Complex Networks - Homework 8

Group 1: Iwan Pasveer, Flavia Leotta, Noah Frinking, Dylan Gavron

19th November 2024

1 Introduction

This document is supported by markdown file "CN_HW8_Group1.ipynb", refer to it for the code. For more information about the libraries used, see section "Softwares and libraries". Each function is called with $random.seed(15)$ unless otherwise stated.

2 Homework 8.1

We implement the function:

`fail_attack(graph, initial_attack_list, time_steps)` (1)

that simulates an attack that aims to deactivate the nodes in a network. The attack starts from the nodes in the list *initial_attack_list* and then propagates for *time_steps* iterations by inactivating the nodes with the highest degree between all of the neighbors of the already inactivated nodes. For more information about the arguments and the results, please refer to the markdown file mentioned in the Introduction.

On the left of Figure 1 we show the plot of how the size of the Largest Connected Component (LCC) changes if the initial 5 nodes are chosen randomly (G1) or if they're the nodes with the highest degrees (G2), both in a BA network with $n = 1000$ and $m = 3$. As we expect, both networks fail rather quickly: around 10 iterations, the Largest Connected Component for both networks falls to 0, showing that the network was successfully fragmented.

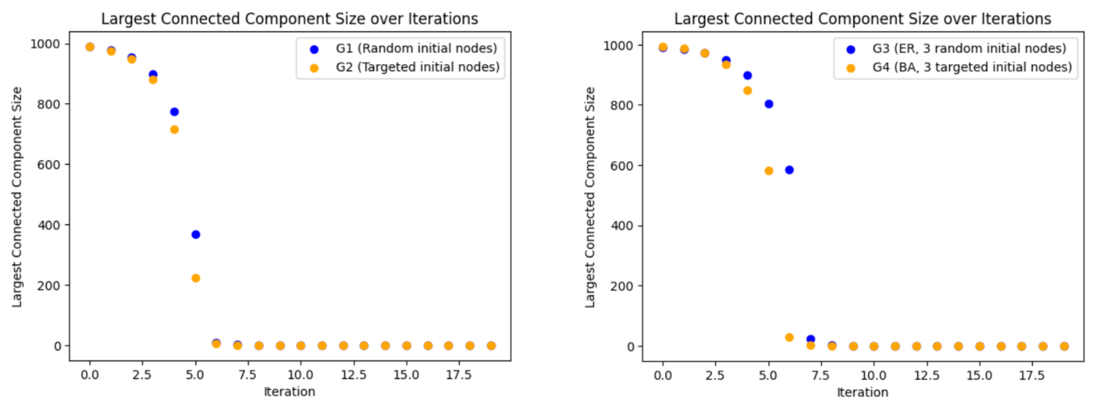


Figure 1: Attack on graphs by using function (1). On the left, an attack on two BA networks (n = 1000 and m = 3), on the right an attack on an ER and a BA network, both with 1000 nodes and 3000 edges.

The slope between the two is rather similar, with the only difference being that G2 starts failing slightly earlier: this is because, by starting from the highest degree nodes, the attack has what we can call as an "head start". After that, the attacks on both graphs show a similar behavior, given by the fact that at each iteration it spreads towards the highest degree nodes linked to the inactivated ones. Even if in G1 we start from randomly selected nodes, the attack doesn't continue randomly and that's the reason between this similarity.

On the right of the same image, instead, we show how the same model attacks two graphs with the same amount of nodes (1000) and edges (3000), but built in two different ways: G3 is an ER network, while G4 is a BA network. In this case we choose 3 random nodes to attack in the ER graph and the 3 nodes with the highest degrees in the BA model, but the difference between the two is even more evident than when comparing two BA networks. The difference not only lays in how the attack begins, but also in the structure of these two models: a BA network is a scale-free network in which there are few hubs (very well connected nodes) and a lot of nodes with a low/medium degree. This makes a BA network very resilient to random attacks but particularly vulnerable to targeted attacks.

It's hard to appreciate in the graph due to the scale of the y axis, but the Largest Connected Component Size for the ER graph doesn't get to 0 (while, instead, it always happens in the BA network). In this case, with `random.seed(10)`, the lowest the LCC size goes in the case of the ER network, is 1. This happened because, by construction, the ER networks allows for the possibility of nodes with degree equal to 0, and in this graph there were 3 nodes in this situation. These isolated nodes, if they're not randomly chosen as the first nodes to be attacked, will never fail, giving rise to little connected components that survive until the very end.

3 Homework 8.2

We implement the function:

$$\texttt{immunize\_k(graph, initial\_propagator, k)} \tag{2}$$

that, given a *graph*, immunizes k nodes, by setting their state as 'immune', thus making their infection impossible. These k nodes are the highest degree nodes chosen between all those having distance less than 3 from the nodes in the *initial_propagator* list, which are those that start the attack.

We then attack a BA network ($n = 1000$ and $m = 3$) with function:

$$\texttt{IndependentCascade(graph, propagator, p, iterations)} \tag{3}$$

that simulates an attack following an Independent Cascade model with an influence probability p. Each node in the *propagator* list, has one chance to influence each of its neighbors with probability p: at each iteration, the *propagator* list is updated to contain only the nodes that have been influenced on that time step. We simulate this attack twice: the first time we don't immunize any node, the second time instead we immunize 10 nodes with the same mechanics shown for function (2). In figure 2 we show how the total number of influenced nodes increase at each iteration of function (3).

We can see that the non-immunized graph gets quickly infected, reaching a plateau of around 800 infected nodes. The probability of infection was set to 0.4, which means that for each node, at each iteration, around 40% of its neighbors get influenced. It follows easily that this is a big problem in the case of hubs (that, we remind the reader, are characteristics of BA networks): if they get infected at a certain time step, we will have an important increase in the number of infected nodes, and in the next time step, we will also have an increased potential of infection. If, instead, we immunize nodes with the same logic as explained before, we block some "paths" that would allow the disease to spread quickly: not only that, but paired with the possibility of the influence to fail, can successfully eradicate the disease.

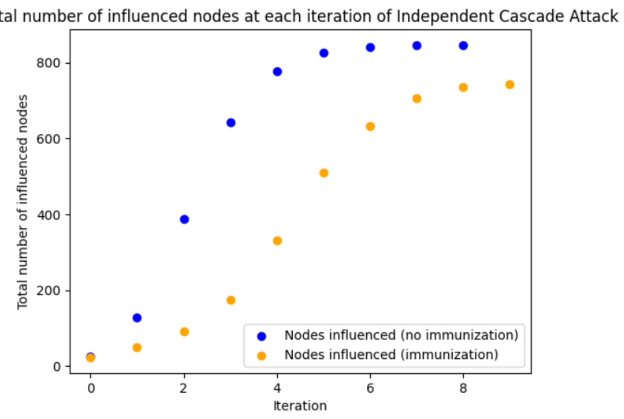


Figure 2: Total number of influenced nodes at each iteration of (3), on a BA network with no immunized nodes (blue) and with 10 immunized nodes (orange), chosen with the criteria shown for (2).

Of course, how much the immunization will succeed in this task is rooted in the choice of the 10 initial nodes: if these nodes have a low degree and/or are surrounded by hubs that get immunized, the disease would have an harder time spreading towards other areas of the graph. Similarly, if the initial nodes are centered in one area, it would be easier to isolate them by immunizing targeted nodes connected to several of the initial ones. Instead, if they're more spread, it will be harder to protect the network being attacked from different directions. In our example, even though the immunization helps in slowing the disease, it quickly raises up to a plateau of around 700 nodes, which suggests that the disease might have reached some hubs anyway.

4 Homework 8.3

4.1 8.3.1

We implement the function:

```
Competitive_Linear_Threshold(G, initial_infected_nodes_1, initial_infected_nodes_2,
threshold_1, threshold_2, time_steps)
```
(4)

that simulates an attack of two 'diseases' on the same graph, while following a Competitive Linear Threshold model.

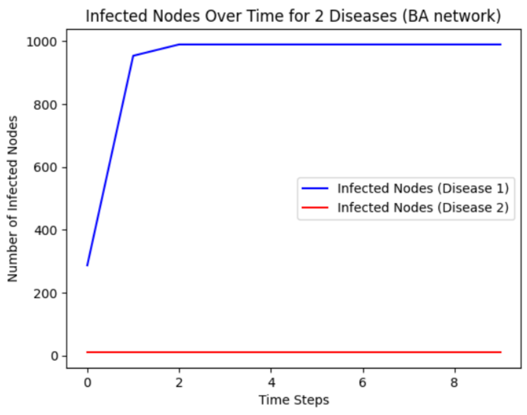


Figure 3: Infected nodes over time in a BA network. Disease 1 has an influence threshold of 0.2, while disease 2's threshold is 0.4

We first simulate this attack on a BA network with $n = 1000$ and $m = 3$, but starting with two different approaches. The starting infected nodes for both diseases are given by the two lists $initial\_infected\_nodes\_1$ and $initial\_infected\_nodes\_2$, and both contain 5 nodes. For the first disease, the nodes are chosen randomly but are connected to each other, while for the second disease we choose the 5 highest degree nodes excluding the ones already chosen for the first disease.

In Figure 3 we can see that, the first disease almost immediately saturates the graph, leaving little to no space for the second disease. This behavior is observed through different seeds (in the figure is reported the one for $random.seed(15)$). Even though the second disease starts from the highest degree nodes, it fails to strongly counter-attack the first disease due to its threshold being bigger ($threshold\_2 = 0.4$ and $threshold\_1 = 0.2$), thus requiring a bigger proportion of infected nodes to propagate the disease.

Not only that, but the second disease suffers from the choice of starting nodes for the first disease, but this behavior is better explained in Figure 4.

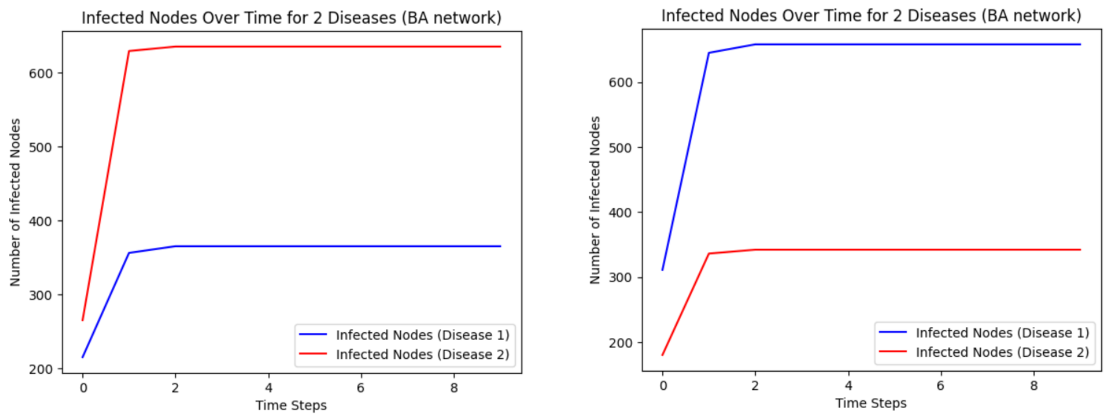


Figure 4: Infected nodes over time in a BA network. Both diseases have an influence threshold of 0.2, but the first graph is generated with $random.seed(15)$, while the second with $random.seed(17)$. The nodes chosen (and their degrees) are reported in tables 1 and 2.

In this image we set both thresholds to 0.2: this way, both diseases have the same potential for spreading, and their speed depends solely on the starting nodes. We noticed that the plot largely changed, depending on the seed used. This is due to the fact that, when choosing randomly the nodes for the beginning of the first disease, we potentially remove some important choices for the starting point of the second disease. In tables 1 and 2 we show degrees of the nodes chosen for the two diseases, according to different seeds. In both cases, the median degree of the graph is 4, while the mean degree is 5. This leads us to believe that the nodes chosen by disease 2 are usually the hubs. In $random.seed(17)$ we noticed that the first disease randomly started from nodes that are almost all of an higher degree than the ones chosen for the second degree: in this case we could almost say that the first disease has randomly ”targeted” the highest degree nodes that were supposed to be chosen by the second disease.

Table 1: $random.seed(15)$

Node	**Disease 1**	**Disease 2**
1	59	68
2	58	63
3	35	55
4	25	42
5	107	35

Table 2: $random.seed(17)$

Node	**Disease 1**	**Disease 2**
1	66	64
2	89	56
3	90	43
4	31	43
5	54	41

4.2 8.3.2

We simulate the same attack on an Stochastic Block model, built used the following command:

```
sizes = [400, 300, 300]
probs = [[0.25, 0.05, 0.02], [0.05, 0.35, 0.07], [0.02, 0.07, 0.40]]
G = nx.stochastic_block_model(sizes, probs, seed=0)
```
(5)

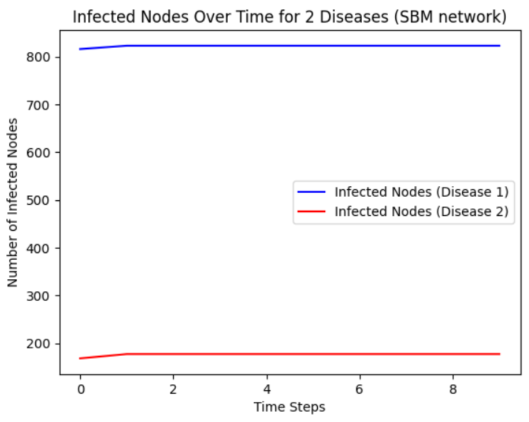
The SBM model generates graphs with nodes that are grouped into communities: for each node, there are different probabilities of forming edges with nodes of the other communities, and an higher chance of forming an edge with nodes of the same one. In our case, there are 3 communities of sizes 400, 300 and 300, with probability matrix shown in Table 3. The values on the diagonal represent the intra-community probability of forming an edge, that is always higher than the inter-community probabilities (the other values).

Table 3: Probability matrix for SBM model

Community	**0**	**1**	**2**
0	0.25	0.05	0.02
1	0.05	0.35	0.07
2	0.02	0.07	0.40

We use the same function (4), but in this case the list $initial\_infected\_nodes\_1$ contains 5 nodes that belong to the same community and form a connected sub-graph. As before, $initial\_infected\_nodes\_2$ contains the 5 highest degree nodes that were not already chosen to begin the spread of the first disease. The structure of an SBM model, being divided into communities, shows a different mechanism than the example from before.

In Figure 5 we show the result of the implementation of function (4) with both thresholds set on 0.02.



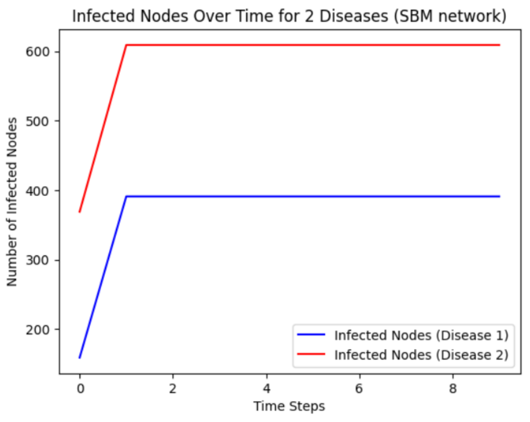


Figure 5: Infected nodes over time in a SBM network. On the left, the graph generated by $random.seed(15)$ and on the right, the one generated by $random.seed(17)$.

As we see, depending on the seeds, the results may change. What we've noticed, though, is that when the first disease would spread faster than the second one (like in the case of $random.seed(15)$, Figure 5, left), it would quickly saturate the graph, with a stark difference in the number of nodes infected by this disease (strongly preferred) and by the second one. Instead, when the second disease would take the lead (like in $random.seed(17)$) the difference between the two would be less prominent. These behaviors are consistent when several seeds are explored.

This means that, in both cases, the first disease is more effective in infecting nodes, and this might be because, by construction, it starts from nodes in the same community. In this way, its easier for those 10 starting nodes to be connected to the same node (remember that intra-community probability is higher than inter-community), thus being at least the required 2% of its neighbors to spread the infection.

The second diseases starts from high degree nodes, but if these are spread in different communities, the infection would be much slower. To influence the same node at the same time, they would need to rely on inter-community probability, which is, by SBM construction, lower.

We also hypothesized that the strong success of disease 1 in some cases, might be influenced not only by the lower efficacy of disease 2, but also by the choice of the starting community (which is random). Community 1 is not the biggest in terms of number of nodes in it, but has the highest inter-community probabilities, which may help the spreading of the disease. Proving this, though, could be rather complicated: it would be recommended to explore cases where the first disease starts in each community at least 30 times (to be able to use the CLT for statistical testing) but the results are influenced by how the second disease starts too. A trick would be to force the second disease to always choose the same 5 nodes, to keep it consistent. This is not the goal of our analysis, though: we conclude that the first disease is designed better for an attack on an SBM model.

5 Softwares and libraries

- Python version 3.12.7
- Networkx version 3.3
- Matplotlib version 3.9.0