Spreading disease through the air

Justine Weber (261458), David Salathé (262712),
Matyas Lusting (301253), Aurélien Pomini (260498)

January 2019

1 Introduction

Imagine a new disease emerges and spreads quickly worldwide. Or that someone does it on purpose as a form of an attack. Can we predict the way it would propagate via the interconnected airports around the globe? We try to answer that question with the help of signal processing and spectral decomposition approaches, as our aim is not to find an optimal solution by copying papers already showing great result (by using SIR[1], Susceptible-Infected-Removed, models for example). In this work we present you a nice way to graphically simulate a disease attack and we attempt to find the most effective way to infect most airports. The question can be formulated such as: *Which 5 airports should be chosen as sources if one wants to infect the whole world most efficiently ?*

2 Exploration

2.1 From dataset to network

2.1.1 Description of the data

The data of this project can be interpreted as a graph: the nodes are the airports; and there is a strong connection between two nodes if there are many flights between these two airports. For the purpose of this project, we have used two datasets from [4]. The first one - *routes* - has been used to build the graph in the most intuitive way: each entry of this dataset describes a flight route, defined by its *airline*, a *source airport* (`IATA` or `ICAO` code and *OpenFlights* ID), a *destination airport* (`IATA` or `ICAO` code and *OpenFlights* ID), a *codeshare* indicator, the *number of stops* and the *equipment*. We have therefore used the entries of this dataset as edges in our graph.

The second dataset - *airports* - has been used in order to get features and labels on the nodes of our graph. Each entry of this dataset describes an airport through twelve features. Five of them have been useful for our project : *name, city, IATA, latitude, longitude*. We have used them mostly for an interpretation and visualization purpose.

2.1.2 Building a relevant network

In order to build the graph, we have only kept the airports which appear both in the *airports* and in the *routes* datasets. In fact, some entries in the *routes* dataset have a IATA code as source or destination airport, which is not referenced in the *airports* dataset. We have decided to discard these, because we don't have any information on the nodes connected by these edges. Conversely, some airports from the *airports* dataset were not referenced in the *routes* one. Adding these nodes to our graph would have added a lot of isolated nodes, which we didn't want to exploit in our project. Therefore we have also decided to discard these. In total, there are 7184 entries in the *airports* dataset, and 3334 distinct airports (either source or destination or both) in the *routes* dataset. After extracting the *intersection* between the two datasets by inner join operation, we got 3186 airports, which we call "active" airports.

2.1.3 Enhancing the network

The graph we use in this project is symmetric and weighted. Originally, our data describes directed edges, with a source and a destination but we have decided to simplify our graph: each edge of our network represents a flight route, which can be taken both ways.

As we have explained above, in the *routes* dataset the entries have a source airport, a destination airport and an airline. Therefore the same connection from a source and destination airport can be represented several times in the dataset, if several different airlines take it. We have used this to add weights to our edges.
In summary, the edges of our graph are undirected, and the weight of each edge represents the number of airlines taking the corresponding route. Note that if an airline takes the route in both ways, it is counted twice.

2.2 Cleaning & Pre-processing

Once the extraction of the active airports had been done, we ended up with a network which is very close to the one we will use in this project. Before we got our final graph, some data-preprocessing operations were necessary.

First of all, we noticed that our network still comprises seven isolated nodes, which have the following IATAs: `RJA, TIR, LPS, AKI, TKJ, SPB and AGM`. These nodes are present in our network because their are referenced in both datasets, but they are isolated because according to the *routes* dataset, they are only connected to other airports which are not referenced in the *airports* dataset and therefore were discarded. Hence we have removed these airports from our network. After this operation, our network is composed of 3179 airports (nodes).

Secondly we have noticed that one element on the diagonal of our adjacency matrix was non-zero. This corresponds to having a self-loop in the graph, and comes from the fact that one entry in the *routes* dataset has the same airport both as a source and destination. We have considered this as an anomaly, as it doesn't make sense to have a self-loop for a flight route network. Therefore we have set this diagonal element to zero.

2.3 Giant component extraction

The network we got so far is composed of seven connected components. As the goal of this project is to get an insight on how a disease can spread across our network, is doesn't make sense to keep several components which are not connected at all. Moreover, the largest connected component of our network is composed of 3154 nodes. This means most of our graph is in this component and the other components consists in 10, 4, 3 or 2 nodes. Our largest connected component is therefore a giant component and we have decided to keep it as our final network.

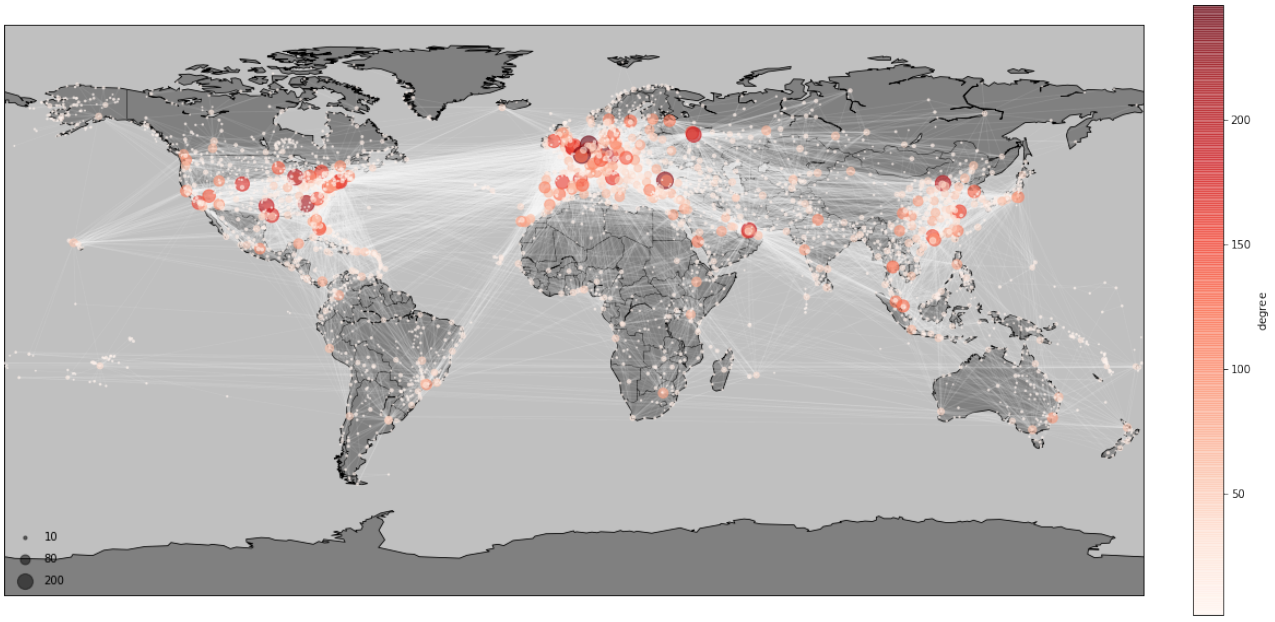

Figure 1: Final network we have used in this project (i.e. only giant component). The darkest and the largest the nodes, the higher their degree.

3 Exploitation

3.1 Spectral Clustering

In this section, we will study how the connectivity of the graph is related to the geographic representation of airports. So the question here would be: *Is there a geographic explanation of why our graph has its actual shape?*

To be able to answer this question, we might need some filtering and good choices of parameters.

3.1.1 Tweaking parameters

After some research [2] we found out that finding a best parameter d (dimensionality reduction for our embedded Laplacian graph) might be hard. Same for the parameter k (number of cluster for the KMeans algorithm), but the latter is already known to be a hyperparameter depending on the current problem. We therefore conclude that both of these parameters should be *grid-searched* (i.e. found by brute-forcing) because a smart solution can be empirically found after a few trials only, assuming both of these parameters cannot be too large. This assumption makes sense for the following reasons:

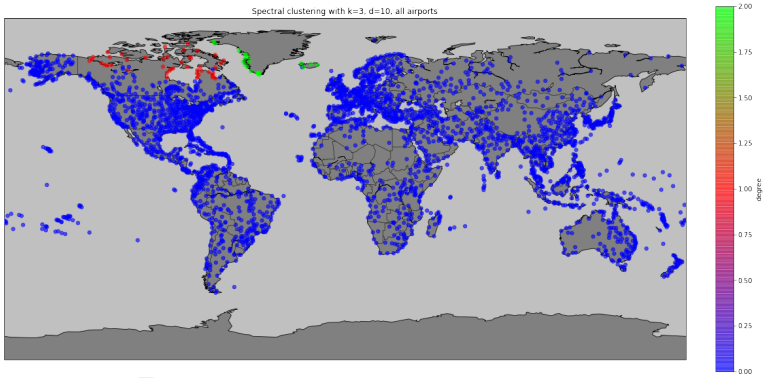

Figure 2: Keeping all nodes of giant component

- Because of the K-Means algorithm, it could suffers of the *curse of dimensionality* problem when comparing distances.
- Because of the nature of our graph, only some k could actually make sense.

Based on these hypothesis, we know we might not reach optimal parameters (in a sense of getting the most meaningful clusters), but we should get satisfactory results. Unfortunately, this is still not the case. By only tweaking these two parameters, it is not enough to get meaningful result (see figure 2). We also need to clean our dataset.

3.1.2 Method for cleaning

We have observed that applying a spectral clustering without filtering any airports (at least not more than the giant component one) is pretty bad. Indeed, even by grid-searching best d and k, by using either between 3 or 10 clustering, there's always a very big cluster including more than 95% of total nodes. That's why we have concluded that we would also need some filtering or more generally, a *selection*. We could have used de-noising methods but we were particularly interested in two other simpler approaches:

- *Considering very small connected airports as noise.* We can simply assume that noise is just the set of all airports having very small different routes, i.e. nodes that have very small degrees. This approach worked surprisingly very well (see figure 3), and a de-noised graph has been obtained by keeping only nodes having degree greater than **13**. Note that

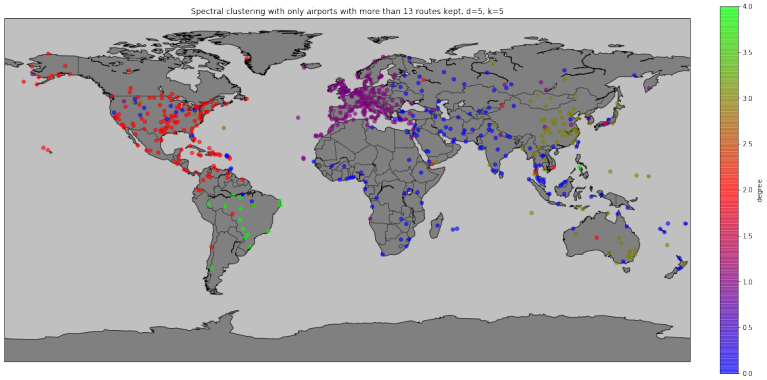

Figure 3: Keeping only nodes of degree higher than 13

here we consider the degree in its unweighted meaning. North America is dominated by the color red, South America by green, Europe by purple, China and Australia in bronze, and finally Africa, middle east and west of Asia are in blue. Blue is the only cluster that does not belong to a specific continent, but it seems to be all "remaining" airports that has the specificity to be poorly linked to some specific hubs.

- *Applying spectral clustering twice consecutively.* If spectral clustering can show that there is a main cluster, it might mean that all other clusters contain the noisy nodes. So instead of wondering which are the noisy nodes, we can let the magical spectral algorithm choose interesting nodes for us. Unfortunately, this method has given terrible results, and nothing seemed to be able to change that. Our hypothesis is that noisy nodes were just split among the three clusters, explaining why this technique hasn't improved anything.

3.2 Kernel models

We apply the heat and rectangle kernels to the different nodes (airports) and observe how the signal (disease) progresses over the graph. Beside is one of the results of a heat kernel applied to a dirac signal, set at an airport in the UK - *Heathrow*. We can see that most of the flights from this airport reach mostly European and US airports.

By tweaking correctly the parameter of the rectangle kernel, it shows the same similitude except that the intensity is already a bit more uniform everywhere. The main difference between them is mostly the intensity at the source airport. The greater the parameter t (of the heat kernel), the more similar the spread we can achieve.

However, our conclusion is to use the heat kernel because the rectangle kernel does not provide any *time* dimension (the parameter t) which could be meaningful for our model.

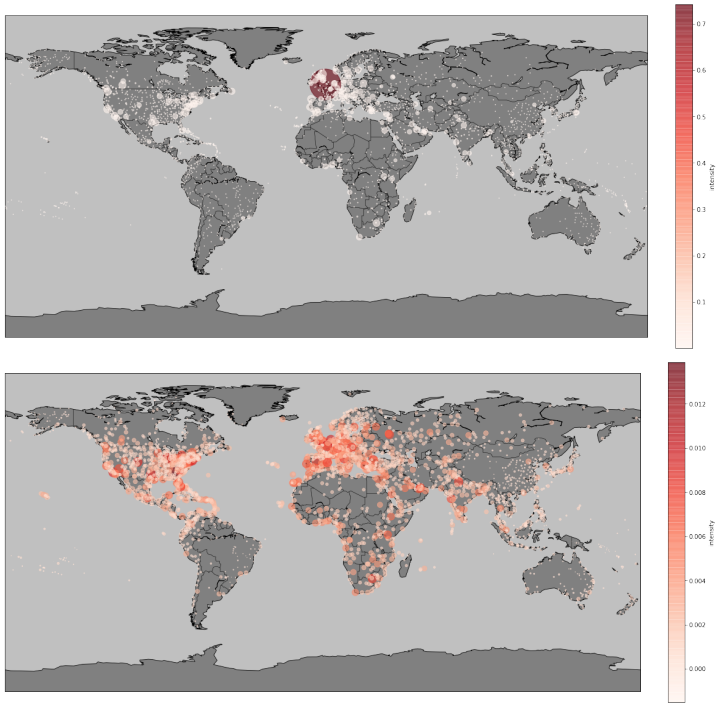

Figure 4: Illustration of the propagation of a dirac signal through our network, using a heat kernel (top figure) versus a rectangle kernel (down figure). The dirac was set on an airport localized in Europe. The darker the color and the larger the node, the higher the signal value.

4 Method and Result

In order to measure the way a disease spreads trough our network, we will base our metric in the following manner:

- Based on the dirac signal, we want to put a signal with **5** entries equals to $\frac{1}{5}$ (and the rest has value 0). We then know that the sum of this signal's diffusion through all nodes sums up to **1**. We will then consider that the whole world would be infected if all airports are uniformly infected.
- We would like to model the propagation in a short amount of time. We're not looking for the best strategy as a virus (and then not considering the SIR model), we have just decided to focus on what could happen in a short amount of time after the attack.

For the first point, we just need to consider that an airport is infected if its node has a value greater than $\frac{1}{\#Airports}$, because if all airports are uniformly infected, all would have this weight. For the last point, we decide to set the parameter t of the heat filter to *0.15* (found empirically), which we consider as a low value, meaning that we look at the spreading after a short period. Finally, we have decided to define our *score* to be the number of infected airports, using the definition of infected described above.

4.1 Top 5 degrees

By simply taking the 5 nodes with highest degree of the graph and plotting them as a signal with a heat kernel, we reached the score of **493**, with the following set of airports: **Atlanta (USA), Chicago (USA), London Heahtrow (UK), Paris (France) and Beijing (China)**. A visual result can be seen in figure 5.

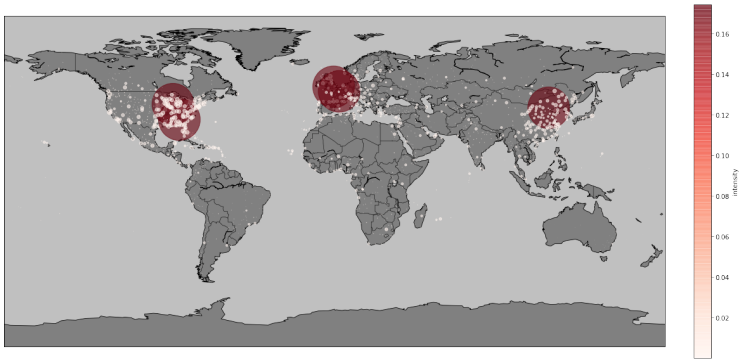

Figure 5: Spreading using heat kernel (t=0.15) and selecting the five nodes having highest degree

4.2 Spectral Clustering

The main problem is that the number of possible k sources over N nodes is not polynomial. A solution is to cluster the network in k groups, then find the best source of each group. The problem becomes quadratic in term of N: each pseudo-dirac is $\mathcal{O}(n)$ when processed with a GFT, and we test N of them. By pseudo-dirac, we refer to a dirac signal which might have k entries equals to k^{-1} instead of one. However we still need to split the network fairly. Since our spectral clustering in the previous section proved to be meaningful, we could use it for this task. We beat the top-5 approach with a score of **566**, by keeping an efficient algorithm. The set of source infected airports is: **Istanbul (Turkey), Amsterdam (Netherlands), Atlanta (USA), Beijing (China) and Sao Paulo (Brazil).** A visual result is shown figure 6. Note that this is quite dependent on the parameter t, if we set it to be a bit higher (i.e. wait a bit longer), *Heathrow* would replace *Amsterdam* for example.

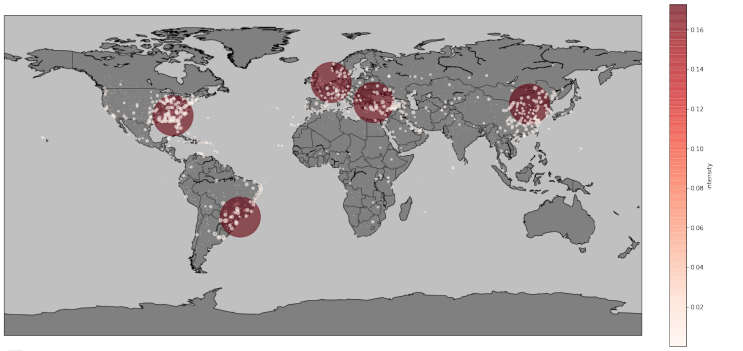

Figure 6: Spreading using heat kernel (t=0.15) and selecting best node on each cluster

4.3 Brute-forcing

In order to compare how efficient are the top-5 and the spectral clustering approaches, we thought we might brute-force the optimal setup. However, a quick calculation proved us wrong. Indeed, if we say that we can measure how many airports are infected by a signal in roughly 1000 ticks, and that we have a powerful 3.5 Ghz processor, brute-forcing combinations of 5 airports over 3174 would take approximately $\frac{1000*\binom{3174}{5}}{3.5*10^9*60*60*24*365} \approx 24$ years. It becomes quickly unscalable since the problem of finding 6 airports would take ≈ 12805 years.

5 Discussion and Conclusion

We are truly satisfied with our results, as we have just proven that the greedy algorithm can be beaten by our spectral approach while keeping a low time-complexity (quadratic in term of nodes if we assume the EigenDecomposition of Laplacian is already computed). The visual results can explain intuitively why our second approach shows better result: Selecting top 5 degree nodes yield some airports that have a lot of connections in common, for example Paris and Frankfurt are strongly overlapping (fig. 5). In the second approach thanks to the additional spectral clustering constraint, all chosen airports are spread around the globe in a better way. We are also very impressed by how well spectral clustering worked. Even when the algorithm had no clue about the geographical location of the nodes, we can clearly observe geographical patterns.

Our method and results might be used for identifying the most exposed airports if a disease outbreak or attack took place as well as to simulate its propagation around the world over time.

References

[1] Smith, D., Moore, L. (2019) The SIR model for Spread of Disease. [online] Available at: https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-the-differential-equation-model [17 January 2019 accessed]

[2] Belkin M. and Niyogi P., (2002) Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. [online] Available at: http://www2.imm.dtu.dk/projects/manifold/Papers/Laplacian.pdf [13 January 2019 Accessed].

[3] Thanou, D., Dong, X., Kressner, D., and Frossard, P. (2017). Learning heat diffusion graphs. IEEE Transactions on Signal and Information Processing over Networks, 3(3), 484-499.

[4] Dataset. [online] Available at: https://openflights.org/data.html [17 January 2019 accessed]