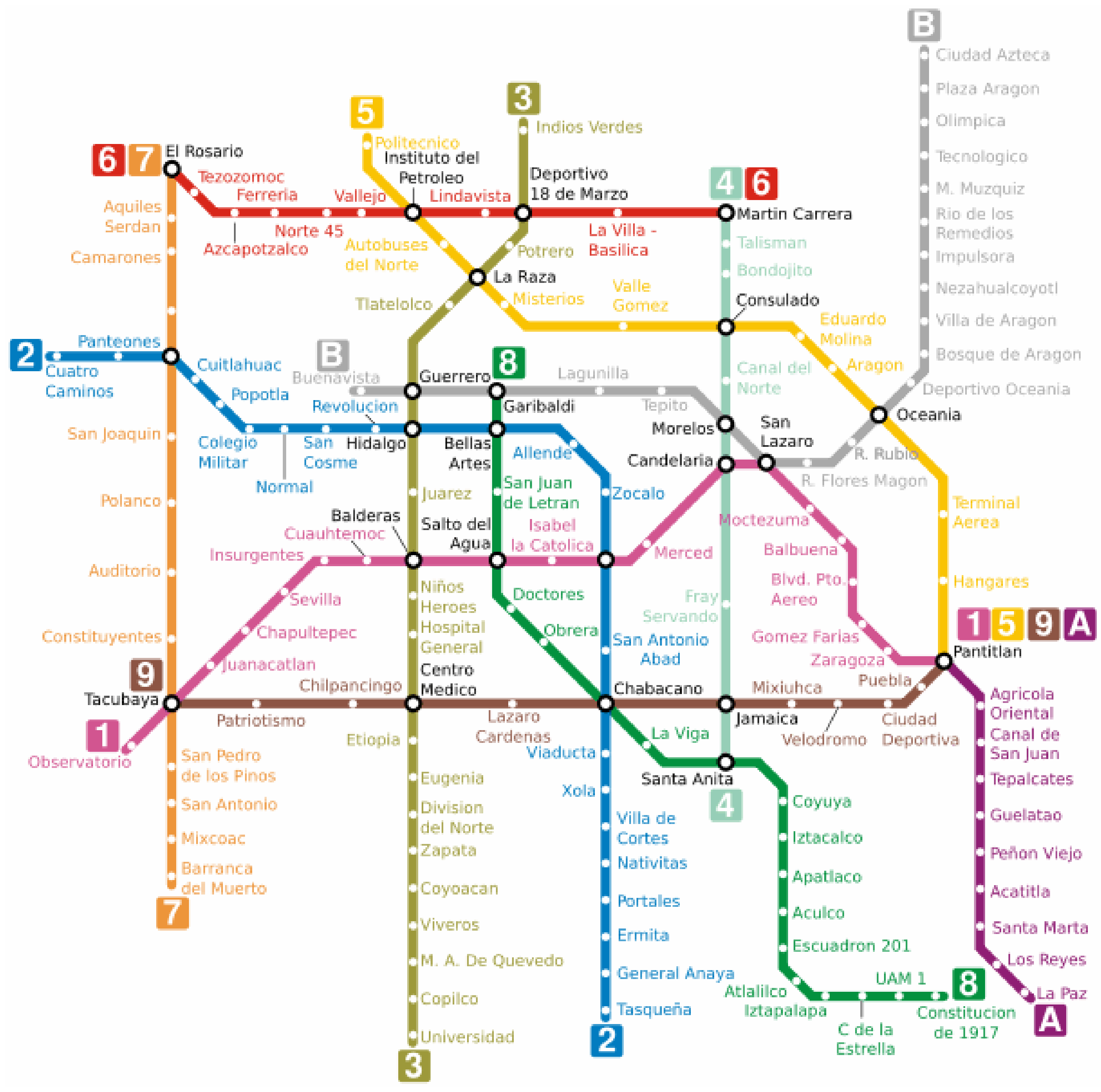


ARTIFICAL INTELLIGENCE PROJECT

APPLICATION OF GRAPH THEORY : A SMART MAP OF A TRANSPORT NETWORK

EDOUARD MIGNIEN – PAULINE BERNARD
KYUNGPOOK NATIONAL UNIVERSITY
2021

Table of contents

Introduction

We are two students in embedded systems engineering at the French engineering school ECE Paris. The concept of artificial intelligence was new for us this semester. That is partly why we chose this course at KNU. The university exchange offered us the opportunity to diversify our technical skills in addition to opening up to a new culture.

Nonetheless, the concept of graph theory is familiar to us and we found out thanks to the AI course of this semester that graph theory could be an application of the notion of artificial intelligence. This is one of the reasons why we have chosen to focus on the field of graph theory. Also, we think that this domain extended to the AI domain allows us to challenge ourselves and pushes us to surpass ourselves in our research and programming work for this project.

Graph theory can be a real asset in artificial intelligence for several points:

- It can be used to analyze geographical data (roads, transportation, railroads or airways) which is the application we have chosen for the project.

- It is useful in the fight against the spread of certain diseases (especially in the current health context and the Covid-19 crisis we are facing) through a monitoring of infected cases or contact cases.

- It also permits to visualize relationships between different objects (people, phones etc) and can support the study of the Internet of Things.

- It is also applicable in the field of medicine and can visualize a neural network.

- And many others applications

As graph theory establishes links between data and creates relationships, it provides artificial intelligence with precise context. It permits the use of AI to solve specific problems.

Our project is therefore based on graph theory to offer an intelligent mapping application.

This document is the report. It first presents an overview of the calculations that the application uses. Then, it describes our project, i.e. the choice of the application subject: a transport network and how our application works. It then gives an analysis of the results of our algorithms. It finally ends with a conclusion opening on other possibilities of use of the application.

Overview of the calculations

In our opinion, the most interesting and accessible way to associate artificial intelligence and graph theory is to analyze the centrality and accessibility of the vertices of a graph according to different criteria. Graph theory itself permits to analyze the degree of importance of the joins of a graph, which makes it an obvious application in the field of AI.

We sought to analyze the importance of vertices based on our computer science courses in France. We analyze the graph thanks to four different calculations more or less precisely which permit thereafter to highlight the most important information and data. The calculations are a way for the AI to understand the graph and to get the necessary information to apply to the context in which it operates.

1. Degree centrality

The 1st calculation we use is the degree centrality calculation. The degree of a vertex is the number of edges of which this vertex is an end. The degree centrality simply consists in analysing the importance of a vertex by observing its degree. The higher the degree of a vertex is the more important the vertex is because the higher the degree is the more direct links the vertex has with other vertices.

2. Eigenvector centrality

This calculation is similar to the degree centrality calculation but it goes further in the analysis of the importance of a vertex. It analyses the importance of a vertex by calculating the importance of its neighbors.

It establishes the idea that if a vertex is adjacent to an important vertex, then that vertex is more valuable than a vertex adjacent to a less important vertex.

The corresponding algorithm initializes the index of each vertex to 1.

$$Let\ say\ that\ we\ have\ n\ verteces$$

$$\forall i\ for\ 0 < i < n, \qquad let\ s_i\ be\ the\ current\ vertex$$

$$Let\ x_{eigenvector}\ be\ the\ index$$

$$x_{eigenvector}(s_i) = 1$$

Then, the algorithm makes several iterations.

For each vertex, it computes the sum of the indices of its neighbors.

$$\forall i\ for\ 0 < i < n\ and\ \forall j\ for\ 0 < j < n$$

$$sum(s_i) = \sum_{s_j \in Neighbors(s_i)} x_{eigenvector}(s_j)$$

s_j is a neighbor of s_i. It belongs to a list of neigbhors of s_i : $Neighbors(s_i)$

The algorithm computes λ which is the the square root of the sum of squared $sum(s_i)$

$$\lambda = \sqrt{\sum_i sum(s_i)^2}$$

Then, the algorithm finally calculates again the index.

$$x_{eigenvector}(s_i) = \frac{sum(s_i)}{\lambda}$$

The iteration is repeated until λ stabilizes.

3. Closeness centrality

This calculation gives the importance of a vertex in comparison to the others by calculating the average distance between this vertex and the others. The distance between two vertices is here the shortest path between them.

$$Let\ say\ that\ we\ have\ n\ verteces$$

$$\forall i\ for\ 0 < i < n, \quad let\ s_i\ be\ the\ current\ vertex$$

$$Then\ we\ have\ n-1\ other\ verteces$$

$$\forall i\ for\ 0 < i < n\ and\ \forall j\ for\ 0 < j < n,$$

$$Let\ d(s_i, s_j)\ be\ the\ distance\ between\ two\ verteces\ s_i\ and\ s_j$$

The algorithm calculates the number of vertices minus the current vertex (that is to say : $n - 1$) divided by the sum of the distances between two vertices s_i and s_j $(d(s_i, s_j))$.

$$x_{closeness}(s_i) = \frac{n-1}{\sum d(s_i, s_j)}\ with\ s_i \neq s_j$$

To find the shortest paths between two vertices, we use the Dijkstra algorithm. It computes the shortest path between two vertices, i.e. the path for which the sum of the weights of the edges and the degrees of the vertices is the lowest.

4. Betweenness centrality

This calculation evaluates the importance of a vertex according to whether it is on the shortest path connecting two vertices or not. In fact it computes the frequency with which a vertex is on the shortest paths connecting two other vertices. The betweenness centrality is independent of the degree of the vertex. It corresponds to the usefulness of the vertex within the graph.

The algorithm computes the sum of the number of shortest paths between two vertices passing through the current vertex divided by the number of the total number of shortest paths between the same two vertices.

It corresponds to the betweenness centrality index.

$$x_{betweenness}(s_i) = \sum \frac{n_{shortest_{paths_{jk}}}(i)}{n_{shortest_{paths_{jk}}}} \ with\ j < k$$

Where s_j and s_k are the two vertices for which the shortest paths are calculated.

To find the shortest paths between two vertices, we use the Dijkstra algorithm. Once the shortest path is found, we go from predecessor to predecessor to check if the current vertex belongs to the path.

The Dijkstra algorithm is slightly modified so that it can give the number of shortest paths between two vertices (it initially gives only one shortest path). Modifying it in this way permits us to calculate the centrality index.

To adapt these calculations to any network and make them independent of the size of a graph, we have normalized them.

1. Degree centrality

To normalize the degree centrality index, we divide the degrees by $n-1$, which is the maximum degree of a vertex in a graph.

2. Eigenvector centrality

The eigenvector centrality index is already normalized.

3. Closeness centrality

To normalize the closeness centrality index, we multiply the non-normalized proximity index by $n-1$.

4. Betweenness centrality

To normalize the betweenness centrality index, we divide it by the maximum value it can reach which is $\frac{1}{2}C_{n-1}^2 = \frac{n^2-3n+2}{2}$.

Centrality	Details of the algorithms	Complexity
Degree	1 *for* loop, 1 calculation	$O(n)$
Eigenvector	1 *do…while* loop, calculations	$O(2n\log(n))$
Closeness	2 *for* loops, Dijkstra, calculations	$O(n(n^2+2n))$
Betweenness	3 *for* loops, Dijkstra, calculations	$O(n((n^2+n)+n^2))$

Complexity of all the algorithms of calculations

All these calculations can hold an essential place in the field of artificial intelligence. As for example, defining the importance of a vertex via the degree centrality or the Eigenvector centrality would permit to know the number of people that a person with a certain disease frequents. The closeness centrality permits to observe the propagation efficiency of a disease. The closeness centrality and betweenness centrality would permit to offer a user a correct and efficient route to go from one point to another in a transport network. All the indices of centralities could also permit to know the number and the proximity of the relationships between different people on a social network. And there are many more possibilities.

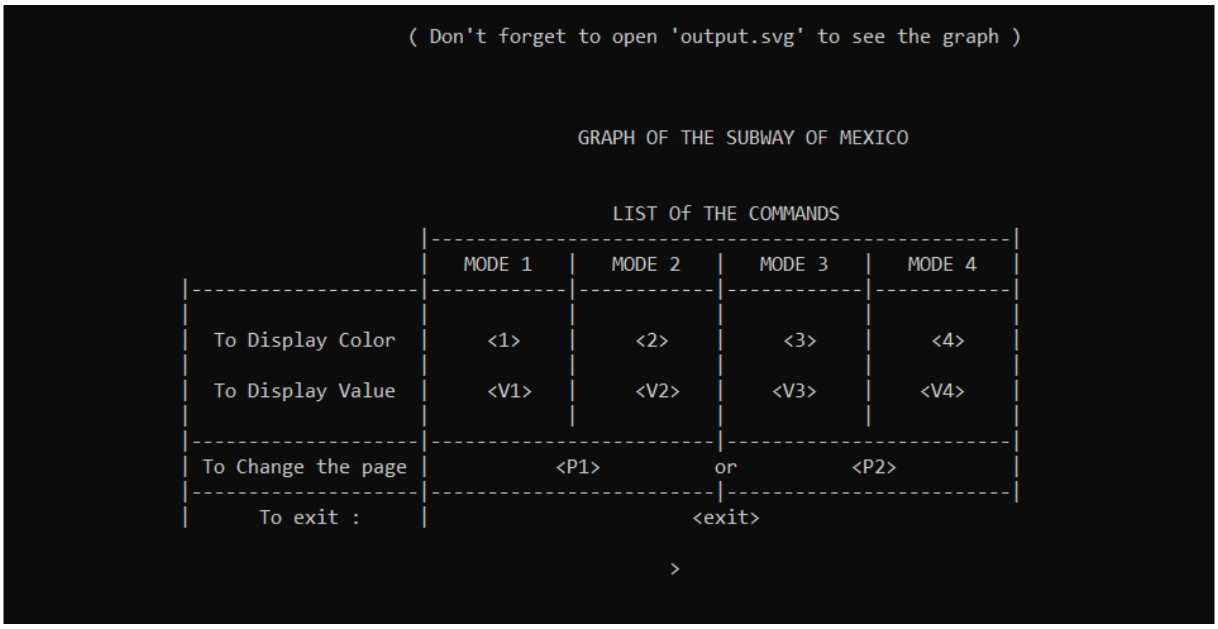
Project description

Our project is based on the transport network of Mexico. It reproduces the subway network and the rail network. It uses the calculations described before to

It operates with two factors :

- The console (.exe file)
- The output (.svg file)

To make it run and understand how it works, we need to open these two files.



```
                            ( Don't forget to open 'output.svg' to see the graph )


                                       GRAPH OF THE SUBWAY OF MEXICO


                                          LIST Of THE COMMANDS
                          |-------------------------------------------------|
                          |   MODE 1   |   MODE 2   |   MODE 3   |   MODE 4   |
        |-----------------|------------|------------|------------|------------|
        |                 |            |            |            |            |
        | To Display Color |    <1>     |    <2>     |    <3>     |    <4>     |
        |                 |            |            |            |            |
        | To Display Value |    <V1>    |    <V2>    |    <V3>    |    <V4>    |
        |                 |            |            |            |            |
        |-----------------|-------------------------|-------------------------|
        | To Change the page |        <P1>        or        <P2>            |
        |-----------------|-------------------------|-------------------------|
        |     To exit :   |                    <exit>

                                              >
```

Screenshot of the console

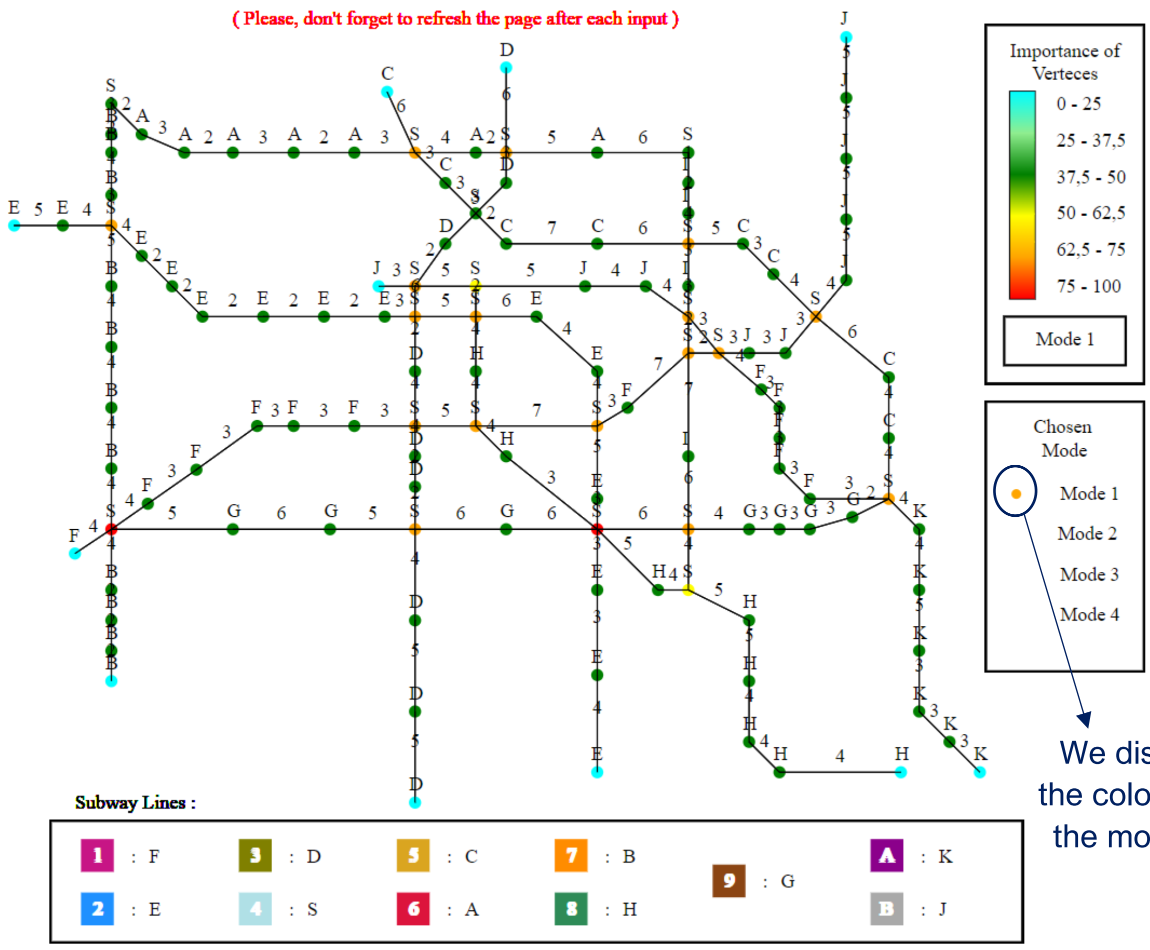
4 different modes permit to chose the calculation we want :

- The mode 1 – the degree centrality
- The mode 2 – the eigenvector centrality
- The mode 3 – the closeness centrality
- The mode 4 – the betweenness centrality

Each mode permits to display colors or values according to the values of the centrality indices.

The different endpoints of the graph, which represents the subway stations or the train stations, are represented according to the calculations (the mode). They can be represented with colors or/and with values.

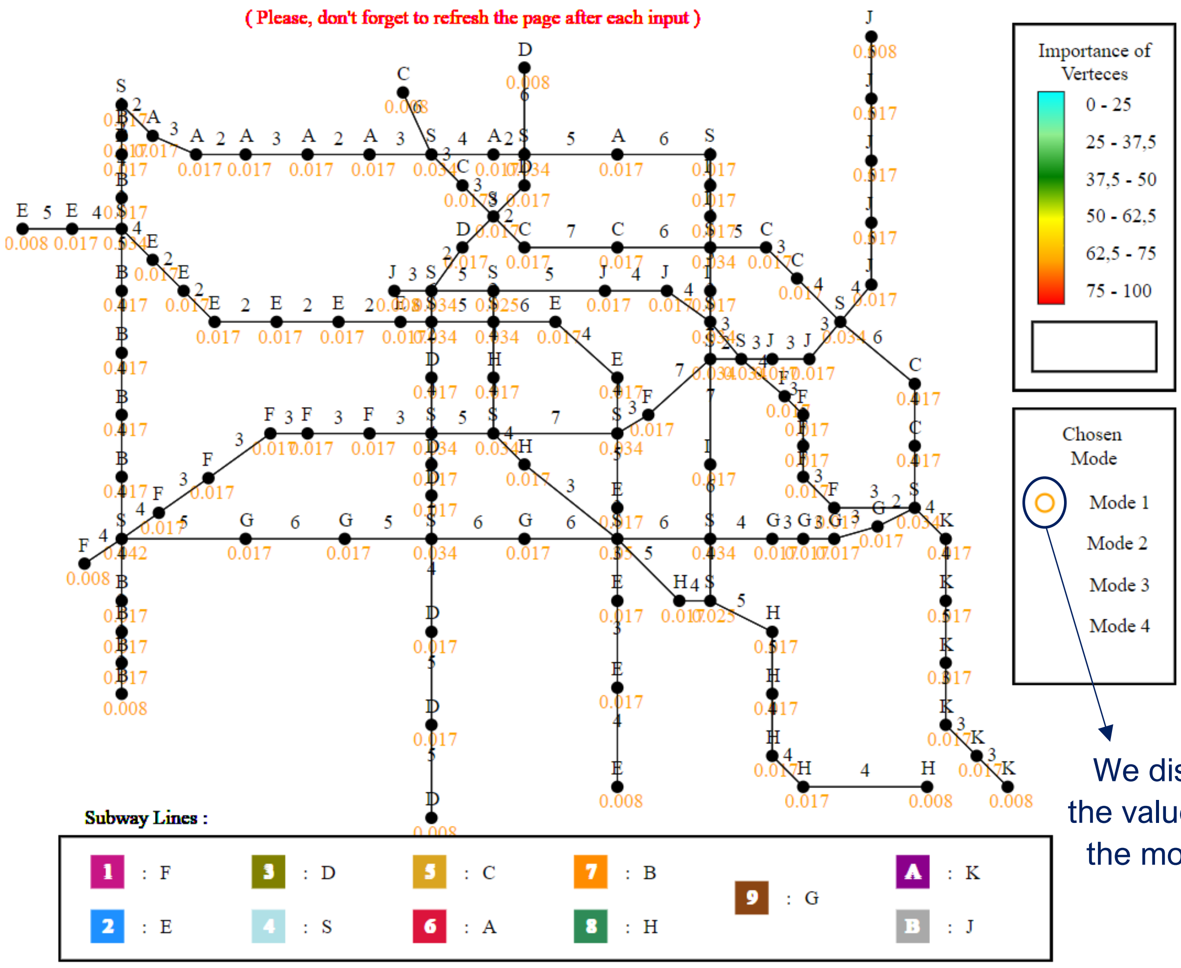
To display color only, we have to enter « 1 » (mode 1), « 2 » (mode 2), « 3 » (mode 3) or « 4 » (mode 4). For example, when we display colors of the mode 1 (when the input is « 1 »), the stations would be colored according to their degree centrality index.



We display the colors of the mode 1

We can see on this output that we display colors of mode 1. The vertices with one edge (the last stops of the lines) are represented in light blue while the vertices with two edges are represented in green. The vertices connected by more than two edges (the subway/train connections) are represented in warm colors from yellow to red. The more a vertex is connected by edges the more its color will tend towards red. Conversely, the less a vertex is connected by edges the more its color will tend towards blue.

To display values, we have to enter « V1 » (mode 1), « V2 » (mode 2), « V3 » (mode 3) or « V4 » (mode 4). For example, when we display values of the mode 1 (when the input is « V1 »), the degree centrality index of the stations would be displayed.

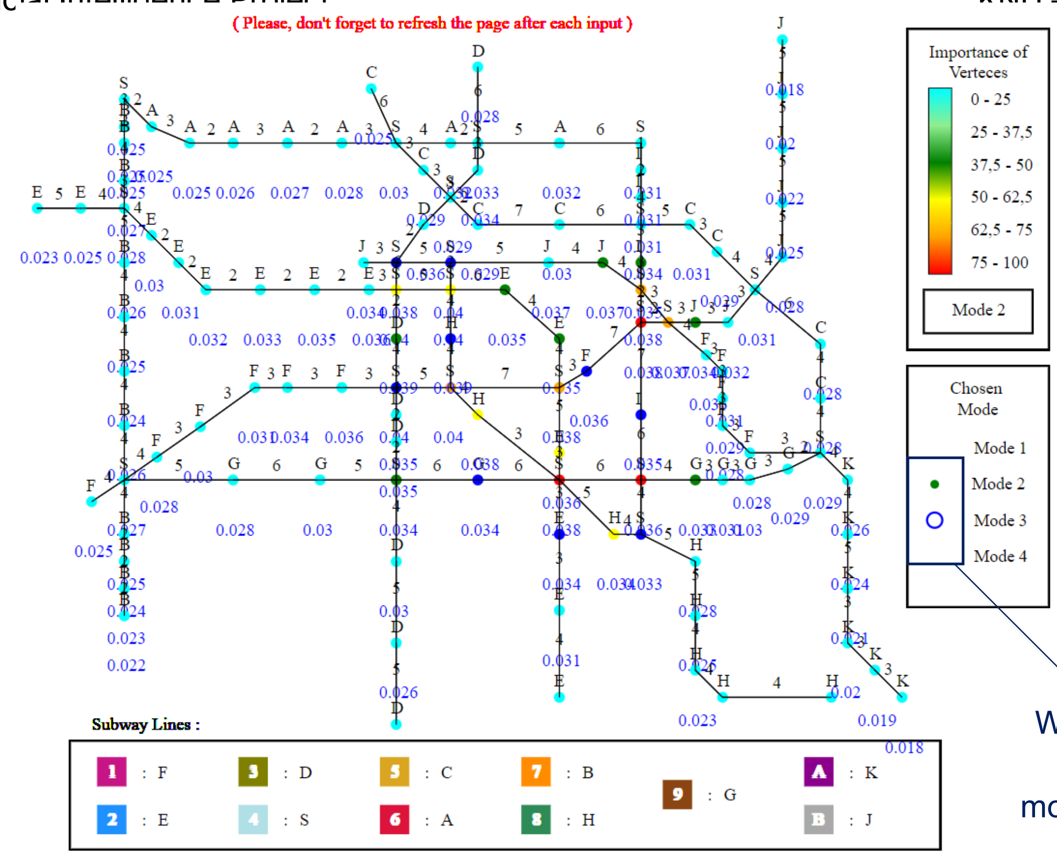


We display the values of the mode 1

We can see on this output that we display the values of the mode 1. The vertices with one edge (the last stops of the lines) have the lowest degree centrality index (0.008) while the vertices with two edges have a higher index (0.017). The vertices connected by more than two edges (the subway/train connections) have the highest index from 0.025 to 0.05 (three edges : 0.025, four edges : 0.034, five edges : 0.042, six edges : 0.05) The more a vertex is connected by edges the more its index will tend towards 0.05. Conversely, the less a vertex is connected by edges the more its index will tend towards 0.08.

To display values and colors at the same time, we have to enter « V1 », « V2 », « V3 » or « V4 » and then « 1 », « 2 », « 3 » or « 4 ». The reverse also works. We can also display the colors with one mode and the values with another.

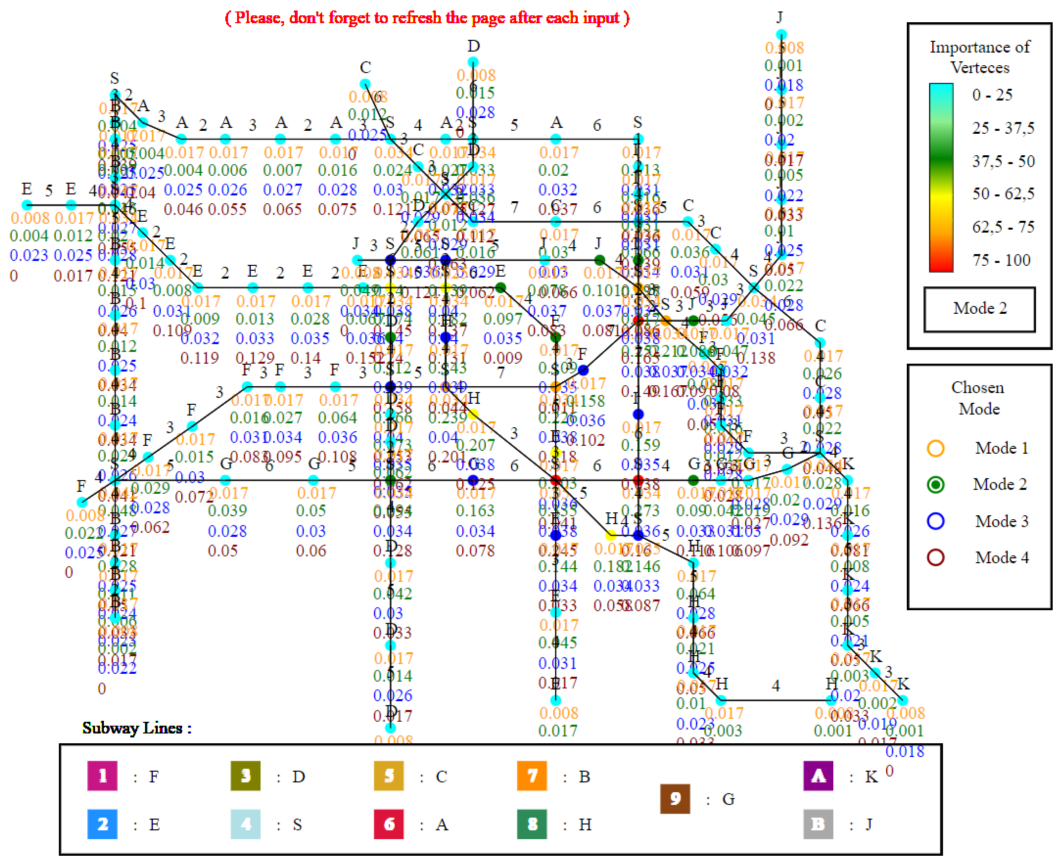
For example, we can display the values of the betweenness centrality indices and the colors corresponding to the eigenvector centrality indices.



We display the values of the mode 3 and the colors of the mode 2

Example of use of the program

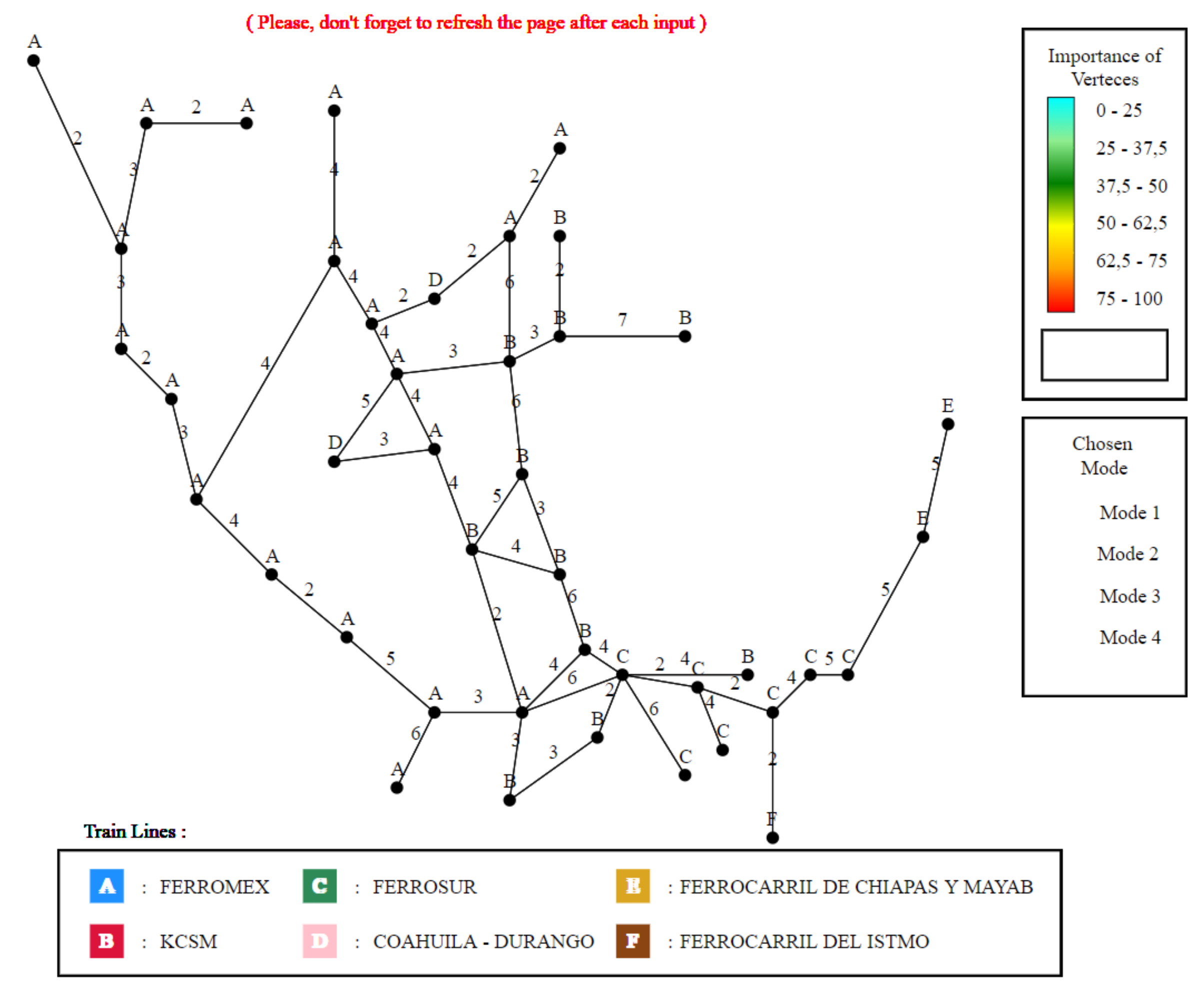
We can also display all the values of all the centrality indices at the same time by enter « V1 », « V2 », « V3 » and « V4 » in the console.



Example of use of our program where all the modes to display values are selected

To cancel the selection of a mode, we simply have to rewrite the choice of mode in the console. As for example, to cancel the display of values of mode 1, we have to rewrite « V1 » in the console. If we display colors of one mode we can not cancel it as soon as one of the modes is selected. We simply have to decide to display the colors of another mode in order to cancel the current display of colors of the mode selected.

In addition to representing the subway lines of Mexico City, we decided to represent the train lines of Mexico City to extend our application and to test the normalization of our calculations. To analyse the train lines, we have to select the command « P2 » in the list of commands in the console. To go back to the page with subway lines, we have to enter « P1 » in the console.



A	: FERROMEX
C	: FERROSUR
E	: FERROCARRIL DE CHIAPAS Y MAYAB
B	: KCSM
D	: COAHUILA - DURANGO
F	: FERROCARRIL DEL ISTMO

<u>The page of the output showing the train lines of Mexico City</u>

To quit the application, we can enter « exit » in the console.

Results

To evaluate our calculation methods, we need to analyze the results of our code. We will do this analysis with the Mexico City subway network.

1. Degree centrality

Degree of a vertex	*Normalized degree centrality index*
1	0.008
2	0.017
3	0.025
4	0.034
5	0.042
6	0.05

What we can see from these results is that the degree centrality index assesses the connectivity of stations. For a given station, it permits to know its directivity with other stations.

2. Eigenvector centrality

The eigenvector centrality index varies from 0.001 to 0.335. The vertex furthest from the center has an index of 0.001 while the central vertex has an index of 0.335.

What we can see from these results is that the further a station is from the city center, the lower its eigenvector centrality index is. The station with the index amounting to 0.335 can be considered as the central station of Mexico City. We note that it is also the station with the most neighboring stations (6).

3. Closeness Centrality

The closeness centrality index varies from 0.018 to 0.04. The most outlying vertex has an index of 0.017 while the most surrounding vertex has an index of 0.04.

What we can see from these results is that the more a station is outlying the lower its closeness centrality index is. This index permits to know the remoteness of a station. It differentiates the most remote and least surrounded stations from those with the most nearby stations.

4. Betweeness Centrality

The betweenness centrality index varies from 0 to 0.253. If a vertex is on the shortest paths connecting several other vertices together, then its betweenness centrality index will be high.

What we can see from these results is that the index assesses the potential of a station to be a gateway between two other stations. It permits to know the intermediate stations of an itinerary.

Conclusion

In conclusion, what we preferred about this project is that it gave us a lot of freedom in the interpretation of the instructions and the choice of the computer language. Coding in a familiar language a project applied to a less familiar domain was a great help for us. It allowed us to better understand the stakes of the implementation of an intelligent application. Nevertheless, the freedom offered by the instructions was also a challenge. Not following to the letter a set of specifications is certainly more stimulating and satisfying, but it is also a difficulty. We had to take risks, find our own solutions and innovate. Still, we enjoyed being challenged like this. We feel that we have improved our thinking and decision making skills; two key skills of a good engineer.

This project has a significant professional potential because it is concrete and complete. It permits the analysis of a transport network from all angles. Applied to AI, this project permits to obtain an intelligent itinerary taking into account the connectivity, the centrality, the remoteness and the crossing points of several stations.

However, there is still room for improvement in our code. We would like to extend it to other AI applications like a neural network, a social network or a pandemic network. The same calculations can be used for this kind of network. Their interpretation is different and this is where our interest lies.

Sources:

- Color gradient on SVG : https://la-cascade.io/les-degrades-svg/
- Rounding values : https://www.developpez.net/forums/d29018/c-cpp/cpp/arrondir-chiffre-2-chiffres-apres-virgule/