigned to the different students. I did this by randomizing the order of values in the distribution and assigned that list to the thresholds of students.

1. What are the limitations of your method? What procedures are you going to take to address such limitations?

*Answer:* This distribution is still based on a sample and not the original values. Another limitation is how the new thresholds get assigned to the students. I randomize the list of thresholds and assign it to each student, but since there are a lot of ones in the distribution it can occur that the ties with the starting students have a lot of thresholds of 1 which makes it harder to spread the “once a beef” campaign. This can be resolved by running the model multiple times with different orders of the distribution assigned to every student, but still won’t resolve the issue fully. Also, the same can be said about there being a lot of zeroes in the new thresholds. A lot of students can join the campaign if the correct students have a low threshold which will also not be representative of reality.

Next limitation is that a lot of thresholds are the same which is not representative to reality. I would resolve this by creating a distribution that follows a sigmoid function and put more values on the lower and higher end of the sigmoid function. This would make it so the influence of the campaign will process a bit more smoothly.

Another limitation is that a threshold of 1 will almost never be reached which is unrealistic. I would suggest reducing the amount of people of having a threshold of 1 and put it between 0.9 and 1. This would make the spread of the campaign also more realistic since a person is the average of the five people, he/she spends the most time with.

1. After you apply the empirical threshold distribution to the Highschool network, by using Node ID=59,63,91,92,99 as seeds, how many people in the network can be activated?

*Answer:* This depends fully on how the new distribution is assigned, but if I would run the model 1000 times with different assigned thresholds it would be an average of: 13

Question 16 (3 points): Search the application cases of threshold model from literatures or other online source, chose one case and explain how they can get the threshold “right” for their model. (please provide the details of the literature or other online source that you are citing.)

*Answer:* The threshold model has found a wide range of applications, for example, in:

* the simulations of computer viruses spread (Kephart, 1992)
* the simulations of obesity spread
* the simulations of rioting behaviour
* the simulation of spread of rumours
* modeling the diffusion of innovation (Rogers, 1975)
* modeling of migration decision-making and many other examples can be found in (Granovetter, 1978)

In case of rioters, Granovetter proposed to use equilibrium solution to find the optimal threshold for a model, buy setting r(t+1) = r(t), where r(t) is a rioting number of people at time t, which eventually will bring to the solution (see Figure 21),

However, this is modelling solution can be far from reality and one of the recent research papers proposed Linear Threshold modeling based on Casual Inference principles of J. Pearl and Structural Causal Modeling (see Figure 22), based on graphs. Such approach can conceptualize the riots as a more heterogeneous process, where for each node the is an individual configuration of predictors X on a potential outcome Y (in our case riot participation). Such model includes thresholds per node, and is defined as I which is the random variable of activation influence on the outcomes of v’s neighbors (See more in Tran & Zheleva, 2022).

|  |  |
| --- | --- |
| Diagram, engineering drawing  Description automatically generated | **Diagram, engineering drawing  Description automatically generated** |
| **Figure 23** Equilibrium solution (Granovetter, 1978) | **Figure 24** SCM solution |

**Sources:**

Centola, D. (2018). How behavior spreads: The science of complex contagions (Vol. 3). Princeton, NJ: Princeton University Press.

Granovetter, M. S. (1973). The strength of weak ties. American journal of sociology, 78(6), 1360-1380.

Centola, D., & Macy, M. (2007). Complex contagions and the weakness of long ties. American journal of Sociology, 113(3), 702-734.

Rogers, E. M. (1979). Network analysis of the diffusion of innovations. In Perspectives on social network research (pp. 137-164). Academic Press.

Kephart, J. O., & White, S. R. (1992). Directed-graph epidemiological models of computer viruses. In Computation: the micro and the macro view (pp. 71-102).

Tran, C., & Zheleva, E. (2022, June). Heterogeneous Peer Effects in the Linear Threshold Model. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 36, No. 4, pp. 4175-4183).

Granovetter, M (1978). Threshold Models of Collective Behaviour. American Journal of Sociolology 83, 1420–1443.

# Exercise four: Influence maximization

In the lecture, we discussed a few heuristics for the influence maximization problem in social network. Apply degree heuristics and betweenness heuristics to the IC model you have developed in Question 11 (! Please change the initially infected node to S107!). Answer the following questions (Question 17, 5 points):

1. You can immunize 3 nodes in the network, which after immunization, will never spread the virus to other connected nodes. According to degree heuristics and betweenness heuristics, which 3 nodes should be immunized in order to contain the virus?

Answer:

* according to degree heuristic: S54, S20, S110
* according to betweenness heuristic: S37, S4, S96

1. Immunize the 3 nodes suggested by degree heuristics and betweenness heuristics, respectively, which heuristic provides the better outcome regarding a) the final activated number of people and b) flattening the daily infection curve (please provide figure in your answer)?

*Answer:*

**Table 1** Average final number of activated nodes in High School data

|  |  |  |  |
| --- | --- | --- | --- |
| **Probability of contagion** | **No immunity** | **Degree Heuristic** | **Betweenness Heuristic** |
| Final number of activated nodes | | |
| **0.5** | 122 | 119 | 119 |
| **0.15** | 122 | 119 | 119 |
| **0.1** | 120 | 113 | 108 |
| **0.06** | 91 | 55 | 50 |
| **0.01** | 9 | 7 | 4 |

Chart, line chart

Description automatically generated

**Figure 24** Infected per day in the High School data

Answer:

So, from both numbers and the picture we see that betweenness centrality heuristic prevents the spread at least several nodes better in terms of final number of activations. However, IC model has a random character meaning that our results for final activation numbers are not representative, which made us to perform the IC model 100 times for the daily infection curve. From the results it is evident that if the probability of contagion is high enough, the whole network will be activated, but in a case of 0.1 and smallerheuristics may prevent the contagion of the whole network.

While the betweenness heuristic slows down the virus spread better, both heuristics flatten the curve. The performance of betweenness heuristic may be explained with its trait more global characteristic: while the degree considers the properties of a given node separately (i.e. the number of its connections), the betweenness takes into account the shortest paths from the whole graph.

1. Do you think the observation in 2) (i.e., degree heuristic preforms better than betweenness heuristics, or the opposite) is sensitive to a) the network structure and b) parameter in the IC model? And Why?

With regard to the contagion mechanism, as already mentioned in the previous question, two heuristics do not vary in terms of final activation number, however they differ in terms of the time, needed to cover every possible node (i.e. left skewness of the curve).

With regard to the dependency of the heuristic effectiveness on the network structure, from Table 2 and 3 it is evident that betweenness heuristic constrains the spread more effectively in Barabasi models but have little difference in Small model and in Highschool model (See Figure 25). The reason behind this we have partially covered in the previous question, claiming that betweenness considers more global traits of the network than degree. None the less, the more rigorous mathematical research in this topic of F. Morone, H. Makse claim that betweenness heuristic does not outperform other metrics (2015).

For small world model as well as for Barabasi model, we increased the number of days to figure out at what probability the whole network will be covered, and the bigger time period needed as the networks taken from the previous exercises are bigger than Highschool data.

|  |  |
| --- | --- |
| Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated |

**Figure 25** Infected per day in the Small world and Scale-free models

**Table 2** Average final number of activated nodes in Barabasi model with superliner probability dependency ***(power 1.5, size 300, seed node randomly selected every iteration)***

|  |  |  |  |
| --- | --- | --- | --- |
| **Probability of contagion** | **No immunity** | **Degree Heuristic** | **Betweenness Heuristic** |
| Final number of activated nodes | | |
| **0.5** | 299 | 288 | 159 |
| **0.15** | 276 | 266 | 157 |
| **0.1** | 233 | 227 | 135 |
| **0.06** | 49 | 50 | 36 |
| **0.01** | 3 | 3 | 1 |

**Table 3** Average final number of activated nodes in Small world model ***(size 300, seed node randomly selected every iteration)***

|  |  |  |  |
| --- | --- | --- | --- |
| **Probability of contagion** | **No immunity** | **Degree Heuristic** | **Betweenness Heuristic** |
| Final number of activated nodes | | |
| **0.5** | 300 | 297 | 297 |
| **0.15** | 300 | 297 | 297 |
| **0.1** | 294 | 291 | 292 |
| **0.06** | 213 | 217 | 206 |
| **0.01** | 3 | 3 | 1 |

(Important note: In Question 12, the initially infected node is S5. Please change it to S107 to answer Question 18. In other words, at Day 1, an infected node (N0, node ID= S107) is introduced to the network.)

In addition to heuristics, we also introduced the greedy algorithm in the lecture. Develop a greedy algorithm to the IC model you have developed in Question 11 (! Please change the initially infected node to S107!). Answer the following questions (Question 18, 5 points):

1. You can immunize 3 nodes in the network, which after immunization, will never spread the virus to other connected nodes. According to greedy algorithm, which 3 nodes should be immunized in order to contain the virus?

*Answer*: It is possible to solve this task with greedy in two ways: we may call them Influence maximization method and Influence minimization method. First implies finding the most influential node, as it was proposed by Kempe et. al who suggested to use greedy algorithm for IMP (Kempe et. al 2003). The second one implies that we look for nodes which immunized, reduce the spread to the minimum. For this we used some pieces of method from (Kingi, 2008), for example the usage of MonteCarlo simulations for making our solution more determined (See the code).

We run greedy algorithm multiple times and also switched from one criterion to choose the best nodes (the spread at the end of the time period) to the sum of number of infected per timestamp. That gave us the nodes which gained spread more rapidly. As the algorithm output is dependent on the using MC simulations and the number of MC simulations, we provided the best solution which works stable for both of us and gives the best result.

With greedy for Influence maximization problem: S2, S93, S20

With greedy for Influence minimization problem: S96, S37, S70

1)    Compared to the result from greedy algorithm to those from degree heuristic and betweenness heuristic. Regarding a) the final activated number of people and b) flattening the daily infection curve (please provide figure in your answer), does greedy algorithm provide the best result? And explain the reason.

*Answer*:

However, it is proven that greedy algorithm only approximates the optimum solution [], in our case sometimes it has not reached it, and the set of immunized nodes chosen with betweenness heuristic performed better. With greedy algorithm we add nodes one by one, achieving the Influence maximization with linear combinations of nodes ordered by their degree of influence. But the desired combination of node may produce its effect due to a relational nature but not due to a cumulative nature. So independently from each other, or better say dependent not fully on each other but only on the preceding node, the set of nodes from greedy may be less influential than set retrieved with betweenness heuristic (see Figure 26).

|  |  |
| --- | --- |
| Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated |
| **Figure 26** With S37, S21, S9 Greedy Minmization | **Figure 27** With nodes S97, S14, S102 from greedy Minmization, the best solution |

**Table 4 Average final number of activated nodes   
for Greedy on Highschool data** S97, S14, S102

|  |  |  |
| --- | --- | --- |
| **Probability of contagion** | **Greedy**  **Minimization** | **Betweenness Heuristic:** |
| Final activated number | |
| 0.5 | 119 | 119 |
| 0.15 | 117 | 119 |
| 0.1 | 99 | 108 |
| 0.1 | 56 | 50 |
| 0.01 | 4 | 4 |

(Important note: In Question 12, the initially infected node is S5. Please change it to S107 to answer Question 18. In other words, at Day 1, an infected node (N0, node ID= S107) is introduced to the network. Please submit the codes of this question along with your answer.)

***Answer:*  Note that in the code the indices of nodes are i – *1, functions\_EX4 is a script containing custom functions, they are also provided***

# Import packages

%matplotlib inline

import matplotlib.pyplot as plt

from random import uniform, seed

import numpy as np

import pandas as pd

import csv

import time

from igraph import \*

from functions\_EX4 import \*

def compute\_mean\_by\_index(list\_of\_lists):

# Get the number of inner lists

num\_lists = len(list\_of\_lists)

# Get the length of each inner list

list\_length = len(list\_of\_lists[0])

# Initialize a list to store the means

means = []

# Iterate through the indices

for i in range(list\_length):

# Initialize a sum variable for the current index

sum\_for\_index = 0

# Iterate through the inner lists

for j in range(num\_lists):

# Add the value at the current index to the sum

sum\_for\_index += list\_of\_lists[j][i]

# Compute the mean for the current index

mean\_for\_index = sum\_for\_index / num\_lists

# Append the mean to the result list

means.append(mean\_for\_index)

return means

def IC(g,S,p=0.5,mc=1000, timestamps = 28, Monte\_Carlo = True):

"""

Input:

g - graph object,

S - set of seed nodes(dtype list)

p - propagation probability

mc - the number of Monte-Carlo simulations

timestamps - the number of timestamps

Monte\_Carlo - Boolean, determines whether to make to random seed or not

Output:

- average number of nodes activated in each Monte-Carlo simulation

- average number of nodes influenced by the seed nodes in each timestamp

"""

# Loop over the Monte-Carlo Simulations

spread, sum\_spread = [], []

for i in range(mc):

# Simulate propagation process

Active = S[:]

sum = 0

for \_ in range(timestamps):

new\_ones = []

# For each active node, find its neighbors that become activated

for node in Active:

if Monte\_Carlo == True:

np.random.seed(i)

# Determine neighbors that become infected

success = np.random.uniform(0,1,len(g.neighbors(node,mode="out"))) < p

new\_ones += list(np.extract(success, g.neighbors(node,mode="out")))

new\_active = list(set(new\_ones) - set(Active))

sum += len(new\_active)

# in case the network is fully activated

if new\_active == []:

break

# Add newly activated nodes to the set of activated nodes

Active = list(set(Active).union(set(new\_active)))

spread.append(len(Active))

sum\_spread.append(sum)

return(np.mean(spread), np.mean(sum\_spread))

def IC\_immunized(g, S, immunized, p=0.15, mc=1000, timestamps = 28, full = False, Monte\_Carlo = True):

"""

Input:

g - graph object,

S - set of seed nodes(dtype list)

immunized - set of immunized nodes(dtype list)

p - propagation probability

mc - the number of Monte-Carlo simulations

timestamps - the number of timestamps

Monte\_Carlo - Boolean, determines whether to make to random seed or not

Output:

- average number of nodes activated in each Monte-Carlo simulation

- average number of nodes influenced by the seed nodes in each timestamp

"""

# Loop over the Monte-Carlo Simulations

spread, sum\_spread, infected\_a\_day = [], [], []

for i in range(mc):

S = list(set(S).difference(set(immunized)))

# Simulate propagation process

Active = S[:]

sum = 0

infected\_per\_day = []

for \_ in range(timestamps):

new\_ones = []

# For each active node, find its neighbors that become activated

for node in Active:

if Monte\_Carlo == True:

np.random.seed(i)

# Determine neighbors that become infected

success = np.random.uniform(0,1, len(g.neighbors(node,mode="out"))) <= p

new\_ones += list(np.extract(success, g.neighbors(node,mode="out")))

new\_ones = list(set(new\_ones).difference(set(immunized)))

new\_active = list(set(new\_ones).difference(set(Active)))

sum += len(Active)

if new\_active == [] and full == True:

break

infected\_per\_day.append(len(new\_active))

# Add newly activated nodes to the set of activated nodes

Active = list(set(Active).union(set(new\_active)))

spread.append(len(Active))

sum\_spread.append(sum)

infected\_a\_day.append(infected\_per\_day)

return(np.mean(spread), np.mean(sum\_spread), compute\_mean\_by\_index(infected\_a\_day))

def greedy(g, k, p=0.15, mc=1000, timestamps = 28):

"""

Input:

g - graph object

k - number of seed nodes

p - propagation probability

mc - the number of Monte-Carlo simulations

timestamps - the number of timestamps

Monte\_Carlo - Boolean, determines whether to make to random seed or not

Output: optimal seed set, resulting spread, time for each iteration

"""

S, spread, timelapse, start\_time = [], [], [], time.time()

# setting list of nodes and making the node 107 the first node in odrer to compare with other methods

l = [i for i in range(g.vcount())]

l.remove(106)

l.insert(0, 106)

# Find k nodes with largest marginal gain

for \_ in range(k):

# Loop over nodes that are not yet in seed set to find biggest marginal gain

best\_spread = 0

#for j in set(range(g.vcount())) - set(S):

for j in set(l) - set(S):

# Get the spread

s = IC(g, S + [j], p, mc, timestamps, Monte\_Carlo = True)

# Update the winning node and spread so far

if s[1] > best\_spread:

best\_spread, node = s[1], j

# Add the selected node to the seed set

S.append(node)

# Add estimated spread and elapsed time

spread.append([best\_spread, s[1]])

timelapse.append(time.time() - start\_time)

return(S,spread, timelapse)

def greedy\_immunized(g,k,p=0.1,mc=1000):

"""

g - graph object

k - number of seed nodes

p - propagation probability

mc - the number of Monte-Carlo simulations

timestamps - the number of timestamps

Monte\_Carlo - Boolean, determines whether to make to random seed or not

Output: optimal seed set, resulting spread, time for each iteration

"""

immunaized\_list, spread, timelapse, start\_time = [],[], [], time.time()

# Find k nodes with largest marginal gain

for \_ in range(k):

# Loop over nodes that are not yet in seed set to find biggest marginal gain

best\_spread = IC\_immunized(g, [106],[], p, mc, timestamps = 28, full = True, Monte\_Carlo = True)[1]

for j in set(range(g.vcount())) - set(immunaized\_list):

# Get the spread

s = IC\_immunized(g, [106], immunaized\_list + [j], p, mc, timestamps=28, full = True, Monte\_Carlo = True)

# Update the winning node and spread so far

if s[1] < best\_spread and s[1] !=0 :

best\_spread, node = s[1], j

# Add the selected node to the seed set

if 'node' in locals():

immunaized\_list.append(node)

# Add estimated spread and elapsed time

spread.append(best\_spread)

timelapse.append(time.time() - start\_time)

return(immunaized\_list, spread, timelapse)

## START of TASK

with open('Highschool\_network\_edge.csv') as csvfile:

reader = csv.reader(csvfile)

edges = [(int(row[0][1:]), int(row[1][1:])) for row in reader]

NodeID = []

Gender = []

Hall = []

Threshold = []

with open('Highschool\_network\_att.csv') as csvfile:

reader = csv.reader(csvfile)

next(reader)

for row in reader:

NodeID.append(int(row[0][1:]))

Gender.append(row[1])

Hall.append(row[2])

Threshold.append(row[3])

g = Graph(edges, directed=False)

g.delete\_vertices(0)

# Number of components

g.vs["NodeID"] = NodeID

print("Number of components: ", g.vcount())

greedy\_immunized(g, 3, 0.15, mc=600)

greedy(g, 3, 0.15, mc=200) # works to slow

# creating the list of infected nodes per day

No\_immunity = IC\_immunized(g, [106], [], p = 0.15, mc = 100, timestamps = 28, Monte\_Carlo=False)

Degree = IC\_immunized(g, [106], [53, 19, 109], p = 0.15, mc = 100, timestamps = 28, Monte\_Carlo=False)

Betweenness = IC\_immunized(g, [106], [36, 3, 95], p = 0.15, mc = 100, timestamps = 28, Monte\_Carlo=False)

Greedy = IC\_immunized(g, [106], [53, 91, 72], p = 0.15, mc = 100, timestamps = 28, Monte\_Carlo=False)

Greedy2 = IC\_immunized(g, [106], [95, 36, 69], p = 0.15, mc = 100, timestamps = 28 , Monte\_Carlo=False)

# Creating a list of days as x-axis

days = list(range(1, len(Greedy[2]) + 1))

# Plotting the three lists

plt.plot(days, No\_immunity[2], label='No\_immunity')

plt.plot(days, Degree[2], label='Degree heuristic')

plt.plot(days, Betweenness[2], label='Betweenness heuristic')

plt.plot(days, Greedy[2], label='Greedy with Influence Maximization')

plt.plot(days, Greedy2[2], label='Greedy with Influence Minimization')

# Adding labels and title

plt.xlabel('Days')

plt.ylabel('Number of Infected')

plt.title('Daily Infected Cases')

# Adding legend

plt.legend()

# Displaying the plot

plt.show()

Question 19 (8 points):

Following below steps to answer

1. Use the threshold model for the “once-a-beef” campaign you build in Question 14, reset the seed nodes to a null set;

*Answer:*

threshold\_model(g, [], Threshold\_list, 28, mc=100, Monte\_Carlo = True)

1. According to degree heuristics, which nodes should be included in the seed set in order to maximize the spread of the campaign? The size of seed set is 7, i.e., you can choose 7 nodes to activate to kick off the contagion process.

*Answer:* 54, 20, 110, 64, 96, 28, 37

degree\_centrality = g.degree()

sorted\_nodes = sorted(range(len(degree\_centrality)), key=degree\_centrality.\_\_getitem\_\_, reverse=True)

top\_7\_nodes = sorted\_nodes[:7]

print("Top 7 Nodes based on Degree Centrality: ", top\_7\_nodes)

1. According to betweenness heuristics, which nodes should be included in the seed set in order to maximize the spread of the campaign? The size of seed set is 7, i.e., you can choose 7 nodes to activate to kick off the contagion process.

*Answer:* 37, 4, 96, 24, 70, 77, 17

betweenness\_centrality = g.betweenness()

sorted\_nodes = sorted(range(len(degree\_centrality)), key=betweenness\_centrality.\_\_getitem\_\_, reverse=True)

top\_7\_nodes = sorted\_nodes[:7]

print("Top 7 Nodes based on Betweenness Centrality: ", top\_7\_nodes)

1. According to greedy algorithm, which nodes should be included in the seed set in order to maximize the spread of the campaign? The size of seed set is 7, i.e., you can choose 7 nodes to activate to kick off the contagion process.

*Answer:* 42, 108, 61, 22, 31, 46, 5

greedy\_Th(g, 7, Threshold\_list, mc=10)

1. Compared the results from degree heuristics, betweenness heuristics and greedy algorithm, which method provides the best outcome? Please show the results of three methods in a figure.

Chart, line chart

Description automatically generated

**Figure 28** the cumulative number of spreads

(Please submit the codes of this question along with your answer.)

Code:

# Original dictionary with with values from beef campaing

original\_dict = {0:15, 1:5, 2:5, 3:1, 4:2, 5:2, 6:1, 7:1, 8:1, 9:0, 10:4}

# Calculate total sum of original values

total\_sum = sum(original\_dict.values())

# Desired total sum

desired\_sum = 122

# Scale factor

scale\_factor = desired\_sum / total\_sum

# Create a new dictionary to store the scaled values

scaled\_dict = {}

# Scale up each value proportionally and round to nearest integer

for key, value in original\_dict.items():

scaled\_value = round(value \* scale\_factor)

scaled\_dict[key] = scaled\_value

# Check if the total sum of scaled values is less than the desired sum

# If so, add the difference to the highest key (">=10")

total\_scaled\_sum = sum(scaled\_dict.values())

if total\_scaled\_sum < desired\_sum:

diff = desired\_sum - total\_scaled\_sum

scaled\_dict[10] += diff

Threshold\_list = []

for key, value in scaled\_dict.items():

Threshold\_list.extend([key] \* value)

# Shuffle the list to randomize the distribution

np.random.shuffle(Threshold\_list)

def threshold\_model(network, node\_seed, threshold, n\_day, mc=100, Monte\_Carlo=False):

spread, sum\_spread, infected\_per\_day, infected\_per\_iteration = [], [], [], []

for i in range(mc):

if Monte\_Carlo == True:

np.random.seed(i)

nNode = network.vcount()

node\_status = np.zeros(nNode, dtype=int) # start from a healthy population

adj\_matrix = np.array(network.get\_adjacency().data)

each\_neighbors = {node: np.where(adj\_matrix[node, :] > 0)[0] for node in range(nNode)} # get the neighbor list of each node

infected\_a\_day,infected\_List = [] , []

for seed in node\_seed:

node\_status[seed] = 1 # adopt (value=1), don't adopt (value=0)

sum\_of\_ifected = 0

for day in range(1,29): #n\_day + 1):

n\_infected = 0

for node in range(nNode):

if node\_status[node] == 0:

neighbours = each\_neighbors[node]

n\_neighbors = len(neighbours)

n\_adopters = np.sum(node\_status[neighbours] == 1)

#if n\_adopters/n\_neighbors > threshold[node]:

if n\_adopters > threshold[node]:

node\_status[node] = 1

n\_infected += 1

infected\_at\_all = np.sum(node\_status == 1)

sum\_of\_ifected += np.sum(node\_status == 1)

infected\_a\_day.append(n\_infected)

infected\_List.append(infected\_at\_all)

# Collect all of the necessary information

spread\_per\_simulation = np.sum(node\_status == 1)

spread.append(spread\_per\_simulation)

sum\_spread.append(sum\_of\_ifected)

infected\_per\_day.append(infected\_a\_day)

infected\_per\_iteration.append(infected\_List)

return np.mean(spread), np.mean(sum\_spread), compute\_mean\_by\_index(infected\_per\_day), compute\_mean\_by\_index(infected\_per\_iteration)

def greedy\_Th(g, k, threshold, mc=1000, timestamps = 28):

"""

Input:

g - graph object

k - number of seed nodes

p - propagation probability

mc - the number of Monte-Carlo simulations

timestamps - the number of timestamps

Monte\_Carlo - Boolean, determines whether to make to random seed or not

Output: optimal seed set, resulting spread, time for each iteration

"""

# Threshold\_list = [0] \* 15 + [0.1] \* 5 + [0.2] \* 5 + [0.3] + [0.4] \* 2 + [0.5] \* 2 + [0.6, 0.7, 0.8] + [0.9] \* 4 + [1] \* 85

# # shuffle the list

# np.random.shuffle(Threshold\_list)

S, spread, timelapse, start\_time = [], [], [], time.time()

# setting list of nodes and making the node 107 the first node in odrer to compare with other methods

l = [i for i in range(g.vcount())]

l.remove(106)

l.insert(0, 106)

# Find k nodes with largest marginal gain

for \_ in range(k):

# Loop over nodes that are not yet in seed set to find biggest marginal gain

best\_spread = 0

#for j in set(range(g.vcount())) - set(S):

for j in set(l) - set(S):

# Get the spread

s = threshold\_model(g, S + [j], threshold, 28, mc, Monte\_Carlo = True)

# Update the winning node and spread so far

if s[1] > best\_spread:

best\_spread, node = s[1], j

# Add the selected node to the seed set

S.append(node)

# Add estimated spread and elapsed time

spread.append(best\_spread)

timelapse.append(time.time() - start\_time)

return(S,spread, timelapse)

Question 20 (7 points): You have compared the performance of degree heuristics, betweenness heuristics and greedy algorithm. Can you propose an even more efficient algorithm (i.e., achieve even higher diffusion rate with even less percentage of nodes using as seeds)? Answer this question by the following steps:

1. Description of your algorithm and the reason why you think it will be more effective;

In this task we were about to propose to use a Cost Effective Lazy forward algorithm (CELF), which was developed and suggested by Leskovec et al in 2007. Their approach aimed to reduce the computational complexity of Greedy algorithm which occurs in larger networks even with Greedy. This algorithm is based on the submodularity property of the spread function, which is why it is faster than Greedy. For this task we use the instruments from article of Kingi, H. with little adjustments for threshold models (Kingi, 2008).

The submodularity works that way:

Let us define *f* (S)– as a spread function where S is a set of seeded nodes, and we want   
*f* (S) –> max with least possible |S|

If we compute *f(Si)* for every ith node and then order them by their marginal spread, then we can bound new iterations by setting Greedy that it cannot be any larger than its marginal spread in the previous iteration.

* 1. The first top ith node comes naturally from the stored sorted list of spread Si, and then removed from the list.
  2. In the next iteration we only compute the spread for the top node of remaining list.
  3. If, after resorting, that node stays at the top of the list, then we include it in a set S, as it must have the highest marginal gain of all nodes.
  4. Continue until |S| = k, where k is a desired number of seeds (Kingi, 2008).

This should be more effective (in the ideal case with the proper code) as it imposes some boundaries for greedy.

1. Test your algorithm in the threshold model for the “once-a-beef” campaign used in Question 19; Comments on its effectiveness

Unfortunately, this algorithm has not outperformed the greedy in computational time due to double sort and inefficient code of threshold model (see Figure \_). However, in comparison, to other methods it proposed better solution for seeding (see Figure \_), meaning that CELF has found more global optimum of *f(S*) then Greedy hypothetically due to not sufficient amount of MC simulations.

|  |  |
| --- | --- |
| Chart, line chart  Description automatically generated | Chart, line chart  Description automatically generated |
| **Figure 29** computational time Greedy vs CELF | **Figure** **30** Final activated nodes Others vs CELF |

1. Test your algorithm in a large real-world network (e.g., n≥1000) using threshold model. For the larger network, you can choose one from the [database](https://snap.stanford.edu/data/), and make a subgraph from it (e.g., choose only 1000 nodes). Describe the network you choose, your setting about threshold, and comment on the effectiveness of your algorithm on this network.

For this task we used the data from (S. Kumar et. al 2016) and (S. Kumar et. al 2018) which consist of trust relationship in the Bitcoin Alpha platform, meaning if the edge between node is present there is a trustful relationship established. N = 3783 and n of edges is 24000. We used a subgraph consisting out of 999 nodes.

For the threshold we used the previous distribution from the “once-a-beef” campaign as it has a lot of low thresholds and would not create any burdens in terms of network structure.

|  |  |
| --- | --- |
| Graphical user interface, application  Description automatically generated with medium confidence |  |
| **Figure** computational time Greedy vs CELF in Bitcoin data | **Figure** The Subgraph of the network |

(Note: The proposed algorithm does not need to excel in all the settings, but should outperform at least some of the existing heuristics or greedy algorithm in some scenarios. And please submit the codes of this question along with your answer.)

def celf(g,k,threshold,mc):

"""

Input: graph object, number of seed nodes

Output: optimal seed set, resulting spread, time for each iteration

"""

# --------------------

# Find the first node with greedy algorithm

# --------------------

# Calculate the first iteration sorted list

start\_time = time.time()

marg\_gain = [threshold\_model(g,[node], threshold, mc)[0] for node in range(g.vcount())]

# Create the sorted list of nodes and their marginal gain

Q = sorted(zip(range(g.vcount()), marg\_gain), key=lambda x: x[1],reverse=True)

# Select the first node and remove from candidate list

S, spread, SPREAD = [Q[0][0]], Q[0][1], [Q[0][1]]

Q, timelapse = Q[1:], [time.time()-start\_time]

# LOOKUPS = [g.vcount()]

# --------------------

# Find the next k-1 nodes using the list-sorting procedure

# --------------------

for \_ in range(k-1):

check, node\_lookup = False, 0

while not check:

# Count the number of times the spread is computed

node\_lookup += 1

# Recalculate spread of top node

current = Q[0][0]

# Evaluate the spread function and store the marginal gain in the list

Q[0] = (current, threshold\_model(g, S + [current], threshold, mc)[0] - spread)

# print(Q)

# Re-sort the list

Q = sorted(Q, key = lambda x: x[1], reverse = True)

# Check if previous top node stayed on top after the sort

check = (Q[0][0] == current)

# Select the next node

spread += Q[0][1]

S.append(Q[0][0])

SPREAD.append(spread)

# LOOKUPS.append(node\_lookup)

timelapse.append(time.time() - start\_time)

# Remove the selected node from the list

Q = Q[1:]

return(S,SPREAD,timelapse) #,LOOKUPS)

**Sources:**

Kempe, D., Kleinberg, J., & Tardos, É. (2003, August). Maximizing the spread of influence through a social network. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 137-146).

Kingi, H. (2018, September 7). Influence Maximization in Python - Greedy vs CELF. Retrieved from <https://hautahi.com/im_greedycelf>

Morone, F., & Makse, H. A. (2015). Influence maximization in complex networks through optimal percolation. Nature, 524(7563), 65-68.

Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., & Glance, N. (2007, August). Cost-effective outbreak detection in networks. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 420-429).

S. Kumar, F. Spezzano, V.S. Subrahmanian, C. Faloutsos. [Edge Weight Prediction in Weighted Signed Networks](http://cs.stanford.edu/~srijan/pubs/wsn-icdm16.pdf). IEEE International Conference on Data Mining (ICDM), 2016.

S. Kumar, B. Hooi, D. Makhija, M. Kumar, V.S. Subrahmanian, C. Faloutsos. [REV2: Fraudulent User Prediction in Rating Platforms](http://cs.stanford.edu/~srijan/pubs/rev2-wsdm18.pdf). 11th ACM International Conference on Web Searchand Data Mining (WSDM), 2018.