Finding Continents from a Flight Routes Network

O.Boujdaria, F.Dessimoz, A.Duvieusart, A.Vandenbroucque

***Abstract*—In many of the networks that are available today, we can find nodes that are highly connected together compared to the rest of the network. They form communities, and this kind of structure happen to show up very often in real-world networks. Detecting such structures is actually a big challenge, and in the context of this project, we questioned whether it was possible, from our graph representing flight routes, to identify continents by detecting the communities in the network.**

I. Introduction

Among the different choices of dataset available, we chose the graph representing flight routes. The main question we will try to answer in this report is the following: **Is there a way to extract the continents directly from such a graph?**, i.e. does the structure of our network exhibit properties that can let us identify the continents?

This can be done via community detection, so one of our assumption is that continents actually form communities in our graph. By looking at key statistics of our graph, this will first give a more precise idea about its structure. We then review techniques seen in class, such as **Spectral Clustering**, but we will also introduce notions such as the **modularity** in a graph, which will be useful when presenting alternative algorithms.

The report is organized as follows. We first present the network and its main properties, then discuss the methods used in our work and provide the reader with the obtained results. We also give a comparison of the various algorithms used. Eventually, we summarize in the conclusion the general trends observed in the project.

II. Data Acquisition And Graph creation

A. A first look at the data

The graph we used is stored as two *.dat* files, and contains information about flight routes, and also airports. We present in Table 1 the features we kept for this project.

Columns	Description
Airline ID	Unique OF identifier for airline .
Source Airport ID	Unique OF identifier for source airport
Destination Airport ID	Unique OF identifier for destination airport
Airport Id	Unique OF identifier for this airport
Name	Name of the airport
City	City of the airport
Latitude	Latitude of the airport
Longitude	Longitude of the airport

Table I
Original Dataset Organization

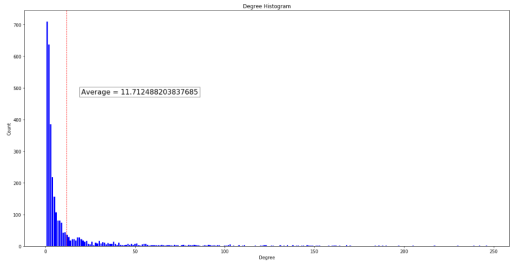

Figure 1. Degree Distribution

In a first step, we clean the data to only keep airports that appear in both datasets, and add extra information in order to easily compute distances between airports. We then create the graph using routes as edges, and keep name labels for the nodes.

III. Main properties of the network

First, we explore the connectivity of the graph and plot the degree distribution represented in Figure 1. We can see that the distribution seems to follow a power law. Next, we check the connectivity of our graph, which has 7 connected. Out of these, we have one main component containing more than 95% of the nodes surrounded by 6 small components (in terms of the number of nodes they contain).

We now proceed to analyzing the largest component :

Statistic	Value	Description
Graph Density	0.373%	Actual # of egdes divided by the maximum # of edges.
Ave. Clustering coef.	0.4914	The clustering coefficient of a node is the fraction of the nodes neighbors that are connected.
Diameter	12	The diameter of the graph is the length of the longest shortest path between any pair of nodes.

Table II
Largest Connected Component Analysis

A. Centrality Measures of the graph

Suspecting the predominance of certain nodes in the dataset, we run various centrality measures on our graph in order to find the most relevant nodes in means of betweenness, of eigenvalues, and finally as pagerank coefficient.

The betweenness centrality of a node v is the sum of the fraction of all-pairs shortest paths that pass through v. The eigenvector centrality computes the centrality for a node based on the centrality of its neighbors. The eigenvector centrality for node i is the i-th element of the vector x defined by the equation $Ax = \lambda x$, where A is the adjacency matrix and λ is the highest eigenvalue. Finally, the pagerank computes a ranking of the nodes in the graph G based on the structure of the incoming links, and was originally designed by Google as an algorithm to rank web pages.

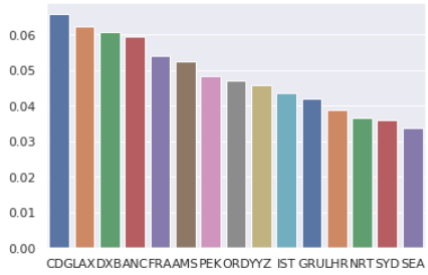

Figure 2. Betweenness

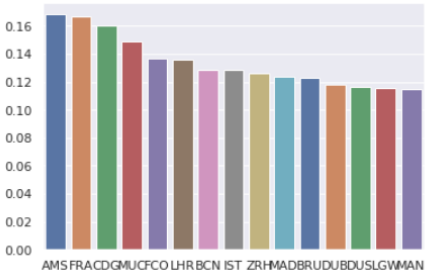

Figure 3. Eigenvalues

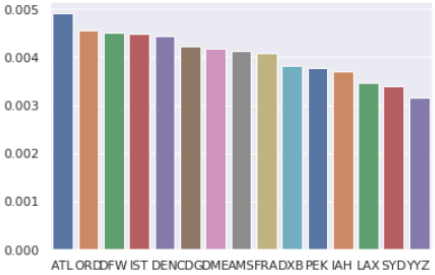

Figure 4. PageRank

The results are summed up in the following figures, where we see that the top ranked airports in the merged final ranking are Paris Charles De GaulLe, Amsterdam Airport Schiphol and Frankfurt International Airport. which comes out without surprise as they are pivot airports for the world flight traffic.

Rank	Airport Name, City
1	Charles de Gaulle International Airport, Paris
2	Amsterdam Airport Schiphol, Amsterdam
3	Frankfurt am Main International Airport
4	Atatrk International Airport, Istanbul
5	Dubai International Airport, Dubai

IV. Detection of communities

We now dive into the main part of this project, which is the community detection. There exists many different ways for finding community structures within networks, and we decided to compare those seen in class with some other ones. While we will see that for our network using spectral clustering do not give us the results we hope, using modularity as a measure of quality of partitioning a graph into communities will drastically improve the results. Moreover, computing the edge betweenness for the links in our graph will also lead us to a different algorithm.

We first start by reviewing the spectral clustering method for finding communities.

A. *Spectral clustering*

Let's first recall why spectral clustering works. What define communities is the fact that there are many edges within communities compared to the low number of edges between communities. One way of representing this formally is with the *RatioCut* [1] equation, defined as follow:

$$RatioCut(A, \bar{A}) = \frac{1}{2}\frac{C(A, \bar{A})}{|A|} + \frac{1}{2}\frac{(A, \bar{A})}{|\bar{A}|}, \qquad (1)$$

where the $(A, \bar{A})$ is a cut of a graph and $C(A, \bar{A})$ the sum of the weight of all edges that goes from A to $\bar{A}$. By minimizing the RatioCut, this gives us a way to split the graph into two communities. Note that this can be generalized for $k > 2$, and thus the optimization formulation becomes

$$\min_{A_1, A_2, ..., A_k} RatioCut(A_1, A_2, ..., A_k). \qquad (2)$$

Using spectral clustering gives us a way to efficiently solve a relaxed version of this problem, which is still a good approximation.

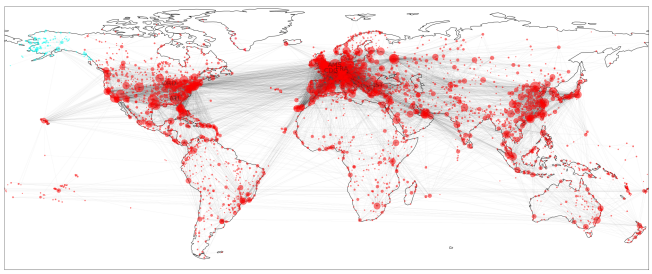

Figure 5. Communities obtained with spectral clustering

As we have seen in class, the main tool for spectral clustering is the graph Laplacian matrix, defined as $L = D - W$, where D is the *degree matrix*, and W is the *weighted adjacency matrix*. Now to find the best A_i, the algorithm works as follows:

1) Compute the matrix F of the first k eigenvectors of L, for some given k.
2) Apply K-Means to the rows of F to obtain the cluster assignments.

Note also that by looking at the spectrum of L, that is the plot of its eigenvalues, this can gives us insights about the structure of the graph. Indeed, we should be able to observe a gap between the k^{th} and $(k+1)^{th}$ eigenvalues if the graph has k clear communities.

B. The Girvan-Newman algorithm

The Girvan-Newman algorithm rely on a another type of measure which is called the *edge betweenness*, which for an edge is defined as the number of shortest paths between any pairs of nodes that run along it. What the algorithm does on a high-level is to iteratively remove edges from the original graph, leading to a disconnected graph whose connected components are the communities. More precisely, it works as follows:

1) Compute all shortest paths for all pairs of nodes in the graph.
2) Assign to each edge the number of shortest paths they are a part of (i.e. their betweenness).
3) Delete the edge with the highest betweenness
4) Repeat step 2-3 until the graph is separated in two non-communicating subgraphs.

Then we can recursively apply this algorithm to the biggest existing subgraph as we made the assumption that it is the one still containing different communities. It is quite intuitive to notice that edges with high betweenness represent inter-communities edges since they are some kind of bottleneck, i. e. many shortest paths will go trough these edges.
Since computing all shortest path is computationally expensive we decided to implement a randomized version of this algorithm where at each iteration we sample uniformly at random a subset of edges, which gives us a subgraph, and we then compute the edge betweenness on this subgraph of smaller size. Using this scheme, this allows us to reduce the running time of the algorithm while making the right choice of edge to remove at each step with high probability.

C. Modularity maximization

A popular way for detecting communities in a graph is by using a measure called *modularity*. Modularity is a measure of the quality of a particular partition of a network into communities. It works by comparing the number of edges inside each community to the number of edges that would be expected if the nodes were connected at random (Erdos-Renyi model), without any preference for edges within or outside the community. More formally, the modularity Q for a graph $G(V, E)$ is defined as:

$$Q = \frac{1}{2|E|} \sum_{i,j \in V} \left(A_{ij} - \frac{d_i d_j}{2|E|} \right) \delta_{C_i C_j}, \tag{3}$$

where A is the adjacency matrix, d_i is the degree of node i, δ_{ij} is the Kronecker function, equals to 1 if both node i and j are in the same community, and C_i is the community to which node i is assigned.

With this formulation, it turns out that the higher the modularity, the better the partition is, since networks with high modularity have many edges between nodes in the same community but very few edges between nodes in different communities.

So a natural way to find communities is simply to find a partition of our graph which has the maximum modularity, between all possible partitions. This is of course not possible, because the search space is exponential in the number of nodes, and so we present algorithms such as Greedy Modularity Maximization, that aim to find the a partition of the graph into communities such that the modularity is particularly high.

1) The CNM algorithm (greedy modularity optimization): The CNM algorithm is a greedy algorithm where at each step, we merge existing communities such that it allows us to get the maximum gain in modularity. Let us first define the change in modularity more formally. Let C_i, C_j be two communities. The change ΔQ_{ij} in modularity after merging C_i and C_j to form a new partition of our graph is given by

$$\Delta Q_{ij} = \frac{1}{2|E|} 2 \sum_{u \in C_i} \sum_{v \in C_j} \left(A_{uv} - \frac{d_u d_v}{2|E|} \right). \tag{4}$$

We now describe the CNM algorithm:

1) First assign each node $u \in V$ to its own community C_u.
2) Find a pair of communities C_i, C_j such that ΔQ_{ij} is maximum, and merge these two communities into one.
3) Repeat step 2. until there is only one community left.
4) Output the partition which has the largest modularity.

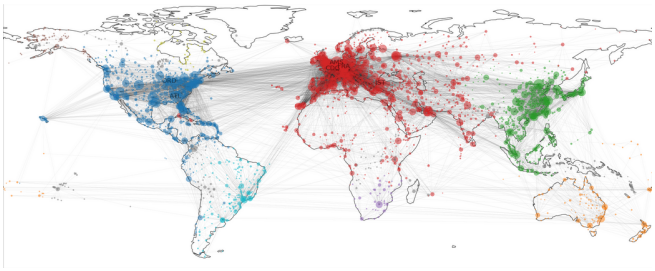

Figure 6. Communities obtained with Girvan-Newman algorithm

So after initialization, the algorithm performs $(|V| - 1)$ merging steps, resulting in $|V|$ different partitions of the graph. Out of these $|V|$ partitions, the algorithm outputs the one which has the largest modularity, since it approximates the maximum modularity best.

2) The Louvain method: The Louvain method is also a method that aim to maximize the modularity of a graph. It consists in two steps that are repeated many times until the modularity of the graph is no longer increased.

The initialization is done by assigning each node to its own community. Thereafter it iterate on the two following steps :

1) Assign each node to a community that locally optimizes the modularity. More precisely it iterates on each node in the network, removes it, and compute the change in the modularity if we place this node in the community of one of its neighbor.
   - If we are not able to obtain a better modularity with one of these changes, we keep the node in its current modularity.
   - If we obtain a better modularity, we move the considered node to the community where the modularity is maximized.
2) Construct a coarse grained network with the communities found in the first step.

The algorithm iterates on these two steps until the modularity cannot be increased anymore.

V. Results and comparison of methods

We present here the results obtained for the various algorithms and provide a comparison of these methods.

A. Spectral clustering

The results for spectral clustering can be seen in Figure 5. As mentioned before, this method did not perform good at all. We see that the algorithm was able to find, when specifying 3 clusters, a tiny community in Alaska, and another one in Greenland. This is due to the structure of the graph and how spectral clustering works. Indeed, this method is effective when there are clear communities in the graph, but for most of our graph, it is not the case. For example, taking America and Europe, even tough they may be communities, there are still many edges between them, which makes it not possible for spectral clustering to detect them.
However, an interesting insight we can get is that in Alaska and Greenland, there are small but clear communities. This may be explained by the fact that many of the airports in those countries are not meant for commercial use, and are thus almost not connected to the rest of the world.

B. Girvan-Newman

The results are shown in Figure 6, and we see that it looks like the method is effective, since the continents are easily detected, but the price to pay is giving up on small communities. Indeed, separating a connected graph with Girvan-Newman often gives two subgraphs of very different sizes. From the figure we distinguish easily 6 communities counting southern Africa. For visualization purposes, we only colored the 6 first communities found, however when the algorithm continues running, it finds more and more precise communities.
Another drawback of Girvan-Newman is the time complexity. Indeed at each step, we need to compute the shortest paths between all pairs of nodes, which makes it almost infeasible on large graphs. The results we show are actually the output of the randomized version, which is a lot faster than the original one.

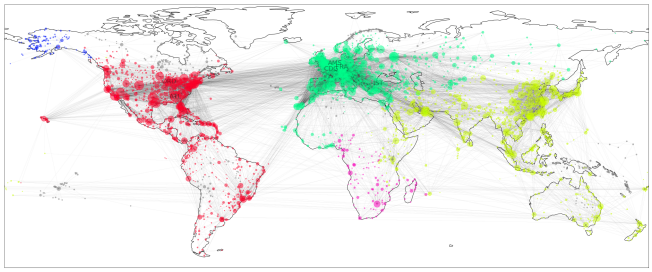

Figure 7. Communities obtained with Greedy modularity maximization algorithm

C. Modularity Maximization

Note that for modularity maximization, both results are quite similar, but the Louvain method still leads better results in terms of modularity. We see that with this simple formulation of a quality of a partition, this gives us very good results in a very reasonable amount of time, which is key when processing large graphs.
With those methods, we get the results shown in Figure 7 and Figure 8. We are able to detect the main continents such as America and Europe. We see that again, Alaska is detected as a community, and Asia is community together with Oceania.

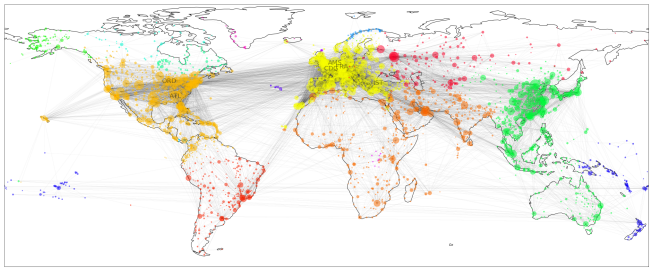

Figure 8. Communities obtained with Louvain algorithm

D. Comparison

In this section, we will compare the different algorithms on different levels. For that we decided to measure the performance of the algorithms using two metrics: *modularity* and *coverage*. Modularity was already presented before and coverage is a natural measure for how good the partition is. The coverage of a partition is defined as the ratio of the number of intra-community edges to the total number of edges in the graph.
We also included the complexity of the algorithms, as it is very relevant when working with large amounts of data. The table below sums up the key values:

Algorithm	Complexity	Modularity	Performance
Spectral clustering	$O(\|V\|^3)$	0.023	0.999
Girvan-Newman	$O(\|E\|^2\|V\|)$	0.595	0.914
CNM	$O(\|V\|(\|E\|+\|V\|))$	0.603	0.907
Louvain method	$O(\|E\|)$	0.659	0.901

Table III
COMPARISON OF THE VARIOUS ALGORITHMS

One can notice how time complexity can vary from one method to the other. From the worst one which is Girvan-Newman, to Louvain which has an impressive running time of $O(|E|)$ [4].
In terms of quality of partition, when considering coverage, we see that all partition are very good, with a score always greater than 0.9. This measn that more than 90% of nodes are inside communities, and not between communities. The best one is when we use spectral clustering, which makes sense with the way it performed since it detect the "clear" communities, i. e. those which are almost not connected to other communities.
When considering modularity however, Louvain is definitely the best, and we see how spectral clustering performs poorly.

VI. CONCLUSION

There exist various ways to detect communities in graphs, each having his own specific procedure. For example, some take advantage of the spectral properties of the graph, while others attempt to maximize the quality of a partition of the graph.
In any case, it is never simple to find the best method to detect those cluster. Indeed, we saw that this is very dependent from the graph structure. With a network such as ours, that is actually quite connected and with no clear communities at first glance, spectral methods perform poorly. Detecting communities as big as continents with many interconnections between them is not possible using them. However with methods maximizing modularity, it seems that we are able to perform well even with such graphs. This turned out to be also the case when using the edge betweenness measure.
It becomes then natural to reduce the time complexity of methods, leading to algorithms like the Louvain method. We are also able to gain a lot of speed by implements a randomized version of the Girvan-Newman algorithm, while preserving its performance.
Overall, communities is a vast subject, as it allows not only to discover the general structure of the graph, but also analyze smaller communities and get insights about specific nodes. There are many exciting algorithms that have been created in the last few years, and so this gives plenty of opportunity to further understand this subject and apply it to many real-world networks.

REFERENCES

[1] U. V. Luxburg, "*A tutorial on spectral clustering*," *Stat. Comput.*, pp. 395–416, 2007.

[2] M. Girvan and M. E. J. Newman, "*Community structure in social and biological networks*," *ArXiv.org*, pp. 2–3, 2001.

[3] F. Botta and C. I. del Genio, "*Finding network communities using modularity density*," *ArXiv.org*, pp. 1–3, 2016.

[4] S. Papadopoulos, "*Community Detection in Social Media*," 2011.

[5] S. Iyer, "*Simple explanation of the Louvain Method*," https://www.quora.com/Is-there-a-simple-explanation-of-the-Louvain-Method-of-community-detection?fbclid=IwAR1R$_g i_P WU1uCgENUAxFxX c3e_r g5AbzQ - BAbZKUIes08GZ_q RLXRmgCr0$.