UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

Personalized Routes: Network Analysis and Shortest Path Algorithm

Author:
Luis RODRÍGUEZ BALLABRIGA

Supervisor:
Carlos CARRASCO FARRÉ
Dr. Daniel ORTIZ MARTÍNEZ

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

June 30, 2022

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Personalized Routes:
Network Analysis and Shortest Path Algorithm

by Luis RODRÍGUEZ BALLABRIGA

We spend our lives running from one place to another, always looking for the fastest way to go anywhere, but what if in that transfer, a priori insignificant, we could satisfy needs that, most likely, improve our mood?

In this project we present a personalized route creator, where from the user's needs and interests we create a route that passes through certain points of attraction. To do this, we will start from the world of networks, understanding their geographical nature and the correct application of the most common algorithms. Gradually, we will move from simple examples to more complex versions that will simulate real life. Once this knowledge is consolidated, we will begin to treat the data provided by OpenData from different cities and we will suggest different algorithms to solve the proposed problem. Based on the different results, we will choose the one that best suits our work environment. Finally, we will see where the developed algorithm could be implemented and the advantages it would bring.

Acknowledgements

First of all, I would like to thank Carlos Carrasco and Dr. Daniel Ortiz for their help in the development and approach of this project, as well as in the correction of the final report.

Secondly, I would like to thank my parents and my brother for always helping me with whatever I needed and guiding me on the beautiful path of life, giving me valuable advice and teaching me to enjoy it.

Next, I want to express my gratitude to Miquel and Xavier for all these years living together, listening and supporting us whenever we needed it.

Last but not least, thanks to you Maria, you know that if it were not for you I would not be delivering this project.

Thank you from the bottom of my heart, this is also yours.

Contents

Chapter 1

Introduction and Motivation

1.1 Motivation of the subject

We live in a society where everything happens very fast, in a continuous race against ourselves to, let's say, make the most of every moment. No one wants to feel that they are wasting time. But it is often in the small moments when we forget about everything or think only about our life that we enjoy it to the fullest.

This is why current routing systems fail in their attempt to improve this aspect as they only recommend the fastest routes, which are guaranteed to enhance the user's feeling of not dawdling. But are we always in such a hurry? Do we want to spend our lives rushing from one place to another without even thinking about where we are going?

With all of the above in mind and with the aim of making each route more than just a simple transfer, we propose a recommended route creator. In it, the path will not only take into account the optimization of distance, but also the user's interests. In this way, the route will be slightly modified to pass by local stores and businesses that can attract their attention and that, in some way, can make the walk a more pleasant moment to enjoy and disconnect from the rush. In addition to meeting other day-to-day needs, such as shopping for clothes, the pharmacy, bazaars or travel agencies.

This will benefit both people who want to continue to make the most of their time and those who simply want to let themselves go. The former because they will also be making the most of their time and satisfying other needs in a common transfer. The others because by letting themselves be carried away by their interests, they can see their happiness increase.

I, as the author, was unaware of many of the concepts that we will discuss throughout the project or simply was not conscious that they could be applied in this environment. Therefore, when I discovered the topic I knew that it could be a great opportunity to learn about this branch that I have not dealt with during the master's degree and that definitely catches my attention.

1.2 About personalized routes

Previous work on personalized routes has not taken into account the role of human interests in the urban context when recommending routes. They have not considered bringing recommenders together with network analysis.

In recent years, applications have emerged that attempt to enhance a user's enjoyment of a route. They use the concept of *psychogeography*. This is defined as the study of the precise laws and specific effects of the geographical environment, whether consciously organized or not, on the emotions and behavior of individuals (Debord, 1955). Among these applications are Random GPS[1], Derive app[2], or Drift[3].

Our work aims to deviate a little from all of the above and go further, to propose ways of automatically generating routes that are not only short but also interesting and profitable for the user.

1.3 Goals

The objectives of the project are as follows:

(i) Define network analysis from a formal and mathematical point of view and comprehend some of the main algorithms for path creation.

(ii) Understand the geographical approach to graphs, catching the structure of locations and metric spaces.

(iii) Relate the two previous points to the data analysis environment and learn how to deal with it all, starting from the basics.

(iv) Develop a logical algorithm to solve the proposed problem that can be applied to almost any location for which a dataset is available.

(v) Compare situations to assess the correct performance of the algorithm for different environments.

(vi) Think about the usefulness of the developed algorithm and where it might be interesting to use the implementation.

1.4 Structure of the memory

The work has a progressive structure in which no prior knowledge of the subject is assumed. All of them are introduced as they are required, accompanied by the necessary mathematical detail.

Chapter 2 provides an introduction to network analysis, where everything that will be covered later is put into context. It also serves to define different concepts that will be used during the project. Some of them are familiar to the reader, but it is useful to comment them in order to remember the details. All of this will allow us to situate ourselves at the starting point, with the necessary knowledge to be able to go deeper into the subject.

The next chapter, Chapter 3, deals with the problem on which the project is based: the creation of an algorithm to generate personalized routes according to the

[1] https://d-w.fr/
[2] https://deriveapp.com/
[3] http://www.brokencitylab.org/drift/

user's interests. All the processes carried out for the development of the algorithm are presented in an organized way, from the treatment of the used database, through the logic of the algorithm, showing the different versions considered, as well as the results.

Chapter 4 compares the different algorithms. The one with the best results is chosen and some extra analyses are carried out to get an idea of how it works.

Finally, in Chapter 5, the whole process is reviewed and commented on, as well as the objectives, checking whether all of them have been met. Also, future lines of work and environments where the algorithm could be of great use are discussed.

As can be seen, many experiments have been performed. Although, in this memory, we only provide the main results, all the code can be found on the GitHub repository `https://github.com/luisrodri97/TFM_Personalized_Routes`.

Chapter 2

Network Analysis

After a brief overview of the main themes of the project, this chapter will introduce the concepts necessary to put the starting point in place. We will talk about networks, Geographical Information Systems, and how to relate them. We will finish by talking about what can be done with it in programming, in our case in the Python language, going through the different libraries that make the processes easier and that can be very useful.

2.1 Network Theory

The main objective of this section[1] is to introduce some concepts to the reader to familiarise him/her with the working environment and to facilitate the understanding of what is to come.

2.1.1 Basics of Network Theory

A network is a collection of linked objects. We call them the nodes or vertices and usually represent them as points. The connections between these nodes are called edges and typically drawn as lines between points. Network theory corresponds to the study of these networks as a representation of relationships between the different objects. This theory corresponds to a sub-field of *Graph theory*, and that is because a network can be defined as a graph with some attributes on the points or the edges.

Mathematically,

Definition 2.1.1 *A **graph** $G = (V, E)$ is a set of elements $V = V(G)$, called **vertices**, and a set of **edges** $E = E(G)$ joining two vertices. If two or more edges join the same pair of vertices, they are called **multiple edges**. If one edges joins a vertex itself, we call it a **loop**. We call a graph **simple** if it doesn't have loops.*

Example 2.1.2 *Graph with 4 vertices and 6 edges: $V = \{a, b, c, d\}$, $E = \{1, 2, 3, 4, 5\}$ in Fig. 2.1.*

Definition 2.1.3 *Let G be a graph, a **subgraph** of G is a graph whose sets of vertices and edges belong to G.*

Definition 2.1.4 *Let G be a graph and $u, v \in V$. We define $d(u, v)$, distance from u to v, the length of the shortest path (called geodesic) joining these vertices. We call diameter of G the length of the longest geodesic of G, which is denoted as $D(G)$.*

[1] This section is based on notes taken in the class of the Graphs course of the Mathematics degree.

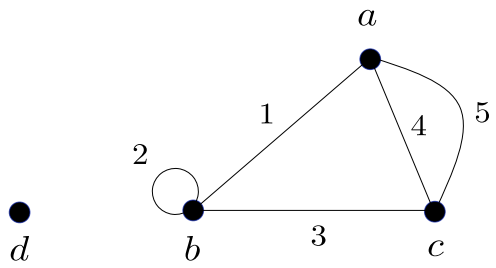

FIGURE 2.1: Graph corresponding to Example 2.1.2.

Example 2.1.5 *It is easy to see that d is a metric on the set of vertices V. In this example, $d(a,b) = d(a,c) = d(b,c) = d(c,d) = 1$; $d(a,d) = d(b,d) = 2$. The diameter of G is $D(G) = 2$, as can be seen in Fig. 2.2.*

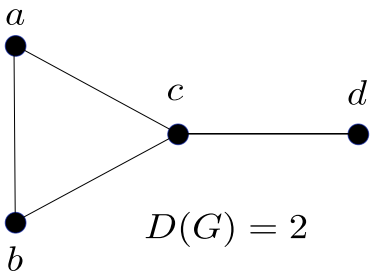

FIGURE 2.2: Representation of a graph with its metrics corresponding to Example 2.1.5.

When attempting to model systems such as those mentioned above, one quickly realizes that a simple network model with identical nodes and edges can not describe important features of real networks. One of the problems is that the edges of this simpler network model are undirected.

Definition 2.1.6 *A **directed graph** is normal graph $G = (V, E)$ where the set of edges E consists of ordered pairs, i.e., the edge joining the vertices A and B is not the same as the one joining the B and A.*

Another significant problem is that the edges are unweighted, and the path linking two nodes does not always have to have the same cost. For example, in the case where we want to consider distance between points, or the difficulty of the path.

Definition 2.1.7 *Let G be a connected simple graph. We say that (G, w) is a **weighted graph**, if $w : E(G) \longrightarrow R^+$ is a function, denoted as a weight.*

Remark 2.1.8 *For a given weighted graph (G, w), we can define the weighted length of a walk $c = u_1 u_2 \cdots u_k$, where $u_i \in G$ $1 \leq i \leq k$, as*

$$length_w(c) = \sum_{j=1}^{k-1} w(\overline{u_j u_{j+1}}).$$

Where $\overline{u_j u_{j+1}}$ represents the edge or path joining u_i and u_{i+1}. The distance $d_w(a,b)$ is defined as the least weighted length among all paths between the vertices a and b.

Another interesting way to represent graphs is by using what is known as adjacency matrix.

Definition 2.1.9 *Let G be a graph. The **adjacency matrix** of G is the $n \times n$ square matrix*

$$A(G) = (a_{i,j})_{i,j=1,\cdots,n} ,$$

where $a_{i,j}$ is the number of edges joining the vertices i, j. By consistency, loops count twice.

In the case we are dealing with a weighted graph, then the matrix is called **cost matrix** and $a_{i,j}$ will correspond to the weight of the edge joining i and j. We will assume that there aren't two different edges joining the same pair of vertices.

2.1.2 Dijkstra's algorithm

One problem we might be interested in solving is given a weighted graph, to calculate the minimum distance (usually the geodesic) between two arbitrary vertices. This is denoted as the optimal path between such vertices. For this purpose, the computer scientist Edsger W. Dijkstra published in 1959 the algorithm with its own name, Dijkstra's algorithm (Dijkstra, 1959).

Lemma 2.1.10 *Given a set of vertices $S \subset G$, and denoting by $\overline{S} := G \backslash S$. Let (G, w) be a weighted graph. It is easy to see that*

$$d_w(u, \overline{S}) = \min\{d_w(u, v) + w(\overline{vz}) : v \in S, z \in \overline{S}\}, \tag{2.1}$$

where $d_w(u, \overline{S}) := \min\{d_w(u, v) : v \in \overline{S}\}$.

With the above lemma in mind, we can now look at the algorithm.

Algorithm 2.1.11 *Let (G, w) be a weighted graph:*

***Step 1:** We choose $u \in V(G)$, and denote $u_1 := u$ and $S_1 := \{u_1\}$. Using Eq. (2.1) we can find $u_2 \in \overline{S_1}$ such that*

$$d_w(u, \overline{S}) = \min\{d_w(u_1, v) + w(\overline{vz}) : v \in S_1, z \in \overline{S_1}\} = w(\overline{u_1 u_2}).$$

We define $S_2 := S_1 \cup \{u_2\}$.

***Step 2:** Assume that we have found the following set of vertices:*

$$S_1 \subset \cdots \subset S_k$$

with $k < n$. Using 2.1 we find $u_{k+1} \in \overline{S_k}$ satisfying that

$$d_w(u_1, \overline{S_k}) = d_w(u_1, u_j) + w(\overline{u_j u_{k+1}}),$$

for some $j \in \{1, \cdots, k\}$. We denote by $S_{k+1} = S_k \cup \{u_{k+1}\}$.

The process stops if $k = n$, where we finally get $S_n = V(G)$.

Note that with the above algorithm we calculate the minimum distance from a given vertex to the rest, and, consequently we will obtain the distance between the two required vertices. Let's see what we have just explained with an example.

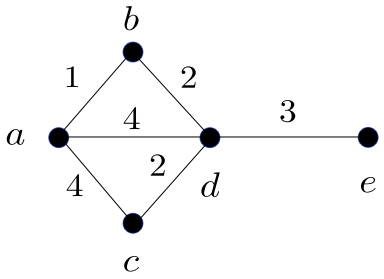

FIGURE 2.3: Complete graph for Example 2.1.12.

Example 2.1.12 *Given the following graph, Fig. 2.3, let us calculate the distances from the vertex a to all the others, showing also a geodesic on each case.*

We take $u_1 = a$ *and* $S_1 = \{u_1\}$. *On the first step, we have to choose the edge incident to a with the smallest weight, which is* $\overline{ab}$, *and denote* $u_2 = b$ *and* $S_2 = \{u_1, u_2\} = \{a, b\}$, *Fig. 2.4.*

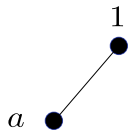

FIGURE 2.4: Result of Dijsktra's first step, Example 2.1.12.

On the second step, we analyze all edges incident to a (there are two, both with weight equals 4), and also incident to b (adding up the distance between a and b, which is 1), and we select the smallest distance, which in this case equals 3, corresponding to the edge $\overline{bd}$. *We write* $S_3 = \{a, b, d\}$, *Fig. 2.5.*

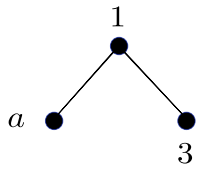

FIGURE 2.5: Result of Dijktra's second step, Example 2.1.12.

At this moment, we still have an edge ($\overline{ac}$) *incident to a and with weight equals 4, from b we have no more options (all adjacents to b have been already chosen), and from d we can get a 6 (adding* $\overline{de}$) *or a 5 (adding* $\overline{dc}$). *Hence, we select the first option from a and set* $S_4 = \{a, b, d, c\}$, *Fig. 2.6.*

Finally, there is only an edge left $\overline{de}$, *and we get* $S_5 = \{a, b, d, c, e\}$, *Fig. 2.7.*

Thus, the distances we obtain are the following:

$$d_w(a,b) = 1,\ d_w(a,c) = 4,\ d_w(a,d) = 3,\ d_w(a,e) = 5.$$

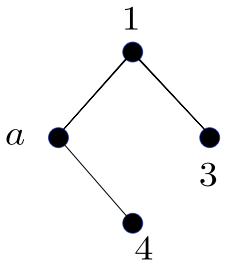

FIGURE 2.6: Result of Dijkstra's third step Example 2.1.12.

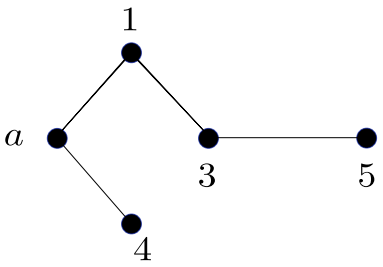

FIGURE 2.7: Final result of the algorithm, Example 2.1.12.

2.1.3 Travelling Salesman Problem

Related to the previous algorithm, there is a more complex problem that consists in finding the shortest path between two vertices passing through all the others. This one is called the Travelling Salesman Problem (TSP). The statement is the next one:

Statement 1 *Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the initial point.*

In order to address this problem we must define some new concepts:

Definition 2.1.13 *We say a graph is a **complete graph** if every pair of vertices are neighbours.*

Definition 2.1.14 *A simple graph G is called **Hamiltonian** if it possesses a closed path without repetition of edges or vertices that contains all vertices of G. This cycle is denoted as **Hamiltonian Tour**.*

So, now that we know these terms, we can see what the TSP is all about. It consists in finding a Hamiltonian tour, on a weighted complete graph, having the smallest possible weight.

This problem is conceptually simple but mathematically complex and challenging. It is a question for which no algorithm has been found to solve it efficiently.

2.2 Geographical Network Analysis

Before talking about Geographical Network Analysis is necessary to introduce the concept of Geographic Information Systems (GIS). GIS is a digital computer environment used to create, capture, analyze and visualize data and its spatial equivalent, its geographic information (Jensen and Jensen, 2013).

Both Network Analysis and GIS are closely related and growing fields. And they together can become a very powerful computational tool. For doing this kind of analysis we have to define the concept of a spatial network.

Definition 2.2.1 *A spatial network corresponds to a graph* (V, E) *where the nodes* V *are embedded in some metric space.*

From this definition, it follows that a distance function can be applied between two nodes in this network. Usually, this distance is a geographic one.

All this can be used in many different contexts and spaces, such as social media or demographic analysis. In this project we will focus on cartographic data, mainly working on routing algorithms, as anticipated in the previous sections. The element that distinguishes cartographic information from all other types of information is the location. Everything has a location in space-time, and this feature can be used specifically to make better predictions and examinations. Without it, data is non-spatial and would have little value. Location is therefore the basis for many of the advantages of GIS: the ability to map, the ability to measure distances, and the ability to relate different types of information since they connect with the same place (Rey, Arribas-Bel, and Wolf, 2020). For it, an arbitrary coordinate system is used.

To illustrate the above, we have to talk about geographic processes (Fischer, 1959). These are represented by three different categories:

- **Objects:** Things that take a position in time and space, for example, buildings.
- **Fields:** Something that can be measured at any location in space and time, like distance.
- **Networks:** Connections between objects and fields.

Understanding the differences between these three representations is crucial because they influence the types of interactions. Gross distances may not be a valid indicator of their underlying geographical relationship, given their interconnectivity. We cannot assume that every node is connected, and we cannot infer these relationships only from the nodes. Two subway stations, for example, could be quite far apart but be connected by a train.

These kinds of structures challenge us on how processes are conceptualized. It is useful to see that how a geographical process operates can be different from how we can measure it.

A network can represent possible routes from one location to another. There exist human-made and natural networks. Cars, subways, trains, airplanes, and boats, all move on pre-determined networks. We also walk and ride bicycles in known networks and even stuff like water and electricity does.

We, as humans, are interested in the movement of goods, services, and resources. And we are even more interested in doing it in the optimum way, i.e, finding the optimum path through the network. Therefore, it is not surprising that a lot of research has gone into developing GIS functions that can be used to analyze the spatial characteristics of networks to address this important problem.

2.3 Geographical Network Analysis in Python

In this section, we will build the ideas discussed during this chapter in a practical context. There are many ways of doing Network Analysis. We have chosen to make it in Python as it is a familiar language for the author and because of the number of existing open-source libraries that will help us to deal with graphs. We are going to cover how to implement data structures and, also, the data models. As well as interacting with them.

During this block we will be using datasets and information related to the city of Barcelona[2], it will will be used as a toy problem to see all the features and procedures. The experiments and results performed in this section can be found in the notebook `network_analysis_experiments.ipynb`.

2.3.1 Vector Data and Shapefiles

Vector data stands for discrete geometric locations, composed of x and y values, which are known as coordinates[3]. These define the shape of the different items. There are three types of object, represented in Fig. 2.8:

(i) **Points:** Individual coordinates, composed by single values of x and y. They usually represent just locations.

(ii) **Lines:** Objects composed by two or more vertices that are connected. They are used to represent paths, for example roads.

(iii) **Polygons:** Consists of three or more points that are connected and form a closed path. They are used to set boundaries, for example countries, buildings or lakes.

This vector data representing geospatial elements can be stored and treated in different ways. The first is in the so-called shapefiles and the second is in spatial graphs.

The format called `shapefile` is a specific way for storing this kind of data and deal with GIS software (Wasser et al., 2018). It can easily detail the vector features described above and, often, include more attributes, for instance, texture for points, the street name for lines or neighborhood for polygons.

This structure consists of a collection of files with a common filename that needs to be stored in the same directory. Each of these files contains only one of the vector types. You will never find all the different objects in a single file.

[2]The files have been downloaded from Open Street Map API.
[3]We will consider this coordinates the vertices of our graphs.

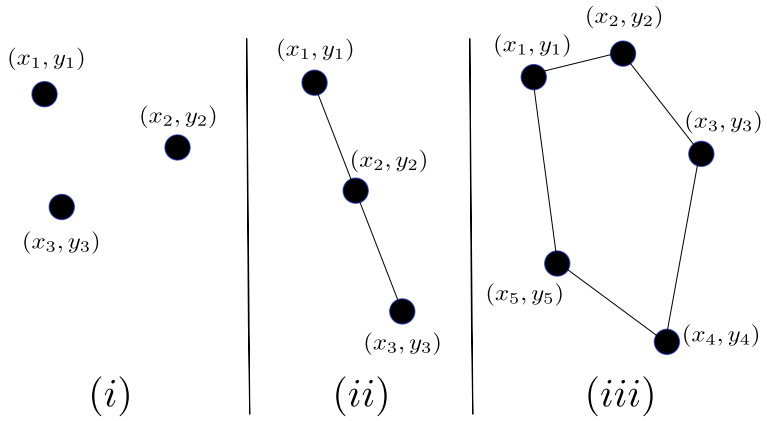

FIGURE 2.8: An electron (artist's impression).

The organization is the following one. First, we have the three mandatory extensions:

- `.shp`: Contains the geometry for the features.
- `.shx`: Contains the positional index of the feature geometry.
- `.dbf`: Stores the attributes for each shape.

After this ones, we can have other files like:

- `.prj`: Contains the project description.
- `.sbn`: Contains a spatial index of the feature geometry.
- `.shp.xml`: Contains geospatial metadata in `XLM` format.
- ...

In our case, we have two different packs of shapefiles, the first one corresponds to the different points and coordinates in the city, as it can be seen in Tab. 2.1. The second refers to streets and is therefore composed of lines. We can see in Tab. 2.2 that it contains some attributes related to each edge. For example, the distance (*Longitud*) of the path or District it belongs (*NDistric_D*), among others. In this file, we have a total of 24 different features.

C_Nus	Coord_X	Coord_Y	Geometry
N00001D	432,202.434	4,581,246.437	POINT(432202.434 4581246.437)
N00002E	432,243.589	4,581,257.498	POINT(432243.589 4581257.498)
N00003F	432,257.945	4,581,261.119	POINT(432257.945 4581261.119)

TABLE 2.1: Points dataset corresponding to shapefiles *BCN_GrafVial_Nodes_ED50_SHP*.

We can now plot the two datasets in the same figure, so the different points and edges overlap. We can use some of the features, for example, the different Districts in Fig. 2.9.

Coord_X	Coord_Y	Longitud	...	NDistrict_D	...	Geometry
432,223.0115	4,581,252	42.615487	...	Ciutat Vella	...	LINESTRING(...)
432,250.7670	4,581,259	14.805620	...	Ciutat Vella	...	LINESTRING(...)
432,265.2895	4,581,264	15.480301	...	Ciutat Vella	...	LINESTRING(...)

TABLE 2.2: Edges dataset corresponding to shapefiles *BCN_GrafVial_Trams_ED50_SHP*.

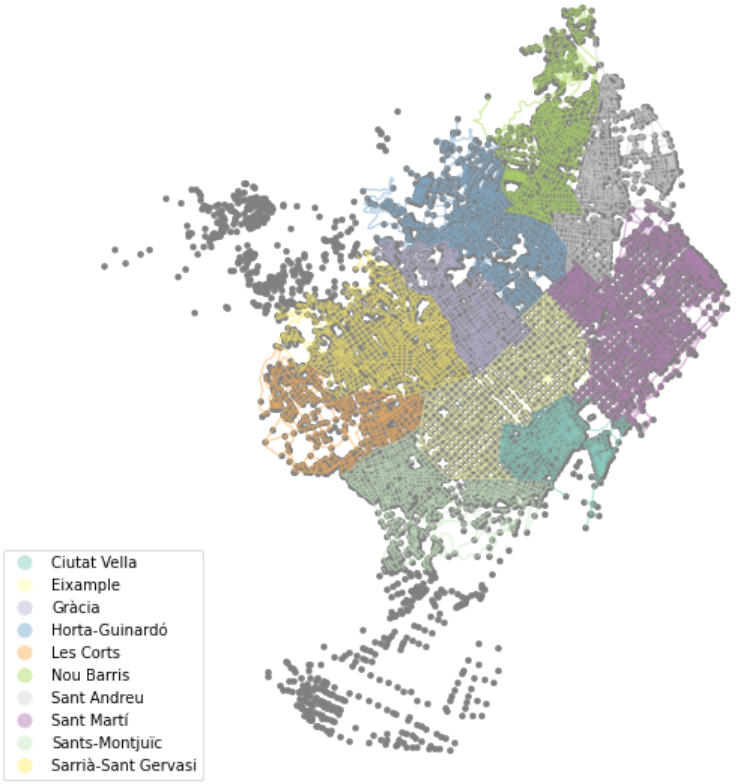

FIGURE 2.9: Plot of the two datasets. Points dataset in grey and Edges dataset in *Set2* colour range in accordance with the District.

From the datasets we can also obtain the spatial metadata of the geographic zone[4]. One to highlight is called Coordinate Reference System (CRS), which provide a standardized way of describing locations. In our case, we can see that both shapefiles have the same CRS information, which is the following:

```
<Projected CRS: EPSG:23031>
Name: ED50 / UTM zone 31N
Axis Info [cartesian]:
    - E[east]: Easting (metre)
    - N[north]: Northing (metre)
Area of Use:
    - name:
```

[4]All the different attributes and features that can be obtained using the documentation of the *Geopandas* library.

```
            Europe - between 0°E and 6°E - Andorra;
            Denmark (North Sea);
            Germany offshore;
            Netherlands offshore;
            Norway including Svalbard - onshore and offshore;
            Spain - onshore (mainland and Balearic Islands);
            United Kingdom (UKCS) offshore.
        - bounds: (0.0, 38.56, 6.01, 82.45)
    Coordinate Operation:
        - name: UTM zone 31N
    - method: Transverse Mercator
    Datum: European Datum 1950
    - Ellipsoid: International 1924
    - Prime Meridian: Greenwich
```

The second way of obtaining the geographic information is using the spatial graphs. These store the connections between the different objects. The structure they follow is quite different to the shapefile format, as these graphs do not record measurements. For using it, we will trust the `osmnx`[5] library (Boeing, 2017), which will allow us to query data from Open Street Map and help us dealing with all the data. In this process we can even specify what kind of roads we want to contain in our graph, whether walking paths, roads or even bike lanes. Depending on these options the graph will be addressed or not and will have different attributes.

The resultant graph of this process is a `MultiDiGraph` from the `networkx`, which is a commonly used format in graph treatments in Python. This result can easily be plotted, Fig. 2.10. We can also obtain characteristics and statistics about the graph, such as the number of nodes, edges, or the nodes related to one of them. In Tab. 2.3 we can see some of them. We can even obtain more advanced information such as centrality, clustering coefficient, or the mean eccentricity of the network. It is important to note that for each of the objects that make up the network there is a unique identifier that will allow us to work comfortably, it is called `osmid`.

```
{'n': 68989,
 'm': 222298,
 'k_avg': 6.444447665569873,
 'edge_length_total': 7874134.064000145,
 'edge_length_avg': 35.42152454812974,
 'streets_per_node_avg': 3.2283697401034948,
 'streets_per_node_proportions': {0: 0.0, 1: 0.06067633970633, ...},
 'intersection_count': 64803,
 'street_length_total': 3937067.031999984,
 'street_segment_count': 111149,
 'street_length_avg': 35.42152454812894,
 'circuity_avg': 1.0496045685261224,
 'self_loop_proportion': 0.0006747699034629191}
```

TABLE 2.3: Some statistics about our graph.

[5]The library *osmnx* corresponds to a combination of the libraries *OpenStreetMap* and *Networkx*.

During this project we will be mainly using the second of the options to take advantage of the benefits that Python provides us with these libraries.

FIGURE 2.10: Plot of the graph using the *osmnx* library.

2.3.2 Shortest Path Algorithm

As we have seen, finding the shortest path is a common GIS problem (Tenkanen, 2017). In this section we will performn an analysis of finding the shortest walking route between two different points based on the information we have obtained previously.

For this purpose, we will be using the function `shortest_path()` from `networkx`. The algorithm in this function used to obtain the final path is the Dijkstra one. In it we can specify what we want to optimize, and, depending on our desire the weights of the graph will change. We will choose the length feature, as it seems the logical one when trying to find the shortest path. Perhaps we could choose the time it takes to travel, but since this time can vary depending on the person who travels it, we have preferred to choose the distance, which is a fixed measure[6].

First we need to specify the starting and destination locations of our route. The initial point will be the Faculty of Physics and Chemistry of the University of Barcelona and the final point will be the Faculty of Mathematics and Computer Science. Checking online we can obtain the coordinates of both places, as shown in Tab. 2.4.

Applying the algorithm we obtain a list of all the nodes that form the shortest path. This, together with the minimum weight axes linking the vertices, can be plotted as shown in Fig. 2.11.

[6]If we were minimizing car routes it would make sense to do it with the time feature, using the speeds of the different roads.

Place	Latitude	Longitude
Faculty of Physics	41.385327	2.116779
Faculty of Mathematics	41.386310	2.164115

TABLE 2.4: Coordinates for both places obtained from Google Maps.

FIGURE 2.11: Route obtained using *shortest_path* function.

This route is 4,549.961 meters long. Comparing it to the one obtained using the platform Google Maps, Fig. 2.12, we can see that it is almost the same, with a minor difference in the last sector, where the Google's one takes a much straight forward path.

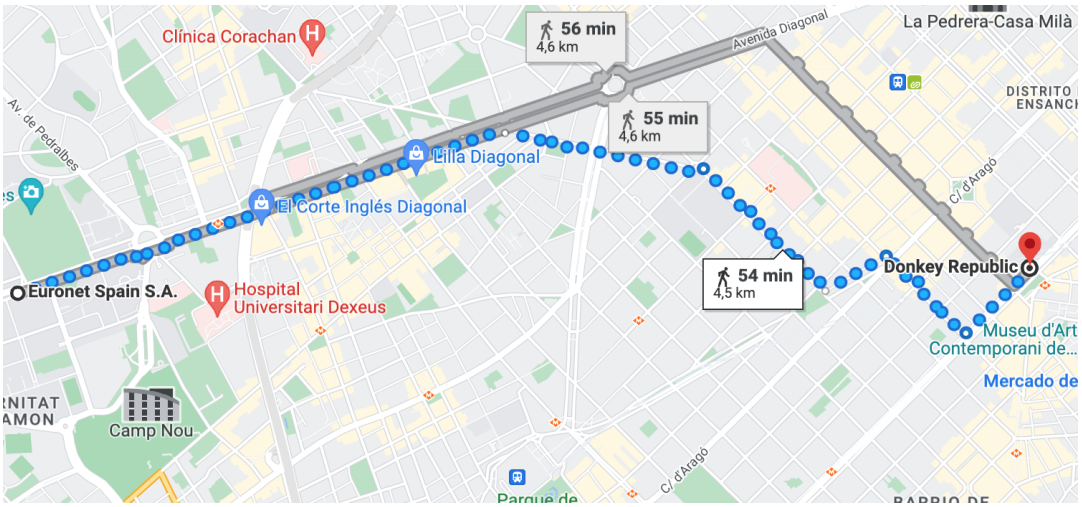

FIGURE 2.12: Route obtained using Google Maps.

If we check the distance of this last one, we can see it is 4,600 meters long. So, we get two routes that are very similar but not the same. After researching documentation on Google's algorithm (Mehta, Kanani, and Lande, 2019), we can come to the conclusion that the differences are due to the fact that Google also takes some other factors into account. For example, it takes into account the average time it takes users to go through the segments, or it takes into account that the fewer times

you have to change direction, the easier it is to follow the route. When there are several very similar routes, as in our case, these features become more important and relevant in the result.

Nevertheless, the distance to be covered is practically the same, so we can confirm that we have obtained the shortest route connecting the two points.

Chapter 3

Personalized routes

During this chapter we aim to present the project we will be working on, personalized routes. First, we are going to show what is currently available in the literature. We will ntroduce the datasets that will define the problem and we are going to comment on the treatment that has been carried out as well as the different solution proposals that have been worked on.

3.1 Related work

Trying to maximize the enjoyment of the citizens through urban routes is not something new. A great deal of work has been done by researchers trying to find algorithms to achieve this goal. All efforts so far to improve happiness in this area have focused on using historical buildings, parks, or other urban assets in terms of contribution to attractiveness or on simply finding the most efficient route.

Regarding this last option, Chang et al., 2011 used a backtracking algorithm to recommend car routes that deviate from user's past trajectories. Also Ludwig, Zenker, and Schrader, 2009 used the A*-algorithm to recommend routes for public transports optimizing short walks and little waiting times.

On the side of interesting or different urban routes, the main idea is to use georeferenced online content as for example *Flickr* images. Choudhury et al., 2010 identified movement from users by tracking the photos they upload to this website and the generating trajectories between points of interest in a graph.

More recently, thanks to the emergence and common use of social networks, this branch has been able to be expanded. Cheng et al., 2011 annotated historical travel data with demographic information and used a Bayesian learning model to generate personalized travel recommendations based on demographic segmentation. Kurashima, 2010 tackled a somewhat similar problem: they profiled users based on their travel history. These approaches produce sequences of locations according to different criteria, but do not focus on the nature of the paths connecting those locations. Also Quercia, Schifanella, and Aiello, 2014 developed an algorithm to create routes that are not only short but also emotionally pleasant, by collecting reliable perceptions of urban scenes.

However, none of them takes urban retail as focus of their development, despite consumption patterns are becoming clearer and are part of urban growth (Clark, Lloyd, and Wong, 2002). Much of the vitality of cities is linked to commerce, not only in terms of the sale of goods and services, but also in the social and city-building aspects (Ezmale and Litavniece, 2012). So we set out to do this kind of work taking

all this into account, incorporating it into the algorithmic solutions, and trying to explore what they bring us.

3.2 Proposed problem

Our proposal in this project is a variation of the minimum route algorithm discussed above. The aim is to create the shortest route connecting two points while taking into account the user's interests.

To do this, given a database of shops and businesses in a city, the shortest route must be found that connects an origin and a destination but goes through certain points of interest previously defined by the user. We are going to approach the problem in an experimental and explanatory point of view.

3.2.1 Datasets

This section will describe the datasets used and explain some ideas that need to be reviewed before we start tackling the problem. We will also describe the cleaning and processing of the data.

For this project we will be using data from *EIXOS Economic Observatory* from the cities of Barcelona, Madrid, London and New York. All the treatments are detailed and can be replicated for all the cities in the notebook `dataset_preprocessing.ipynb`.

Dimensionality

The used dataset contains the commercial registration for each of the cities. For every business, we have different attributes such as the location (using both coordinates and street and number), the date of the register, and a category on the type of business or establishment, among others. In Tab. 3.1 we show the volume of retail shops for the different cities.

City	# Registers	# Attributes
Barcelona	49,733	55
Madrid	62,190	13
London	50,822	13
New York	30,192	13

TABLE 3.1: Shape of the datasets for the differents cities.

Focusing now on the variable that refers to the category, `layer_id`, we can see the types of businesses we have in each city and which are the most abundant. This variable takes 154 different numerical values. To be able to understand it, we must use an extra dataset, *layers.xlsx*, where it is linked to its corresponding categorical value. In Tab. 3.2 we can see some of the categories and the number of registers for each one in the city of Barcelona.

Category	# Registers
local buit o tancat	7,287
Bar de tapes	3,703
Roba i complements	2,708
...	...
Indústria fusta i/o paper	1
Activitats primàries	1

TABLE 3.2: Volume of registers for each category.

Processing

Many of the categories available in the dataset are not useful for the problem we are considering. This is due to the lack of registers or given that they are of no interest to an average citizen. For example, paper, wood or textile industry. Consequently, it would make no sense to recommend a route through these establishments.

Therefore, to reduce the number of categories, we will carry out the following treatment based on the dataset of the city of Barcelona:

- First, we will choose those businesses with more than 100 records. This is done in order to guarantee that a minimum number of establishments can appear in the routes that will be proposed later.
- Second, by means of expert criteria, those categories that create interest among citizens are selected and grouped.

With the initial filter we reduce the number of categories to 69. After the second filter, we end with 26 different categories, which will be the ones that the user will be able to choose to generate personalized routes based on his/her interests.

We show in Tab. 3.3 the final labels and in Fig. 3.1 we can compare their volumes.

Index	Category	Index	Category
0	Property for sale or rent	13	Jewellery
1	Restaurants	14	Fast food
2	Hairdresser and beauty salon	15	Shoe shop
3	Clothes shop	16	Hotels
4	Supermarket	17	Real state agency
5	Household items	18	Art and Culture
6	Banks and financial services	19	Perfume shop
7	Bakery	20	Bazaar
8	Coffee shop	21	Car and motorbikes Dealership
9	Mechanical workshop	22	Travel agencies
10	Copy shop	23	Sports shop
11	Pharmacy	24	Optician's shop
12	Fruit shop	25	Florist

TABLE 3.3: List of final labels that will be used as recommendation options.

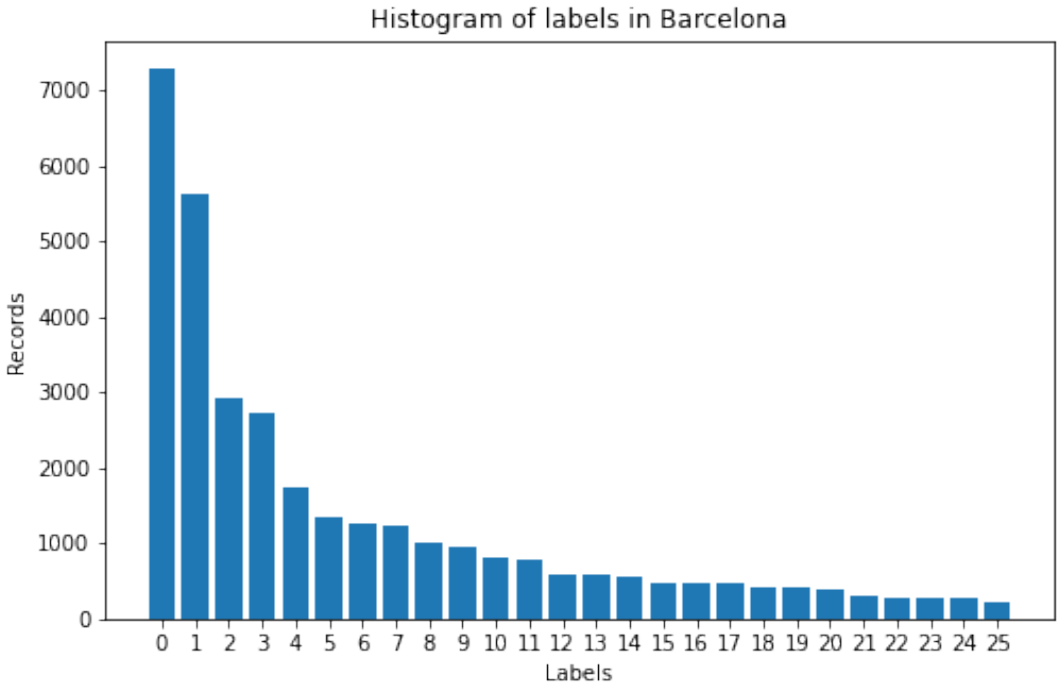

FIGURE 3.1: Histogram of the different categories by index in the city of Barcelona.

3.2.2 Algorithm

In this section, we are going to describe the developed algorithm to solve the problem proposed at the beginning of this section. All the detailed code and results shown can be found in the notebook `route_algorithm.ipynb`.

As explained in Sec. 2.3.2, we already know how to find the shortest path between two locations. The main idea of the algorithm is to add stops to this previous route so that the user goes through places of his/her interest. To do this, we need to know which of all the places do not deviate too much from the proposed route, so that the distance travelled is not drastically increased.

To achieve this, we need to relate each place of interest to the different nodes of the city to which it is close. A tolerance of 50 units in the values of the *longitude* and the *latitude* is established with respect to the coordinates, so that all those establishments that are in the area determined by $x \pm 50$ and $y \pm 50$, where (x, y) are the coordinates of node N, will be related to it, as can be seen in Fig. 3.2. Other options were considered, such as defining this area as a percentage, but then the value of this area depended on the position in space and was not the same for all points in the city: the further north, the higher the dimension of the defined square, for example. This choice was made in order to have fixed dimensions.

In order to take this into account, we generate the file `<city>_relations.csv`. In it we have, for each point forming the city, a list of all related businesses. To create it, dictionaries have been used to speed up the process and to work efficiently.

Now we can obtain a list of all the points through which the initial route passes and, relating it to the generated dataset, obtain all the locals that are close to the route

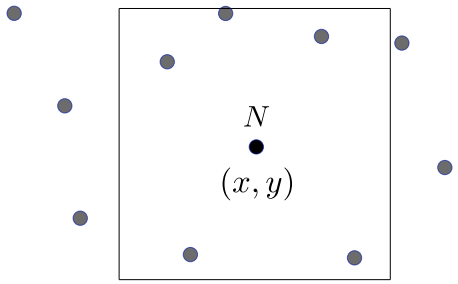

FIGURE 3.2: Area defined by a point in the city around it by the set tolerance. The blue dots represent points of interest that may fall inside or outside this area and, consequently, may or may not be considered.

(Fig. 3.3). We can then filter the user's interests to keep only those establishments that might catch his or her attention. We considered doing this filter at the beginning of the process, but it would have implied having a dataset of relationships for each of the categories to be recommended and for each of the cities, which was not feasible. It was noted that, although it may appear slower when using all available premises, the implemented process is not affected at all. Therefore, it was decided to go ahead with this version.

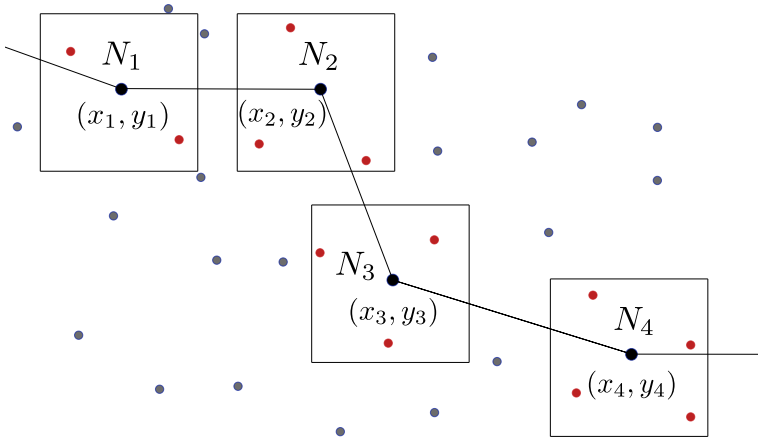

FIGURE 3.3: Example of a route, where we can see the nodes to follow, as well as the area that delimits which shops can be considered to generate the customised route. Those that satisfy the condition are marked in red.

At this point, a wide range of options opens up for the creation of the final route. Normally, the number of nearby establishments is very high, so it does not make sense to add all of them to the route. That is why it seems reasonable to choose a reduced number of them to generate the final route. Different ways have been tested, all of them adding a maximum of 5 businesses. In the following two sections, we explain the procedures and illustrate them with an example. In case the number of available establishments is less than or equal to 5, the route passing through all of them will be created.

Random Selection

This first option consists of randomly choosing 5 locals from the list to create the final route.

To do this we will use a list of the establishments ordered according to when they appear on the route and we will choose 5 according to their position. In this way we will ensure that we will pass by them in order of their appearance on the route and avoid making the user go back and forth, and therefore waste time. With this method, the routes generated between a start and end point do not always need be the same, nor will the establishments be the same.

Let us now look at the results of applying this process to the example (Fig. 3.4) shown in Sec. 2.3.2, using as source the Faculty of Physics and Chemistry and as target the Faculty of Mathematics and Computer Science and setting as user interest the purchase and/or rental of housing, i.e. "Real State Agency".

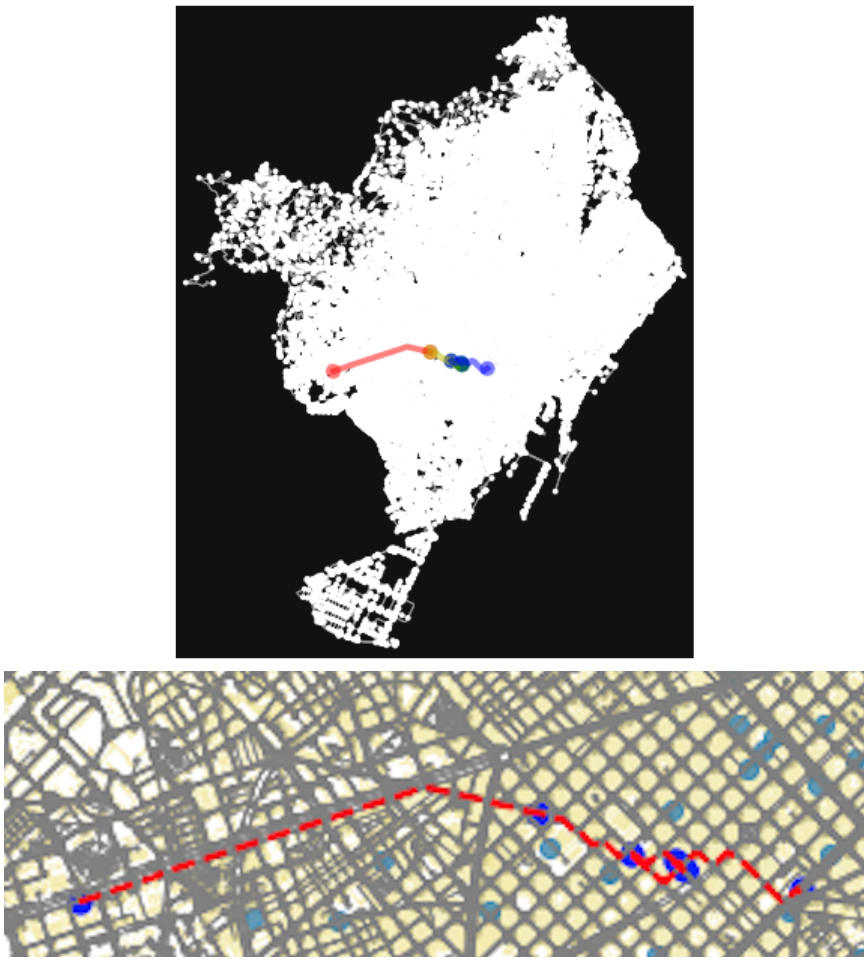

FIGURE 3.4: Route obtained with the Random Selection method. Above, the route with different colours after each stop. Below a more detailed view of same route, with the stops in blue and the path in red.

In this case the total length of the route is 5,401.682 meters long.

Shortest route Selection

Let us now look at the second option. In this one, instead of randomly choosing the stops, we choose those 5 (if there are so many) that minimise the final path. This process, a priori, seems to be more computationally expensive than the previous one, because all the possible routes must be calculated to find the minimum one.

In this case it would not be necessary to maintain the order we discussed in the previous case, but to reduce the number of route to be computed, only the routes where the establishments appeared in the right order have been calculated.

Let us look at the results for the same example as in the previous section (Fig. 3.5).

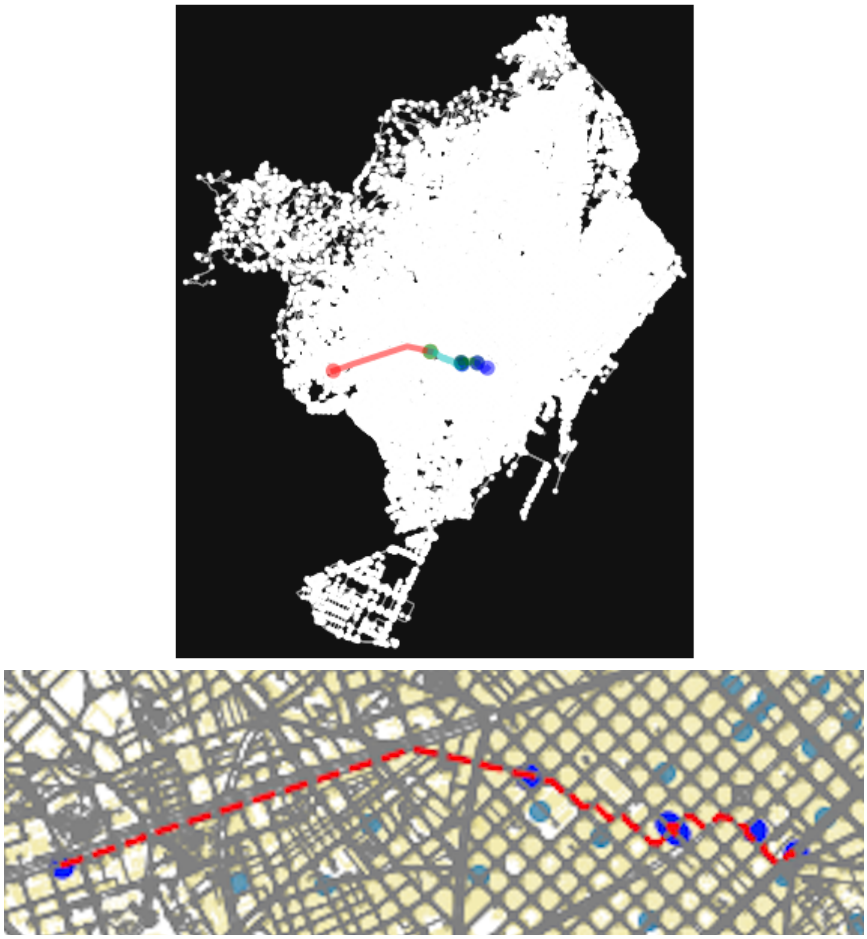

FIGURE 3.5: Route obtained with the Shortest route Selection method. Above, the route with different colours after each stop. Below a more detailed view of same route, with the stops in blue and the path in red.

The total length of the shown route is 4,679.905 meters long.

Chapter 4

Results

In this chapter, we are going to discuss the obtained results with the algorithms developed in the previous section and perform other experiments that may be of interest.

4.1 Algorithm selection

Looking at the result of the first option, Random selection, we can see that the new final distance is increased by almost 1 kilometer compared to the minimum distance calculated in Sec. 2.3.2. To see how much this distance usually increases we have run the algorithm 1,000 times. The average was 7,343.597 meters, with a minimum of 4,732.425 meters and a maximum of 9,893.232 meters. So the minimum distance is increased by approximately 63% with this algorithm.

Another analysis consisted of randomly taking 100 pairs of points in the city of Barcelona and, applying the same user interest to them, checking how the distances increased. In this case, the average distance to the points of interest was 5,763.278 meters, with a minimum of 923.447 meters and a maximum of 10,254.587 meters. The average increase is approximately 41%.

For the second option, Shortest route Selection, we see that the distance is increased by only 130 meters, or 1%. Performing the same experiment as for the first option with the same 100 pairs of points, we can conclude that with this version of the algorithm the routes are slightly affected, as they are only increased in length by 5%.

The main problem with this version of the algorithm is the computational cost. In order to calculate the minimum route, all possible routes must be calculated. This number increases as the number of available establishments increases. Specifically, let k be the number of possible stores on the route,

$$\#\text{ of possible routes} = \binom{k}{5} = \frac{k!}{5! \cdot (k-5)!}.$$

This can translate into hours of waiting for a single route. For example, if we have a choice of 30 locations, $\binom{30}{5} = 142,506$ different routes. Considering that the whole process to get a route takes about one second, we would have to wait 39 hours to obtain it, which is not feasible for a user.

This is why the first version of the algorithm is considered for use. It should also be taken into account that the fact of obtaining different routes each time can be a

plus, given that many people repeat the same routes every day, to get to and from work, school, etc. Being able to visit different places every day, despite maintaining the same interest, will give it value and increase its use.

Here we present a table, Tab. 4.1 with the summary of all the results presented in this section.

	Avg. Inc. 1K (%)	Avg. Inc. 100 (%)	Comp. time (s)
Random Selection	63	41	0.673
Shortest Route Selection	1	5	9.524e3

TABLE 4.1: Summary of the results shown in this section. *Avg. Inc. 1K:* Average increase for the example route over the minimum distance after 1,000 times. *Avg. Inc. 100:* Average increase over the minimum route taking 100 different routes over the city. *Comp. time:* Computation time.

4.2 Extra experiments

To conclude this section of the results and having chosen between the candidates, let's make a final analysis. In this we are going to take the route that we have been analysing during the project and we are going to apply the different interests to observe if any of them is more affected than the rest by the algorithm.

To do this, we will generate 100 different routes for each interest and calculate the average increment with respect to the minimum route. The results are shown in Tab. 4.2.

Category	**Incr.%**	**Category**	**Incr.%**
Property for sale or rent	63%	Jewellery	67%
Restaurants	58%	Fast food	69%
Hairdresser and beauty salon	64%	Shoe shop	66%
Clothes shop	64%	Hotels	61%
Supermarket	59%	Real state agency	42%
Household items	55%	Art and Culture	60%
Banks and financial services	67%	Perfume shop	67%
Bakery	53%	Bazaar	72%
Coffee shop	55%	Car and motorbikes Dealership	39%
Mechanical workshop	53%	Travel agencies	65%
Copy shop	59%	Sports shop	66%
Pharmacy	60%	Optician's shop	66%
Fruit shop	48%	Florist	68%

TABLE 4.2: Average increase in path length after 100 iterations for the different categories.

As we can see, the percentages are similar for all categories, around 60%, except for some categories such as *Fruit shop* or *Car and motorbikes Dealership*. This is simply due to the fact that, by chance, the generated route passes close to all the establishments. It is not because these categories have a wide range of businesses.

Chapter 5

Conclusions

During this project, we have learned to deal with networks, both mathematically and computationally. We have treated basic algorithms that allow us to tackle problems, which at first sight may seem very complicated, but which with the use of these techniques are reduced to much simpler problems. With the help of these, we have given the graphs a geographical focus that has allowed us to apply and use them in everyday situations. This has made us observe how present many mathematical aspects are in human life and how they allow us to optimize many processes.

We have managed to solve the proposed problem, studying and analyzing different approaches and obtaining interesting results. We were able to choose the most appropriate one, based on what seemed most logical and intuitive, and always taking into account the defined objectives that allowed us to address the problem.

All this has enabled us to go further and consider new options. In this project, we have faced the problem from a totally experimental and explanatory point of view, focusing on the network environment and with no intention of developing a product. However, it is true that with the basis that we have laid and with some further implementations, a complete product could be obtained allowing the user to interact directly with it.

Furthermore, going deeper and thinking about how current recommenders work, the interest selection process we have talked about during the project could be fully automatized. For example, we could base it on the user's searches or interests shown on a mobile phone. This would not only allow these interests to be detected in a personalized way but would also allow a third version of the algorithm to be developed in which the 5 establishments most similar to the user would be chosen.

In a society where the volume of data is increasing day by day, this could become very relevant and be really accurate in recommending what is most suitable for each user.

Another perspective of the developed algorithm would be commercial. It could calculate the routes from all points to all points in the city for each of the defined interests and, by doing so, see which points people pass through the most for each of these interests. Then entrepreneurs who wanted to set up a business could use these results to decide which is the best area to open a shop.

Taking it a step further, we can think of more places where such an algorithm could be of interest. For example, the transport of goods or the delivery of packages or food at home. In this way, delivery times could be reduced by optimizing routes and adding stops that allow a delivery driver to handle several deliveries at once.

Thus, with all the work carried out during the project and with all the exploration paths mentioned above, we can conclude that the objectives proposed at the beginning of the work have been fulfilled. They have allowed the author to acquire a great amount of knowledge about this branch that was unknown to him, as well as to create an interest in this field. Improving his analytical and cognitive capacity.

Bibliography

Boeing, G. (2017). "OSMnx: A Python package to work with graph-theoretic OpenStreetMap street networks". In: *Journal of Open Source Software* 2.12, p. 215. URL: https://doi.org/10.21105/joss.00215.

Chang, K. P. et al. (Nov. 2011). "Discovering personalized routes from trajectories". In: *Proceeding of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks*, pp. 33–40. URL: https://doi.org/10.1145/2063212.2063218.

Cheng, A. J et al. (Nov. 2011). "Personalized travel recommendation by mining people attributes from community-contributed photos". In: *Proceedings of the 19th International Conference on Multimedea 2011*, pp. 83–92. URL: https://doi.org/10.1145/2072298.2072311.

Choudhury, M. et al. (July 2010). "Automatic Construction of Travel Itineraries using Social Breadcrumbs". In: *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, pp. 35–44. URL: https://doi.org/10.1145/1810617.1810626.

Clark, T., R. Lloyd, and K. Wong (Dec. 2002). "Amenities Drive Urban Growth". In: *Journal of Urban Affairs*, pp. 491–515. URL: https://doi.org/10.1111/1467-9906.00134.

Debord, G. (Sept. 1955). "Introduction to a Critique of Urban Geography". In: *Les Lèvres Nues* 6. URL: https://www.cddc.vt.edu/sionline/presitu/geography.html.

Dijkstra, E. W. (June 1959). "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1, pp. 269–271. URL: http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf.

Ezmale, S. and L. Litavniece (Jan. 2012). "Spatial planning as a tool for improving attractiveness of the places: case of Latgale region". In: *European Integration Studies* 0. URL: https://doi.org/10.5755/j01.eis.0.5.1071.

Fischer, M. M. (June 1959). "Spatial Analysis and GeoComputation". In: *Springer* 1. URL: https://doi.org/10.1007/3-540-35730-0.

Jensen, J. R. and R. R. Jensen (2013). "Introductory Geographic Information Systems". In: *Pearson* 1, pp. 195–257. URL: https://www.pearson.com/us/higher-education/program/Jensen-Introductory-Geographic-Information-Systems/PGM290591.html.

Kurashima T., Iwata T. Irie G. Fujimura K. (Oct. 2010). "Travel route recommendation using geotags in photo sharing sites". In: *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 579–588. URL: https://doi.org/10.1145/1871437.1871513.

Ludwig, B., B. Zenker, and J. Schrader (2009). "Spatial Analysis and GeoComputation". In: *Communications in Computer and Information Science, Springer* 53. URL: https://doi.org/10.1007/978-3-642-10263-9_9.

Mehta, H., P. Kanani, and P. Lande (May 2019). "Google Maps". In: *International Journal of Computer Applications* 178, pp. 41–46. URL: 1https://doi.org/10.5120/ijca2019918791.

Quercia, D., R. Schifanella, and L. M. Aiello (Sept. 2014). "The Shortest Path to Happiness: Recommending Beautiful, Quiet, and Happy Routes in the City". In: *Proc. of Conference on Hypertext and Social Media*. URL: http://dx.doi.org/10.1145/2631775.2631799.

Rey, S. J., D. Arribas-Bel, and L. J. Wolf (2020). *Geographic Data Science with Python*. URL: https://geographicdata.science/book/intro.html.

Tenkanen, H. (Dec. 2017). *Network analysis in Python*. URL: https://automating-gis-processes.github.io/2017/lessons/L7/network-analysis.html.

Wasser, L. et al. (July 2018). *Introduction to Vector Format Spatial Data*. URL: https://www.earthdatascience.org/courses/use-data-open-source-python/intro-vector-data-python/.