**Final report**

Analysis of bus public transport in Malaga using graphs

Learning from Networks, University of Padova, 2022-23

Jesús Moncada Ramírez

# 1. Introduction

The **motivation** for this project was to analyze the main features of the bus public transport system of the city of Málaga, Spain. Particularly, we wanted to analyze its frequency and speed. This analysis has taken place by creating a graph data structure with all the information about bus lines, bus stops, routes, and timetables… and extracting useful characteristics from it using graph algorithms.

The initial **objectives** were seven for this project, and finally, we had time to complete four of them: (1) “Create a graph representing all the bus lines and stops…”, (2) “Compute some graph analytics at the node level…”, (6) “Do some experiments with random graphs…” and (7) “Create an analysis report with the conclusions”. This way, our work has been centered on computing graph analytics for nodes, especially node **closeness and betweenness centralities**. After this, we studied some **random graphs** following diverse models to discover whether the centrality measures were relevant. Thus, the project title could have changed to “Analysis of node centralities and its significance in the bus public transport of Málaga”.

The **dataset** used was obtained from Malaga’s council open data site ([*Datos abiertos, Ayuntamiento de Málaga*](https://datosabiertos.malaga.eu/)). On this web page, the administration of the city provides several CSV files about the bus system organization. The availability of this data has been a very relevant motivation to choose the city of Málaga as an object of our study. The same data was searched for Padova, without success.



Figure 1: Urban Buses Map of Málaga

In section 2 we will explain all the work that has been done. In section 3 we will discuss some problems we found during the project development and the proposed solution. In section 4 we will present some results and information obtained after our study of the graph. Finally, in section 5 we will present the conclusions.

The project has been uploaded to [this repository](https://github.com/jemoncadar/malaga-bus-graph.git) in GitHub to maintain higher version control and to make it public.

# 2. Work done

All the code developed for this project can be found in a Jupyter Notebook file, *ProjectNotebook.ipynb.* All the code is in *Python* 3 and we use *NetworkX* as the main library to work with graphs. This file also contains some explanations, more technical details, and comments in a notebook way to facilitate its understanding. We have divided it into some sections that will be explained separately.

## Section 0: Installation issues

Here we comment on the versions of the libraries used that must be installed to run all the code properly. This has been discovered by going into bugs and searching on the internet.

## Section 1: Importing the datasets

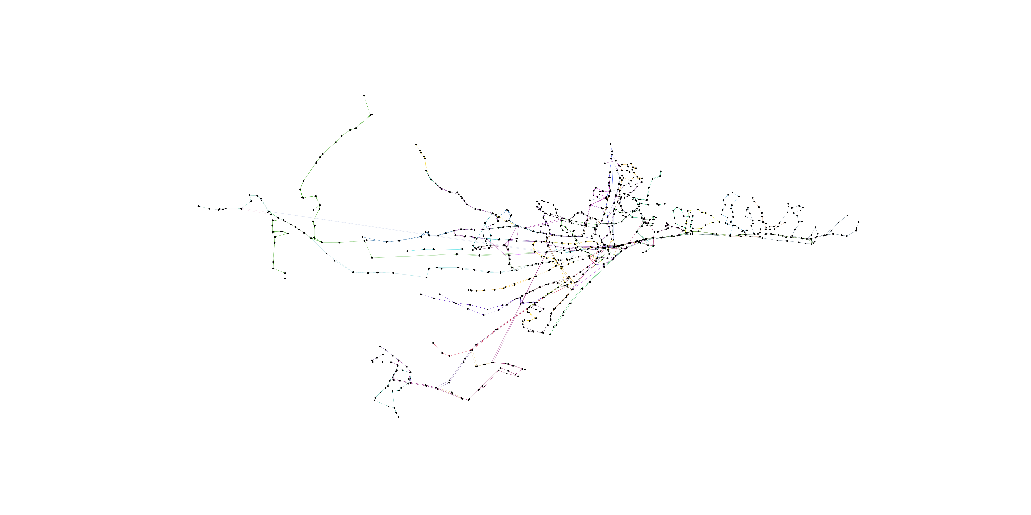


Figure 2: Line-based image generated of the graph

We created the proper Python data structures for the CSV files of our dataset. For this, we have used the library *pandas*; this way the data structures are *DataFrames*, which allow us to perform a lot of implemented operations. After importing, we renamed some column labels of the data, translating them from Spanish to English. Finally, we provided a brief view of each dataset file showing some samples (using the library *tabulate* to print it nicely). After all, this section oversees data preprocessing.

## Section 2: Creating the graph

Here, we created the graph, using a *NetworkX* data structure called *MultiDiGraph* (a graph that allows multiple directed edges between nodes). We added the nodes (all the bus stops in the city), and the edges (all the connections between two stops belonging to the same line). Before adding the edges, we chose a measure of the quality of each edge, this is, its weight (which has been called ***velocity***). The *velocity* of an edge has been decided to be inversely proportional to the distance between the two stops and directly proportional to the number of buses in its line. It is also influenced by the product by a constant that has been set empirically. The formula of an edge belonging to the line , is defined as:

were is constant, is the number of buses of line (in some time interval) and is the distance between the stops and . All the values we got have been normalized and applied to a threshold.

Finally, in this section we generated an image of the generated graph to check if the elements were properly created. Two images have been generated, one of them where the edge color depends on the bus line, and the other where the edge color depends on the *velocity* attribute. Some tests were made in order to display our graph over an *HTML* interactive map using the library *Mplleaflet*, but this library has still some bugs, so this part was classified as optional.

## Section 3: Node centralities

In this section we calculated closeness centrality and betweenness centrality for all nodes in our graph. We had to define a new version of our initial graph as the *NetworkX* algorithm for centrality expect a distance weight for each edge, this is, higher the value poorer the connection between two stops. This way we defined the *distance* of an edge, , as the inverse of its *velocity*. If an edge was “very fast”, its *velocity* value was high, so now the *distance* will be low.

After this, we calculated the closeness centralities for nodes in two cases: (1) considering every edge weight as 1 (not taking into account *distance* attribute) and (2) considering the *distance* as attribute weight.

The betweenness centralities for every node were also computed in the new version of the graph. The graph for the betweenness centrality was forced to have other modifications because *NetworkX* does not implement the betweenness centrality algorithm for the graph data type. It is only supported for directed graphs (no multiples edges allowed). Thus, we created a function to convert a *MultiDiGraph* to a *DiGraph*. In this conversion we need to specify what to do with the edges attributes, as some previous edges can be converted to the same one. The implementation decision was to calculate the maximum among the *velocities* of the previous edges and assign it to the new edge. Again, the betweenness measures were computed for cases (1) not considering *distance* and (2) considering *distance*. All the results will be commented in the *4 Results*.

## Section 4: Random graphs

Now, to evaluate the significance of the measures obtained in the previous section, we generated several random graphs and calculated the same measures on them. The random graphs were generated following two models. (1) The **Chung-Lu model**, this is, probability of edge to appear is:

(2) The **Given Degree Sequence model**, this is, each node will maintain the same degree in the random graphs. This two models were already implemented in *NetworkX*.

In both cases, the results are evaluated using z-scores and p-values.

# 3. Problems found

* Algorithms not implemented.

# 4. Results

# 5. Conclusion