FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Interactive Spider Maps for providing public transports information

Sara Beatriz Gonçalves Santos

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Maria Teresa Galvão Dias

Second Supervisor: Thiago Sobral

July 19, 2019

Interactive Spider Maps for providing public transports information

Sara Beatriz Gonçalves Santos

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: António Miguel Pontes Pimenta Monteiro
External Examiner: Marco António Morais Veloso
Supervisor: Maria Teresa Galvão Dias

July 19, 2019

Abstract

Public Transport systems are a part of everyday life and a fundamental support for cities mobility that ought to be encouraged as an alternative to private transport. Additionally, with the continuous growth and complexity of the networks, it is essential that the users have access to transportation maps that help them easily understand the underlying network, thus facilitating the user experience and public transports ridership.

The term transport map is a comprehensive denomination for maps with the purpose of transmitting public transport information, e.g. stops, routes and points of interest. Schematic maps are a common type of maps used in public transport. This type of map results from the process of simplification that loosens up the geographical constraints in favour of user readability. The goal is to translate the mental representation of the network, presenting the reader the available services and navigation possibilities. The most well-known example is the London Tub map.

One specific type of schematic maps is Spider Maps that mix elements from both geographical and schematic maps. These maps have a great potential for transmitting public transport information, answering the question *"From where I am, where can I go?"*. However, even nowadays spider maps are mostly manually generated. Nonetheless, some research aim at automatically produce them.

Even though several techniques can be applied to support the generation process, current solutions are very time expensive and require great computational effort. This dissertation presents a solution to automatically generate spider maps, proposing an algorithm that adapts current methods and generates viable spider map solutions in a short execution time. Furthermore, interaction and visualisation techniques in maps are explored. Results show successful spider maps solutions for areas in Porto city.

Resumo

Os sistemas de transportes públicos fazem parte do dia-a-dia de milhares de pessoas e são um suporte essencial para a mobilidade das grandes cidades. Como tal, devem ser incentivados como uma alternativa viável ao transporte privado. Para além disso, com o crescimento contínuo e o aumento da complexidade das redes é essencial que os utilizadores tenham acesso a mapas de transporte que os permitam facilmente compreender a rede subjacente, de modo a facilitar a experiência e usabilidade dos transportes públicos.

O termo *mapa de transporte* é um denominação compreensiva que engloba todos os mapas com o proposito de transmitir informação sobre transportes públicos, por exemplo, paragens, linhas e pontos de interesse.Os mapas esquemáticos são um tipo de mapas frequentemente usado para transmitir informação sobre a rede de transportes públicos. Estes mapas resultam de processos de simplificação que enfraquecem as restrições geográficas em favor da legibilidade. O seu objetivo é servir como uma representação mental da rede, apresentando ao utilizador os serviços disponíveis e todas as possibilidades de navegação.

Um tipo específico de mapas esquemáticos são *spider maps* que incorporam elementos de mapas geográficos e de mapas esquemáticos. Estes mapas apresentam uma grande capacidade para transmitir aos passageiros informações sobre a rede de transporte, respondendo à questão *"De onde me encontro, até onde posso ir?"*. Mesmo nos dias de hoje os *spider maps* são maioritariamente gerados manualmente, no entanto, alguns estudos ambicionam automatizar a geração deste tipo de mapas.

Apesar de existirem várias técnicas que podem ser aplicadas no processo de geração de *spider maps*, as soluções atuais são complexas, podendo necessitar de bastante tempo e poder computacional para produzir resultados. Esta dissertação apresenta uma solução para gerar automaticamente *spider maps*, propondo um algoritmo que adapta métodos atuais para gerar estes mapas em curtos tempos de execução. Para além disso, é explorado o uso de técnicas de interação e visualização em mapas. Resultados mostram resultados bem-sucedidos de geração automática de *spider maps*.

Acknowledgements

Special thanks my supervisor and co-supervisor Professor Teresa Galvão and Thiago Sobral for the guidance, support and motivation for approaching this challenge and throughout the development of this project. I would also like to thank OPT[1] for assisting this work by providing all the necessary data, thus making it possible to validate with real data.

To all my colleges and friend whom I shared five year of college experiences, hard work and good memories, I am grateful. Finally, a special thank you to my parents and grandmother who made possible for me to pursue my goals.

Sara Santos

[1] http://www.opt.pt/

This work is partially financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e Tecnologia within project POCI-010145-FEDER-032053

"Follow your bliss"

Joseph Campbell

Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
CEN	European Committee for Standardization
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
DCE	Discrete Curve Evolution
DOM	Document Object Model
ET	Execution Time
GIS	Geographical Information System
GPS	Global Position System
GTFS	General Transit Feed Specification
HCD	Human-Centred Design
HCI	Human-Computer Interaction
HTML	Hyper Text Markup Language
ICA	International Cartography Association
ISO	International Standards Organization
LBS	Location Based Services
PDA	Personal Digital Assistant
POI	Point Of Interest
ROI	Region Of Interest
SIRI	Standard Interface for Real-time Information
SVG	Scalable Vector Graphics
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Context and Motivation

Every major city has a complex public transport system that is part of everyday mobility of millions of citizens. These systems ought to be encouraged as an alternative to private transport, which may help decrease pollution rates and traffic.

Public transport maps provide simplified representations of the underlying network, making them easy to interpret, hence facilitating the user experience and fostering an increase on ridership. Thus, the term *transportation map* is a comprehensive denomination for maps with the purpose of transmitting public transport information. One of the most used map types is schematic maps. They undergo several generalisation and simplification processes, so they translate the mental representation of the network, presenting the readers the available services and navigation possibilities. A specific type of schematic map is designated *spider map*, which combines elements from both geographical and schematic maps.

Spider maps are used to represent very complex areas, such as bus networks in city centres, providing all the travelling possibilities of a certain geographical area. Hence, they are valuable in the pre-planning stage of a trip by providing better geographical context, where awareness of all travelling possibilities may be difficult. These maps are composed by a hub, typically a rectangular geographic map depicting the spatial context and where the schematic lines emerge from. Furthermore, there are several design constraints that these maps need to follow, such as line angles and the location where lines emerge in the hub. Thereby, automating the generation of spider maps is a complex computational problem, since all the aforementioned restriction need to be followed. Figure 1.1 depicts a spider map created for the São João hospital area.

Even though spider maps are a favourable representation for providing passengers public transport information, the generation process is still manual and relies on the expertise of designers. There are several methods and techniques that can be applied to automate the spider map generation process, but current solutions are computational and time expensive.

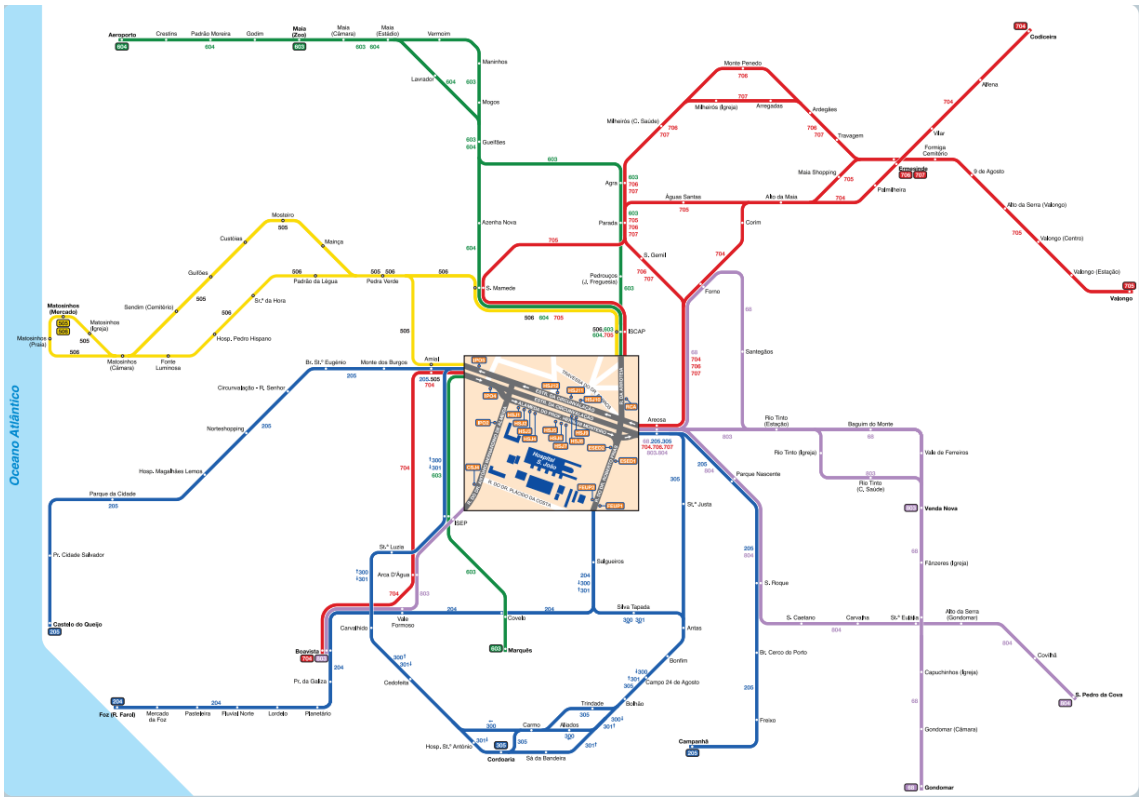

Figure 1.1: Spider map for Hospital São João area in Porto: a busy network area, with multiple stops and lines surrounding the hospital [Soc19]

The increasing ownership of smartphones and the diffusion of Location-based Services enabled new possibilities for map interaction. Hence, the integration of the automatic and personalised generation of spider maps may also provide improvement in usability and spatial context of this type of maps [Mou15].

1.2 Objectives and Research Questions

The main goal of this dissertation project is to develop a solution able to generate a feasible spider map at run time, presenting an algorithm that modifies and adapts current techniques. The ambition is to tackle the complexity of the problem and present viable solutions with short execution times and using less computational power than current solutions. Thus, it aims at simplifying the traditional spider map generation process and potentially make an impact on the use of spider maps for providing public transports information. A spider map result is considered viable if it complies with all the design restrictions set for this type of maps.

Furthermore, this work aims at integrating interactive capabilities, taking advantage of the digital map advantages. Verdi [VCW02] states that knowledge from print maps can be applied to digital maps, but the opposite does not apply, since digital maps have a learning possibility superior to traditional maps. Interactive capabilities can be applied before the generation process,

e.g. enabling the user to choose the spider map hub, and after the spider map is generated, for instance, clicking on stops to check additional information.

Henceforward, two research questions are identified that motivate the development of this work:

- How can the spider map generation process be automated to produce map results in real-time and considering the user input?
- Can interaction and visualisation techniques be integrated with spider maps to provide public transports information?

To answer these questions, the ambition is to develop an algorithm capable of generating spider maps. Afterwards, a prototype is created that incorporates the algorithm with interaction capabilities for the automatic generated maps. Thus, the algorithm should consider the computational effort required, ensuring the prototype usability is not compromised by long execution times.

1.3 Document Structure

Along with the current introductory chapter, this document contains four chapters that aim at discussing the state of the art and present the proposed problem and the developed solution.

Chapter 2 regards the study of the state of the art of the relevant topics on how maps are used to provide public transport information. This chapter contains several subsections that demark areas of interest for the work to be developed. First, the evolution of maps in public transports is briefly presented, then a study of schematic and spider maps is depicted and, finally, research in interaction methodologies is discussed.

Chapter 3 presents the proposed dissertation problem and the developed solution. This chapter depicts all relevant points related to the solution, such as data model, solution architecture, developed algorithm, tests and results.

The final chapter, Chapter 4, concludes the document and presents future work and possible improvements.

Chapter 2

Maps for providing Public Transports information

Cartography, according to the International Cartography Association (ICA) [Vis89], is *"the art, science and technology of making maps, where maps may be regarded as including all types of maps, charts and globes representing Earth or any celestial body at any scale"*. Thus, maps can be depicted as a graphical representation of a geographical setting, that reduces reality, applies levels of abstractions and uses signs and symbolism to represent a portion of space.

Maps have a long history through time, being used for wayfinding and descriptions of the real world, or even cosmological representations of religious manner. Moreover, representations differ by culture and time, culminating in the modern concept and representation of maps, e.g., north-up representation and print maps as known today. Mapping skills impact societies and may be viewed as a cultural heritage [Int19].

With the industrial revolution, maps began being used for describing railways and routes and, since then, with the emergence of Geographical Information Systems (GIS), their use became common in several activities such as in public service planning and representing public transports networks [Mou15].

2.1 Maps for Public Transports

Public transports are any means of transport, such as bus, trains, subways or other form, available to the public, with ridding fares and running on fixed routes [Eng19]. These transportation systems are a fundamental support for passengers' mobility.

The complexity of public transports networks tends to increase as cities grow. Thus, it is essential that users have support in understanding the underlying network in order to facilitate the user experience and public transport ridership. This type of support can be achieved by providing the users transportation maps. A transportation map is a comprehensive denomination for maps

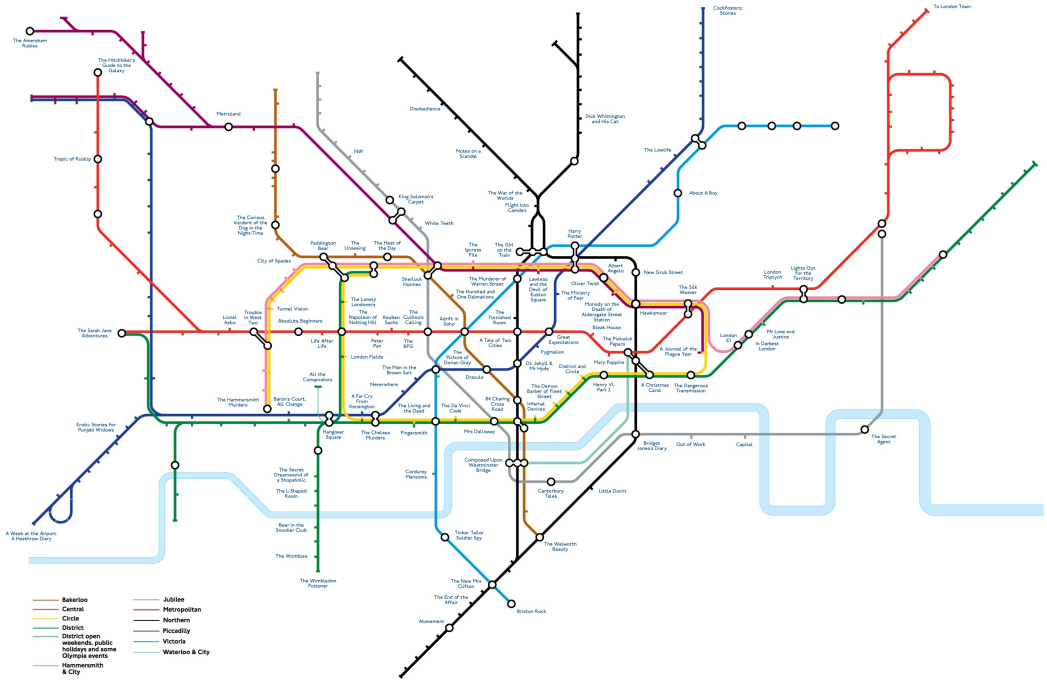

Figure 2.1: London Tube Map (modified from [In 19])

with the purpose of transmitting public transports information, such as, stops, routes or points of interest, and are today a frequently used form of graphic communication. A well know example of a schematic transportation map is depicted in Figure 2.1, that represents the London Tube network.

Even though schematic maps are a common type for representing public transport networks, geographical maps are also commonly used for wayfinding or obtain information about a geographic location. Geographic maps are typically represented with north-up, however, other representations are also used such as head-up typically used in GPS navigation systems [Por07].

Schematic transport maps often have common characteristics of mental knowledge representation, presenting qualitative spatial concepts, i.e., instead of representing accurately physical locations on the map, some geographical features and constraints are loosened so that users are capable of capture the basic structure of the network. Hence, map makers use a wide range of generalisation techniques that aim improving map clarity and emphasise important information. Among these techniques is *simplification* (reduce represented features/points), *aggregation* (group points), *exaggeration* (amplify specific portion of an object), *displacement* (separate objects) and others. The process may cause some information to be lost and imperfection in representing the visual world (diagrammatic representation), however this leads to a straight forward representation, not overwhelming the user with too much information [Mou15]. Schematic maps are created by applying several of these techniques, which will be analysed in Section 2.3.

2.1.1 Network Representation for Public Transport

The creation of GIS and the continuous integration of public transport information in digital devices and systems lead to the need of creating reliable information systems. Public transport services increasingly depend on these systems to ensure quality and efficient operation of their services, and to transmit accurate real-time information to their passengers. Moreover, information systems need a data model that can represent and incorporate public transport information, such as routes, stops and timetables, facilitating interoperability and consistency throughout various systems. There are several data models available that model and incorporate different types of information, for instance, Transmodel [CEN19b], Standard Interface for Real-time Information (SIRI) [CEN19a] and General Transit Feed Specification (GTFS) [Gen19].

Transmodel has been developed in the scope of several European projects and is now the European standard for public transport reference data model. Its goal is to provide a framework for describing and aggregating data models to represent areas of public transport operations. Thus, it incorporates several features of public transport information and service management. It aims at providing a standard to facilitate interoperability, making it possible for different operators, authorities and software providers to work together at improving public transport information [CEN19c].

SIRI [CEN19a] is based on the Transmodel standard and is a CEN (European Committee for Standardization) Technical Standard for exchanging real-time public transport operations between different computer systems. It follows an XML format to structure information about schedules, vehicles, and connections, incorporating general information messages about the current operation of the services. This information can be used to provide passengers with information about real-time progress of individual vehicles, bus trajectories, status messages about the state of the services and other useful information [VSV19].

Nowadays, a widely adopted system is Google's GTFS [Mac12], that delineates a standard format for public transports schedules and associated geographical information. This system is composed of *feeds*, where public transports operators publish their transit data in a series of formatted text files (see Table 2.1), each modelling a particular aspect (stops, routes, trips, schedule and other information). These files follow the Comma-Separated Value (CSV) format, that establish a format to represent different data by enumerating several attributes, such as route's code, name and destination. These feeds can be submitted into Google's Transit API, making the *feed* public and available for developer to use [Goo19].

2.2 Location Based Services

Location Based Services (LBS), also designated as location-aware services, location-related services or just location services [Kup05], denote any application that integrates geographical location with the notion of services in the general term, e.g., car navigation systems, tourist tour planning, among others [SV04].

Table 2.1: GTFS Feed Files [Goo19]
* Required elements

Filename	**Details**
agency.txt *	Defines one or more transit agencies
stops.txt *	Defines individual locations of routes (stops)
routes.txt *	Defines the transit routes
trips.txt *	Defines a trip for each route. A trip is a sequence of two or more stops, occurring at specific time
stop_times.txt *	Defines arrival and departure times for each stop. for each trip
calendar.txt *	Defines weekly dates for the availability specific services. This file is required unless all dates of services are defined in *calendar_dates.txt*
calendar_dates.txt	Defines exception of services dates defined in *calendar.txt* file. If *calendar.txt* is omitted, then *calendar_dates.txt* is required and must contain all dates of service.
fare_rules.txt	Defines rules for fares
shapes.txt	Defines rules for drawing lines on a map to represent routes
frequencies.txt	Defines time between trips for routes with variable frequency of service
transfers.txt	Defines rules for making connections at transfer points between routes
feed_info.txt	Defines additional information about the feed, such as publisher, version and expiration information

The usage of these services evolved along with the usage of mobile devices for providing geographic information and, nowadays, have a greater usability potential by incorporating services and information provided by the Internet [Mac12]. LBS are considered by several authors as the intersection between these technologies: Internet, mobile services, positioning systems (e.g. GPS) and GIS, such as depicted in Figure 2.2.

Furthermore, LBS systems can be used to grant tailored information to the user needs by adding the location of the user as context. The established relationship with GIS enables LBS capable of providing answers to several spatial questions, such as, *"Where can I go?"*, *"Where am I?"*, *"What surrounds me?"* [Mou15].

Hence, LBS can be considered a network-based service that integrates the mobile device's location to improve information value to the user. In [Rei04], the author identified five types of actions users typically performed when using such services:

1. **Locate:** identify the current user's location
2. **Search:** user desires to search specific locations, objects, events or persons
3. **Navigate:** wayfinding to a different location
4. **Identifying:** search for information about a place
5. **Checking:** confirm or search for locations or events near the user

Figure 2.2: Integrated technologies of LBS [Abd16]

The several types of tasks provided by LBS systems enable a set of options for new digital platforms. For instance, some public transports operators currently provide users LBS systems that can assist passengers in ridership. Furthermore, these systems can be improved by integrating schematic maps and spider maps with LBS and GIS systems [Mou15].

2.3 Schematic Maps

A type of map used in public transports is the schematic map, as they fulfilled the need for better and simpler transportation maps to describe complex public networks. Their importance is connected to the resemblance of mental knowledge representation that serve as support for cognitive processes [KRBF05].

Thus, schematic maps can be defined as a diagrammatic representation of transportation systems, e.g. subways, trams and buses, composed by highly generalised lines of the network [AH06]. Schematic maps provide a simpler way to learn the underlying network of public transports, by giving emphasis to certain aspects, while removing unimportant information [Mou15]. This process is called the *schematisation process*.

The term *schematisation* is often defined as the process of reducing information, from both technical and cognitive domains [KRBF05], i.e., selection of certain aspects of a scene to represent the whole. However, this process does not have a strict set of rules and steps, thus it cannot be operationalised and is hard to optimise. Nonetheless, some authors define three distinguish high level processes that schematisation should follow: abstraction, idealisation and selection [OG15]. First, it is necessary an abstract mental conceptualisation of the geographic map, so some mental concepts could be defined. Then, the idealisation process should define what information

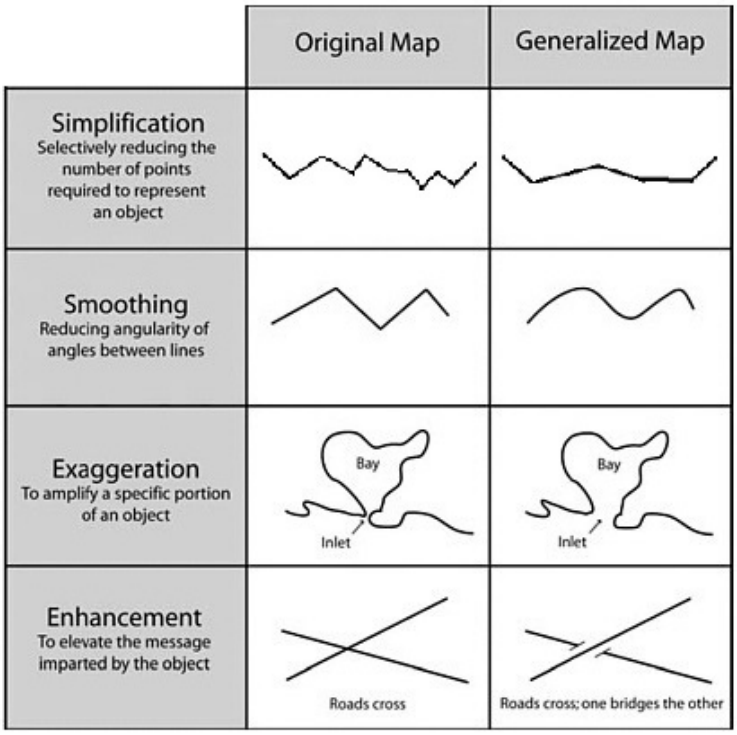

Figure 2.3: Types of generalisation processes (modified from [Cou07])

the schematic map should transmit. Finally, the selection defines what elements are relevant to transmit the information and what elements should be removed.

Schematisation also relies on generalisation techniques that aim at improving map comprehensibility by emphasising the most important information [Mou15]. Among such techniques is simplification, smoothing, exaggeration and enhancement (see Figure 2.3). The resulting maps represent knowledge needed for a specific task [KRBF05], hence, the more clearly the environment is represented, the higher is the map usability.

The schematisation process is mostly a manual process that relies on the expertise of the designers and cartographers, even though some processes are automated or have support from computer systems [Mou15]. Generally, the process starts with a non-diagrammatic map (e.g., a map where position of line points corresponds to their geographical location), then pre-processing methods are applied, for instance, adapt the map to a grid. It follows an incremental optimisation process that applies several methods, such as simplifying the line geometry or moving stops. Finally, in the post-processing phase the aesthetic is improved by finishing the last map details, such as line colours and shapes.

Hence, it is important to provide and improve support for the schematisation problem. There are two types of approaches for the schematisation problem [KRBF05]: a data-driven approach, where maps are constructed by iteratively applying simplifications, and a cognitive conceptual approach that first identifies the map cognitive concepts and then integrates them with spatial knowledge. Throughout this project, the adopted approach was data-driven, focusing on applying

different data operations.

There are several styles of schematic maps [Ave02], e.g., classic, French, Scandinavian or Dutch. Each style defines a different set of rules for presentation, for example, in the French style a different colour is used to distinguish every service of each transport mode, and in the Dutch style trams are represented by a double line. The adopted styles vary by country, for instance, the classic styles are adopted by Great-Britain, Portugal and Italy, while Scandinavian are adopted by Germany, Austria and some cities of Spain.

Even though there are deviations between styles, some common characteristics and constraints of schematic maps can be defined [AH06]. The schematisation process leads to geometry inaccuracies, i.e., lines length, orientation and resolution are not geographic accurate.Thus, a new concept of map point is introduced, translating not the geographical location but some relevant location in the schematic map, e.g., a set public transport stops, obtained by the schematisation process.

In schematic maps, such as the one depicted in Figure 2.4, there are several design constraints. The line directions follow octilinear angles, i.e., angles are of 0, 45 or 90 degrees. For instance, in Figure 2.4 lines only have octilinear angles, which provides a more aesthetically pleasing solution [AM00]. Furthermore, overlapping network lines are separated by a minimum distance, that could be zero. For example, in Figure 2.4, the green, orange and brown overlap in some segments. Hence, they are represented separated, so it is straightforward to the user that several lines are available in that specific part of the network. Also, some breaks and changes in line direction can be incorporated to add some sense of the original geography and provide better visualisation [Ave02]. Furthermore, even though transport lines can be represented with the same colour, many schematic maps represent different network lines in different colours to enhance the legibility and difference the map elements.

Schematisation is an important problem as it provides insights on the advantages and limitations of spatial language [OG15]. Furthermore, schematic maps are important and valuable in the wayfinding process, since they provide information about the surroundings of a given location and establish relations in a simplified manner [CBKF00].

2.3.1 Automatic generation of Schematic Maps

The schematic map production remains mostly a manual or assisted process, however, there are several efforts to develop methods to optimise this process [Mou15]. Even though these processes do not establish a framework for automating the whole schematisation process, they contribute with essential methods to ease this process and provide a foundation to automatically produce feasible solutions that can later be improved by expert designers.

One of the assisted methods is to create the initial non-diagrammatic map using software to translate the geographical coordinates of points in transportation networks, then grid adaptation can be performed. This method can be used to preserve the topological structure of the linear network, as it disclosures the underlying projection [KK00].

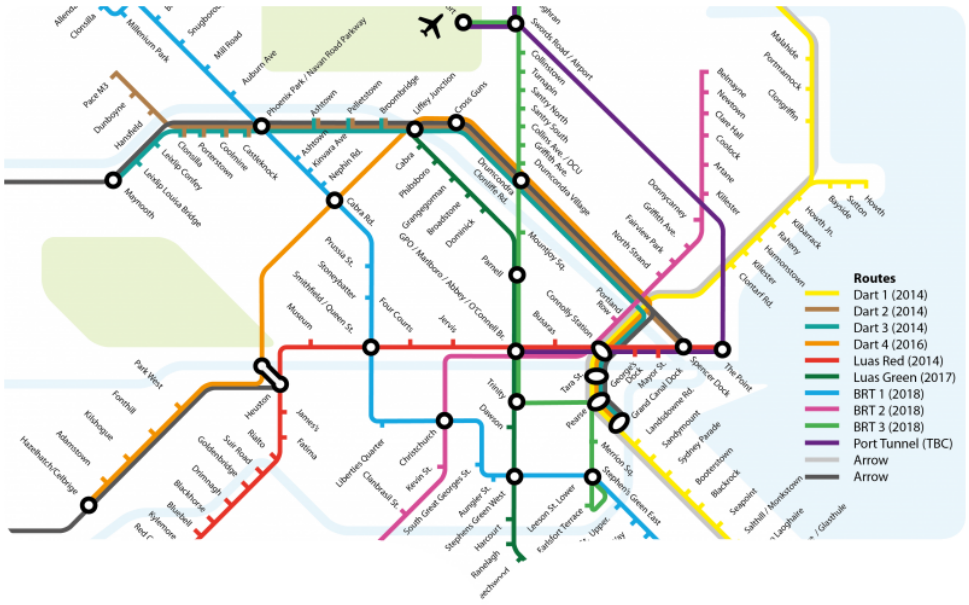

Figure 2.4: Schematic map of Dublin transports [An 19]

The general idea of grid adaptation is to assign each map point to the closest grid point (intersection of two grid lines). However, there could be more than one candidate to a grid point. In [SS] a solution for adapting a map to a grid and solving these conflicts is proposed. The model uses a graph *G(V,E)* for the abstract representation of the schematic map, where *V* is the set of vertices and *E* the connections between the pairs of vertices. Thus, in the schematic map the vertices represent the turning points in network lines and the edges the segments that connect those points. Furthermore, a set of rules is defined to ensure some important characteristics of schematic maps:

- **Orientation Constraint:** angles between segments should be of 0, 45 or 90 degrees (octilinear), such as previously discussed and depicted in Figure 2.4

- **Distance Constraint:** overlapped segments should be separated by certain distance, similar to the orange, brown and green lines in Figure 2.4

- **Length Constraint:** ordering of length in segments should be maintained after schematisation, i.e., if segment *A* is shorter than *B*, after simplification the length of *A* should remain smaller than the length of *B*. For instance, in the blue line of the map represented in Figure 2.4, the map distance between Stoneybatter and Prussia Station is smaller than between Stoneybatter and Liberties Quarter. Hence, it is known that the real distance between Stoneybatter and Prussia Station is also smaller.

- **Displacement Constraint:** the chosen point of displacement should be the one closest to the original point, as depicted inf Figure 2.5, which can result in several points being candidates to a grid point.

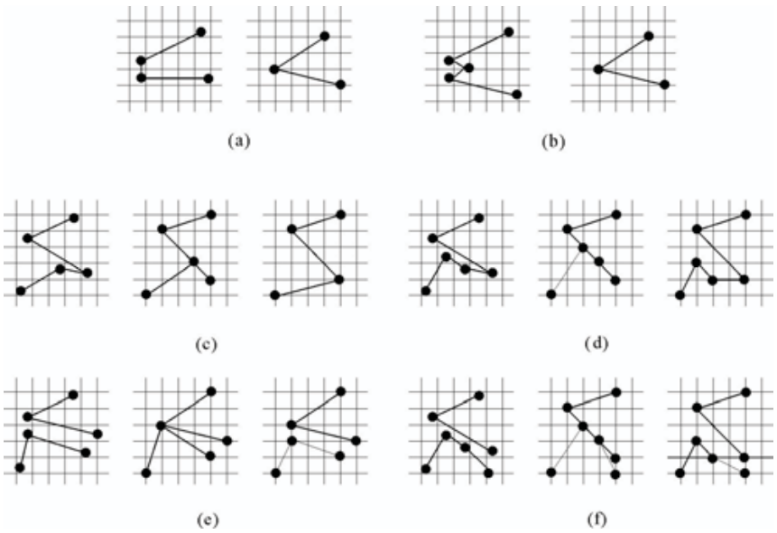

Figure 2.5: Line generalisation in the grid adaptation process [SS]

Afterwards, the graph is enclosed on a regular grid, so vertices are centred on grid points, yet edges do not require to follow grid lines. The initial layout corresponds to the geographical, then a several transformations are applied to produce a schematic map that ensures topological, geometric and semantic constraints [SS]. The next step is to assign each point to the nearest grid point, assigning new coordinates calculated using formula 2.1, where *x(v)* and *y(v)* are the initial coordinates, *x'(v)* and *y'(v)* the resulting coordinates and g the target resolution (distance represented by pixels). However, this will not prevent duplicated grid attributions, thus Weihua et al. [SS] present different generalisation techniques for each case of duplicated inner vertices and segments (Figure 2.5):

$$\begin{cases} x'(v) = g \times \left[\frac{x(v)}{g} + 0.5 \right] \\ y'(v) = g \times \left[\frac{y(v)}{g} + 0.5 \right] \end{cases} \tag{2.1}$$

- When they are contiguous or adjacent the duplicated one should be deleted if it does not alter the topology of the map (Figure 2.5 (a), (b) and (c))
- When they are not contiguous or adjacent (Figure 2.5 (d), (e) and (f)) the overlapping segments and duplicated vertices should be separated to a different grid point.

If a vertex needs to be displaced, it should follow one of the eight possible directions to ensure that the orientation constraint is followed. All possible orientation displacements are computed, and the vertex should be moved to the position with nearest displacement distance. The final step is to ensure that topology is maintained, when performing a displacement from original position q to a new position q'. A topology test is performed by forming a triangulation between these points and the endpoint p of the segment being analysed; if there is no other point inside the triangle and no line segment that crossed the segment qq', topology will be preserved. For instance, in Figure

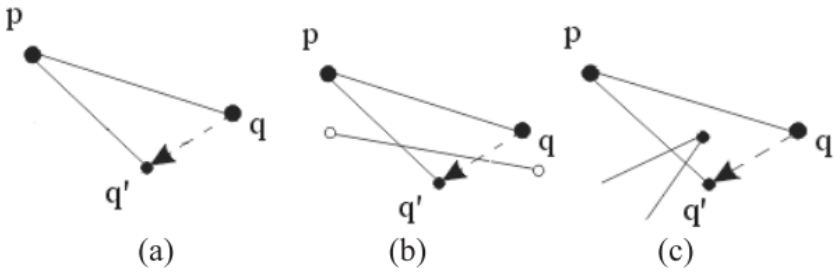

Figure 2.6: Topology tests performed to validate a displacement [SS]

2.6 a), there is no line segment intersecting the edge *qq'*, thus the topology is maintained and the displacement is allowed. On the other hand, in Figure 2.6 b) and c) the topology will change [SS].

Nonetheless, adapting maps to a grid can lead to saturated areas, while others remain almost unoccupied. For instance, this can occur when representing complex centre areas that have lines ending on city outskirts. This effect can affect the map, since these complex areas are harder to understand. Hence, Sarkar and Brown [SB92] proposed a method denominated fish-eye that applies different scales throughout the map, thus enabling magnification of crowded areas. This is a *Focus+Context* technique of great value to the schematization process, since it emphasises important information, while maintaining the global context [Mou15]. Fish-eye views are also useful as a visualisation technique as they avoid switching between multiple views. However, the main drawback of this technique is the introduction of undesired distortion [BGS01].

Line generalisation is considered one of the major processes of schematisation, as it simplifies lines as much as possible, while maintaining their overall shape [Mou15]. One of the most recognised algorithms for this process was proposed by Douglas and Peucker [DP11], which divides recursively the line at the node which is furthest from a line between the two-end point (see Figure 2.8). This algorithm has a temporal complexity of *O(n log n)*, however, several other approaches were developed that aim optimising and improving this method. For instance, in [Saa99], the stopping condition is modified to avoid the topological errors resulting from the original algorithm.

Another important method for schematising local features was developed by Latecki and Lakämper (2000), based on Discrete Curve Evolution (DCE) [KRBF05]. The goal of DCE is to remove in each iteration the section of the object the least relevant for the overall shape. Thus it can be applied in cartography generalisation as it simplifies the line details. Furthermore, this algorithm examines pairs of line segments, so entities represented only by one point of endpoints of segments are not considered. Also, some points must remain unchanged, since removing or changing them will violate topological information. These points usually belong to more than one entity and are marked as fixed points. Figure 2.9 depicts an example of an object shape before and after applying the DCE algorithm.

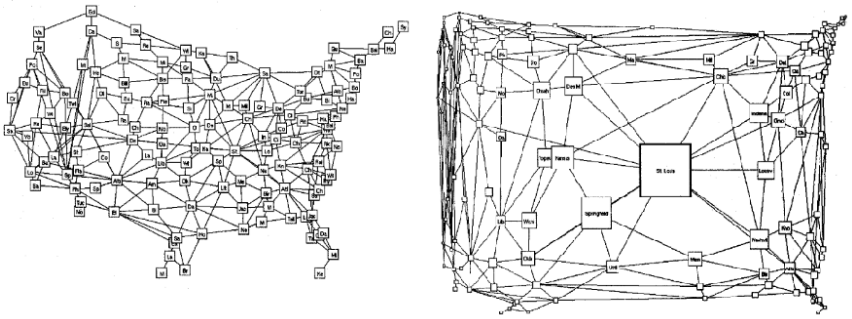

Figure 2.7: Result of applying the fish-eye algorithm to a map (adapted from [SB92])

2.4 Spider Maps

Even though spider maps have a great potential for providing public transports information, few works address this type of maps and how to automate their generation. The majority of the contents of this section focus on the work and efforts made by João Mourinho [Mou15] in the development of techniques to automate the generation of spider maps.

Spider maps are based on spider diagrams [AH06] and combine elements from both geographical and schematic maps. These maps are often used to represent complex public transport network, for instance, bus networks in a city centre, and provide passengers information in the pre-planning stage of trips, answering the question "From where I am, where can I go?" [MD16]. An example of a spider map is depicted in Figure 2.10, that represent the complex bus network around Baker Street in London.

These maps are characterised by a central area, a hub which represents the geographical context [Mac12].The schematic lines that represent the network routes emerge from the hub. Along with the map, a route finder table is also provided to indicate the direction and route that are associated to each stop within the hub.

The hub is generally depicted by a rectangular shape and details a geographic map of the location, proving the spider map a better spatial context. Around there are located the points that connect the route lines with the stop inside the hub. This corresponds to the points where lines emerge from, hence, their location in the frame should consider route orientation and the stop location within the hub.

Similar to the schematic maps described in Section 2.3, the schematic lines in the spider map do not follow the geographic layout, since they are the result of several simplification and displacement operations [Mou15]. Moreover, spider maps adopt the concept of map point, which describes a relevant point in the map, for instance, stops along the line route, located at a certain canvas coordinates that do not relate to the real geographical location.

Spider maps' schematic lines are defined by a set of segments and map points, some of them shared with different lines, and a start and ending map point. Shared segments are drawn parallel and lines only follow 0, 45 or 90 degrees orientation angles. Segment nodes relate to route stops,

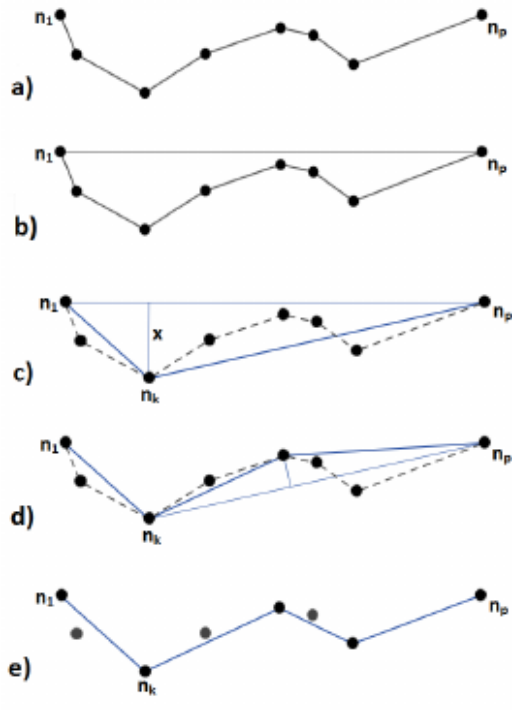

Figure 2.8: Example of Douglas and Peucker algorithm iterations [MM12]

however, some stops may be grouped together if they are geographically closed. Moreover, to increase spatial awareness, geographical accidents, such as rivers or seashore, can be added to the map [Mou15]. For instance, in Figure 2.10, a representation of the river Thames was added to the spider map, thus offering a better spatial context for the schematics.

Spider maps have several other design constraints that should be considered in the generation process. For instance, lines have a certain colour, usually defined by the transportation provider, and position of stops and line labels. Similar to schematic maps, spider maps generation is mostly a manual process. However, several techniques for simplification and generalisation can be borrowed from the schematic process. Hence, João Mourinho [Mou15] depicts a set of eight guidelines that spider maps should follow:

1. **Simplification of lines–** Generalise the line as most as possible, while maintaining overall shape.

2. **Group map points–** If several map points are very close together and have similar names, most likely they are related; hence they can be grouped together. This step must be taken carefully, as it can eliminate relevant map information.

3. **Zoom in crowded areas–** Emphasise crowded areas by zooming, which increases readability.

Figure 2.9: Example of the application of Discrete Curve Evolution algorithm (modified from [JDWH02])

4. **Remove or simplify environment features–** For instance simplify the shapes of geographical accidents. For instance, in Figure 2.10 the shape of river Thames is relaxed, keeping the important aspects, such as the topological relations of stops with the river.

5. **Group segments that have the same start and end nodes–** draw lines parallel if the segments share the same start and end nodes.

6. **Simplicity over completeness–** depict only the essential elements on the map

7. **Symbol shapes and colours–** use different symbols to emphasise or deemphasise certain map characteristics.

8. **Emphasise important aspects–** direct the user's attention to the most important map aspects, for instance, emphasise the hub to enable a better spatial context to the users.

2.4.1 Automatic generation of Spider Maps

As previously mentioned, throughout recent years there has not been great progress for automating the generation of spider maps.

João Mourinho [Mou15] proposed a method to automatically generate a spider map for a hub location, given a transportation network with the same characteristics described above. Since the current general generation process relies mostly on the design expertise and evaluation, the goal is to automatically generate a spider map that preserves topology and ensures the aforementioned restrictions (e.g., line angles).

An important solution this method provides is a complete model representation for the spider map. The spider map SM is defined as $SM = (P, V, H, E, L, A, Gr)$, where P is a set of map points, V the vertices, H the hub, E a set of direct edges, L a set of lines, A a set of angles and Gr a set of geographical restrictions. Figure 2.11 depicts a spider map represented by this model.

The initial algorithm state is a geographical accurate map, i.e., map points correspond to the accurate (or similar) geographic location, then a multi-criteria algorithm is applied where the decision variables are the spatial coordinates of each vertex and point belonging to the spider map.

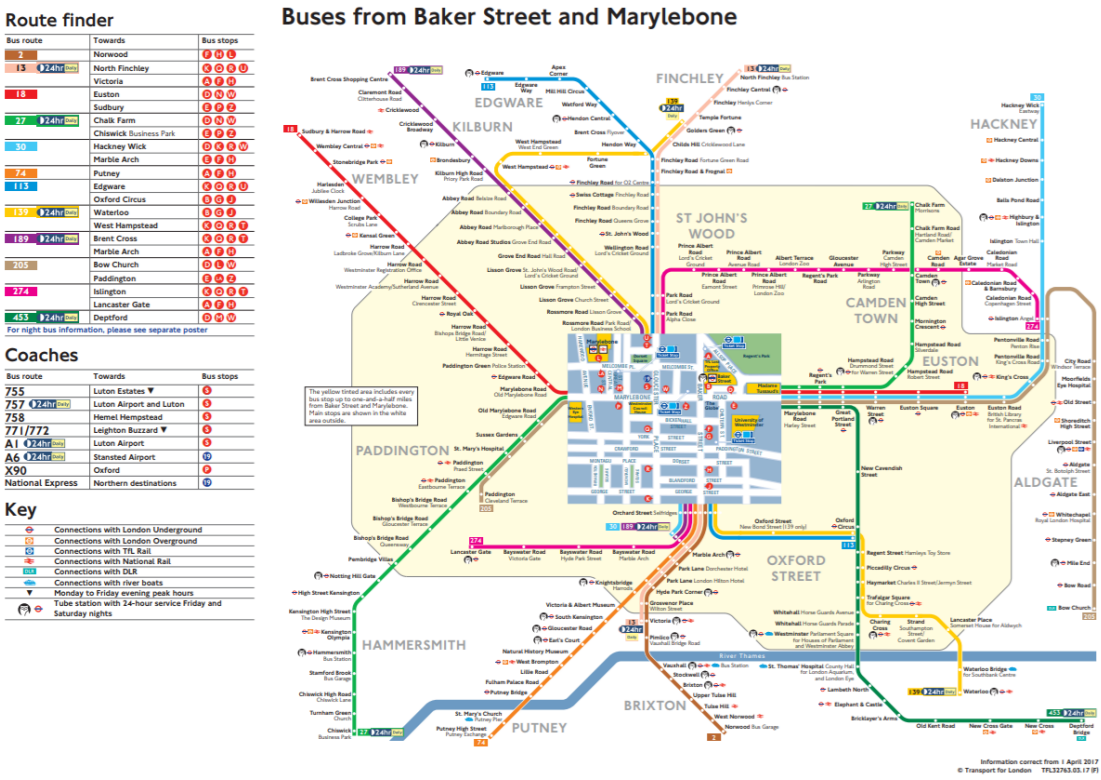

Figure 2.10: Spider map of bus network of Baker Street and Marylebone [Tra19]

The goal is to minimise the objective function while ensuring a set of constraints and design guidelines. Furthermore, two types of constraints are defined: *soft constraints* mostly related to visual qualities and should be followed if possible, and *hard constraints* that ensure a feasible solution and should be enforced. The objective function translates how soft constraints are followed, i.e., if all soft constraints are completely respected, then the objective function is zero, which means the solution is *"optimal"*.

The solution successfully attained the proposed goals. However, this is a complex multi-criteria optimisation problem with great computational effort. For instance, for the default parameters results were obtained in execution times around 5 seconds for simpler maps and 12 seconds for more complex solutions. However, when testing the adjustment of parameters to increase quality, such as the search radius for possible map point displacements, the execution times obtained increased significantly. The quality of the obtained results increased and execution times averaged 14731 seconds [Mou15]. Thus, for a dynamic mobile environment, which has less computational power and should produce results in a shorter time span, this solution needs some adaption, for instance, discarding some constraints.

Ribeiro et al. [RRL12] also proposed a solution to automatically generate a schematic map. Even though it does not fully integrate all the constraints needed for a considerable feasible spider map solution, it proposes a fast solution for the schematic portion of spider maps, which can be an interesting technique for dynamic mobile problems.

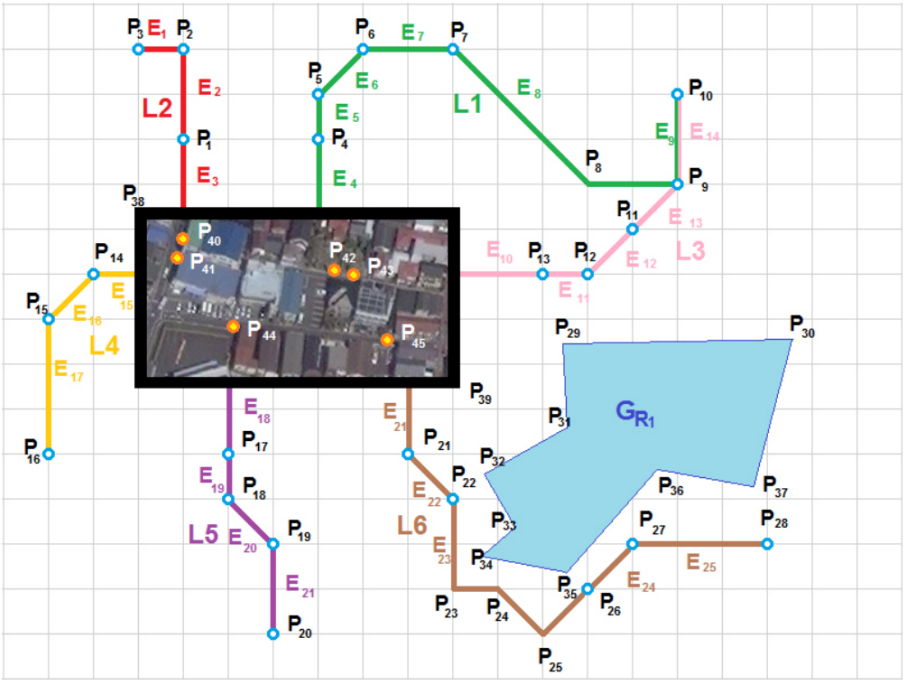

Figure 2.11: Example of spider map model presented in [Mou15]

The proposed approach is based on the application of force-direct algorithms. Force-direct algorithms are a class of graph algorithms that aim at drawing graphs in an aesthetically pleasing way. Also, this algorithm is computationally lightweight and relatively quick to implement, providing acceptable results, though not optimal. The algorithm's goal is to position nodes so that all edges are approximately equal length, there are as few crossings as possible and objects are distributed uniformly. The method is divided in three steps: initial dataset manipulation, force-directed iterative loop and final adjustments.

Force-directed algorithms apply forces to move nodes to better positions. The algorithm ends when the forces have reached an equilibrium. This approach uses Coulomb's and Hooke's laws to represent, repulsion and attraction forces between nodes, respectively. Additionally, the algorithm implements a set of assumptions and rules, such as two lines intersect if at least a pair of edges intersect, or the flow of execution is dependent on the parametrization.

An important step in this approach is the parametrization, since it influences the quality of the final result. Several parameters are defined, concerning a threshold distance in which nearby nodes are merged, definition of k value constant for Hooke's and Coulomb's law and design restrictions, such as pinning start nodes that can be altered to improve the result map.

To test and evaluate the solution, a prototype and an evaluation function were created. Several tests were done with different parameters and total number of nodes. In general, with an average of 15.08 seconds, 95 nodes, 5 lines and 129 edges. The evaluation function combines criteria, such as displacement from original position, line length variation and line crossing variation.

This approach presents relatively fast results and is able to simplify lines, merge close points

Figure 2.12: Map result presented in the work developed by Maciel [Mac12]

and present an aesthetic schematic representation, while preserving the map's topology. However, this solution does not consider many constraints of those spider maps: lines only follow 0, 45 or 90 degree orientations, merge stations (nodes) typically have close names and share the same geographic space and routes that share the same segment are parallel. Furthermore, there is no explanation on how the initial graph is obtained and hub integration is also not considered. Finally, results are linked with parametrisations that are manually provided and change depending on the initial graph.

Finally, in [Mac12] an interactive application was developed that enable the user to select the area of the hub and then the map was produced automatically. Even tough the resultant map could not be designated as a strict spider map representation, it provides a starting point for adapting automatic generation of spider maps with interaction techniques. Figure 2.12 depicts a result presented in the work developed by Maciel.

2.5 Human-Map Interaction

2.5.1 Human-Computer interaction

Human-Map interaction on digital devices is fundamentally based on and can apply the same concepts and techniques studied and developed in the field of human-computer interaction. Hence, to better understand the process of interacting with maps, it is essential to study this field and how interaction design can be applied.

Human-computer Interaction (HCI) is defined as an interdisciplinary subject that studies the relationship between people and computer systems and platforms they interact daily [Lov05]. Thus, it is important to comprehend the user's abilities and expectations and how they should be considered when developing and designing mobile systems or applications. This is a multi-disciplinary field of study integrating knowledge from several disciplines, such as Psychology, Computer Science, Sociology and Design as shown in Figure 2.13 [PRS+94].

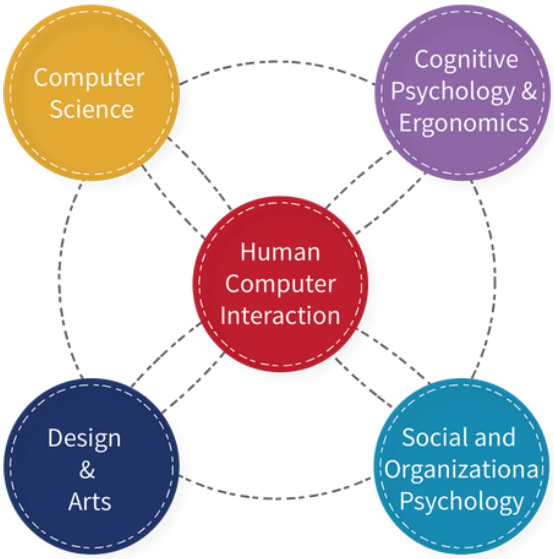

Figure 2.13: Human-Computer Interaction as a multidisciplinary area [UXL19]

Understanding the relationship between humans and devices or applications can be a difficult process, however, there are three main steps to follow when studying HCI for a system development that can ease such task [Lov05]. First, is to identify the target group of users. Second, is to identify what and how user's characteristics will affect the design and performance. Third and final step, is the development of the system that suits the needs and constraints identified in the previous steps. Moreover, it is also essential to understand the environment in which the application or system will operate (context of use), since it has a major impact on the user's ability to interact in an effective, efficient and satisfying manner with the application.

2.5.1.1 Characterising Users

HCI was described as a multidisciplinary area, that integrate several academic subjects. One of those academic areas is Psychology, since comprehending users, their needs and abilities is a key aspect in the design of mobile systems. There are several user characteristics that can be taken into consideration and aid in the process of understanding the human-computer relationship, such as spatial ability, memory, personality or even verbal ability [AA91].

Spatial ability is the individual capacity to deal with spatial relations and the visualisation of spatial tasks. It can be branched into three sub-factors, according to Lohman [Loh89]:

- **Spatial relation-** individual skill to solve problems dealing with mental rotations
- **Spatial orientation-** individual ability to orientate in space
- **Visualisation-** individual capability to visualise the current task

These factors have a relevant impact on HCI, specially on mobile devices with smaller screens than personal computers. Thus, limiting the amount of information than be displayed and requiring

more levels to complete a task, e.g., requires navigation to several screens. Users with low spatial abilities can find difficult navigating through several layers, since it is hard for them to visualise the complete task without getting confused [Lov05].

Memory is the ability to hold information for long periods of time, long-term memory, or for short periods of time, short-term memory. This user characteristic is fundamental, especially short-term memory, since it describes how users can retain information when performing a certain task [Lov05]. Thus, users with lower short-term memory may find difficult to navigate through several layers, retaining the information in each screen, to complete a task[AA91].

Another user factor is **personality** [Lov05]. Early research on the topic concluded that personality did not affect how people interact and perceive systems, however, recent studies show that this is not true, and people preferred interacting with systems whose personality was perceived closed to their own. Furthermore, users tend to attribute some human characteristics to their mobile devices (*anthropomorphic tendencies*) and establish some emotional connection with their devices.

Finally, **verbal ability** should also be considered, since it is concerned with individual ability of communicating and understanding written and spoken words [Hun78]. Thus, it has major impact on how information can be transmitted from the computer systems to its users, and vice-versa. Moreover, the field of sociology can bring additional insights to HCI, since research shows the impact of mobile technologies differs depending on social situations, for instance, using a mobile phone in a restaurant is considered as being rude, however it is acceptable in many other situations [AA91].

The target user group of a given system is heterogenous, thus, it is important to identify all the needs of all user groups and considered them in the design process, as they may need special attention, for instance, elderly people or people with disabilities [PRS+94]. Another aspect to consider is personalisation, since it opens the possibility of offering adapted systems to each user, tailored to their needs and preferences [Lov05]. This subject is further discussed in Section 2.5.3, approaching its adaptation to geographical digital systems and applications.

2.5.1.2 Human-centered design

According to the International Standards Organisation (ISO)[BCE+16], usability is *"the extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use"*. Efficiency is how the system supports users in carrying out their tasks, while effectiveness measures how good a system is for the desired task, and satisfaction presents the user's perception of the system and how easy it is to learn and use it [Lov05].

Furthermore, several researchers state that the best development process to achieve high usability is Human-Centred Design (HCD) [Lov05]. Also designated as user-centred design, it is described by ISO as a development and design approach with the goal of making interactive systems more usable by focusing on human factors and ergonomics [BCE+16]. The ISO standard [BCE+16] defines four stages in the human-centred design process:

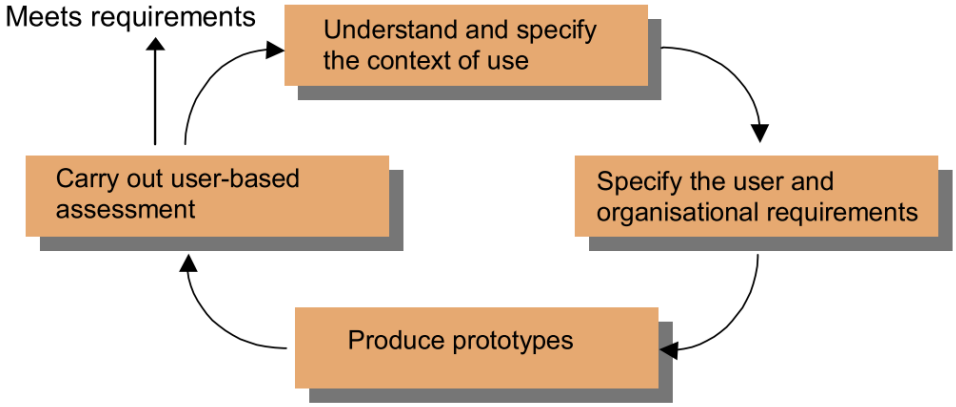

Figure 2.14: Human-Centred Design development cycle [AA99]

1. Comprehend and establish the context of use;
2. Establish organisational and user requirements;
3. Create designs and prototypes;
4. Assessment by the users.

The context of use describes the physical, social, technical and organisational conditions in which the system will operate. This will provide information about some of the system's constraints and what technologies and design techniques to use. Moreover, establishing requirements is an important step since it captures the characteristics and needs that the users and the organisation expect from the system [Lov05].

The third step relates to the creation of designs and prototypes, and since there are several different designs that can meet a system's specification, is important to evaluate them by gathered assessments from the users. Designs are created to ensure that the system matches the requirements, then prototypes are developed from these designs to get feedback [PRS+94]. Computer-based prototypes provide a global view of the system with limited functionality, enabling users to test and evaluate the system. Then, using the techniques described in the previous section, the system is evaluated, and findings are used to improve designs, making HCD an iterative (see Figure 2.14). HCD can also be applied in other areas, such as map making by following a user approach. This topic will be discussed in Section 2.5.2.

Furthermore, another important area of study in HCI is the interaction design fot mobile devices, since they have several constraints and characteristics that differ from traditional visualisation devices, such as desktop computers [KHPG11]. For instance, the limited screen size and the interaction through gestures in smartphones, lead to the creation of new HCI techniques that aim improving usability. The differences and constraints of design in mobile devices will be approached in Section 2.5.4, along with new visualisation and interaction techniques.

2.5.1.3 Research and evaluation methods

Another focus of HCI is how to develop new methods and to evaluate if the created interaction design improves usability [Lov05]. Hence, it is important to develop researches where real users can provide feedback about the systems and applications. There are two types of research studies: the ones conducted on laboratory environment and the ones on a field setting [Rub94]. Both types of studies use volunteers to test the developed system under certain environments.

Research studies conducted on laboratory, as the name suggest, are set on a laboratory control environment to test a specific hypothesis. On the other hand, field experiments do not offer control over the environment setting and variables. However, they can be very useful as they test application and systems dynamic social environment, such as public transports or restaurants, that are closer to the real usage settings.

Furthermore, environment variables can be defined as dependent or independent. The independent variable is the one usually manipulated during the experiment, for instance, testing several navigation styles. Dependent variables are often considered as the result of the dependent variable, e.g., the reaction of users towards a navigation style [Rub94].

There are several experiment designs available that assist the test of different goals. However, independently of the technique used, is important to understand if there is some bias in the result, i.e., if the participants in the test had some type of pre-formed judgment about the application or some factor being tested. Hence, two validity measures are introduced [Lov05], to ensure that results reflect the reality. First is the internal validity that aims at ensuring that results obtained are interpreted correctly and they are not evaluating in favour of the proposed hypothesis. External validity ensures the validity of the generalisations being made from the findings.

Another important topic for research and evaluation is data gathering methods. A most common method used in HCI is questionnaires [Lov05], where factual data about participants experience with the technology being evaluated can be collected. The most important aspect of a questionnaire is the questions depicted, since they must ensure the gathering of useful information regarding the established objectives.

Along with questionnaires, interviews can be a quick and efficient way to obtain information about usability issues, requirements gathering and other important feedback. Interviews can be performed in various ways: informal, semi-formal, semi-structure or structure [Lov05]. The informal interviews do not have a structure and aim at collecting as much data as possible. Semi-formal interviews are performed after gathering and analysing some experiment data. For instance, experiment results can show that users had difficulties completing a certain task. The goal is to cover the topics that resulted from the initial analysis and observations, without having a strict interview structure. A focus group can be used to conduct such type of interviews, where participants of the experiment are brought together so specific issues could be discussed. On the other hand, semi-structured interviews have an established number of questions that should be presented to the users in the same other. However, the goal is to obtain as much information as possible, so the interviewer should encourage participants to give information using follow up questions. In a

structure interview, there is an established set of questions which have an establish set of answers.

Usability testing is an important method to collect information about the performance of users while completing a certain task. The goal is to assess how users perceive the system during the task. Moreover, a cognitive walkthrough is a usability evaluation performed by an expert, assessing each step a user must take to perform a certain task. Hence, the goal of the evaluator is to walkthrough each step and verify if user's needs are met by the system [Lov05].

2.5.2 User-centred maps

Technology development led to new ways of navigation and wayfinding, with systems offering personalised services and user-centred perspectives. Modern GIS uses digital information from multiple sources, some combining Global Position System (GPS) originating Real Time GIS [Mou15].

Humans perceive maps differently and many can find the map reading and wayfinding a difficult process. However, the manner maps are represented can help facilitate these processes. For example, since Ptolemy, maps are usually represented with the north-up, which creates the need for users travelling south mentally rotate the map to correctly navigate. These mental transformations can be difficult and depend solely on the mental and spatial abilities of the individuals. Representing the map in a egocentric, or user-centred, perspective has advantages relative to an exocentric perspective. While an egocentric perspective presents a close representation of the world as seen by the user, an exocentric perspective is typically an exterior view, similar to a bird's-eye [PP08].

Porathe [Por07] demonstrate this in a simple experiment, where users needed to navigate through a six by six meters square using four types of maps: digital static north-up map, digital dynamic head-up map, digital dynamic 3-D map and a print map. Scores were given relating to the amount of time needed to complete navigation and the amount of errors made (going through obstacles). Results showed that users navigated quicker and with less errors using 3-D maps and users also found them easier to use. On the other hand, print maps had the worst results and were found the most difficult to use. Thus, experiment results suggest that maps in an egocentric perspective are more efficient and user-friendly than maps displayed in an exocentric perspective. Nonetheless, this experiment aimed at testing route guidance and not route planning, thus not provide information whether an exocentric perspective may be preferable in the latter cases.

2.5.3 Personalised Maps

The explosion of geographical information online associated with the continuous development of new technologies and data mining techniques opened a new possibility for maps: rather than producing a single map for a great number of people, adapt profiling techniques to create personalised maps for each user.

GIS enable the creation of different representations of the same input data with the goal of understanding different phenomena. Generating different maps for the same area to represent

alternative perspectives by selecting which and how to present information and by using different graphic and symbolic conventions is now common practice. Considering that every person perceives and creates different mental models of the world, generating alternative representations tailored to suit specific tasks or individuals may help with perceptions problems[BB15]. Some studies show that around 60% of adults in the United States of America own a smartphone even a higher distribution is found around young adults, with results surrounding 85% [Mac12]. Hence, creating technological systems with a higher usability for wayfinding and public transport navigation would bring benefits for travellers.

In principle, all digital maps elements can be personalised to increase usability, efficiency and clarity. Such personalisation requires that user information can be obtained through implicit or explicit feedback. Explicit feedback is given by the user in a conscious manner, i.e., by any action that inputs preferential data about the user, for instance, changing the default preferences of an interface. On the other hand, implicit feedback includes any data about the user that expresses a preference indirectly, e.g., movements of the mouse, search queries or even the user's location. Research studies show that personalisation can be achieve by integrating two complementary strands: adaptation and recommendation [BB15].

Nevertheless, such a system requires a coherent conceptual framework (such as the presented in Figure 2.15) that can adapt itself to the user in real time and offer recommendations that fit the user's profile. Hence, a user model, that reflects the user, must be created that will serve as input for the personalisation model. The models should be able to adapt itself to new inputs and changes in the environments, resulting in an integrated combination of all components to produce user tailored maps. For instance, image a system that knows implicitly that a user is interested in football stadiums and has been in a city before, so it adapts the results to better suit these restrictions. Moreover, the system it also aware that lunch hour is arriving so it suggests some restaurants according to the user's taste in an area that the user never visited but may enjoy. These kinds of system fully integrate a user model, have time and spatial awareness, adapting to its user and generating recommendations that fit but may also expand the user's experience [BB15].

However, such levels of personalisation yield several computational challenges. First problem is the real-time detection and prediction, where systems need to apply machine learning techniques to detect and predict in real time what is the current user's task. Another problem is the user's spatial modelling, i.e., personalisation requires a deep understanding of the user's behaviour and context in space and time. Nonetheless, recording, storing and mining data to develop user models is still a challenge. The following concern is related to geo-semantics (semantics applied in Geosciences and Geographic Information Science [JSA13]) and data fusion, since such complex system requires the aggregation of heterogeneous data sources in a consistent manner. This problem may benefit from the research on web semantics, as it can be applied for geo-semantics. Alongside follows the geoparsing and sentiment analysis. Geoparsing is the extraction of geographical information from natural language [GB13]. This is an important technique, as a vast amount of geographical knowledge is express in natural language. Moreover, the detection and analysis of sentiments may also be a relevant factor for extracting and modelling the user's opinion

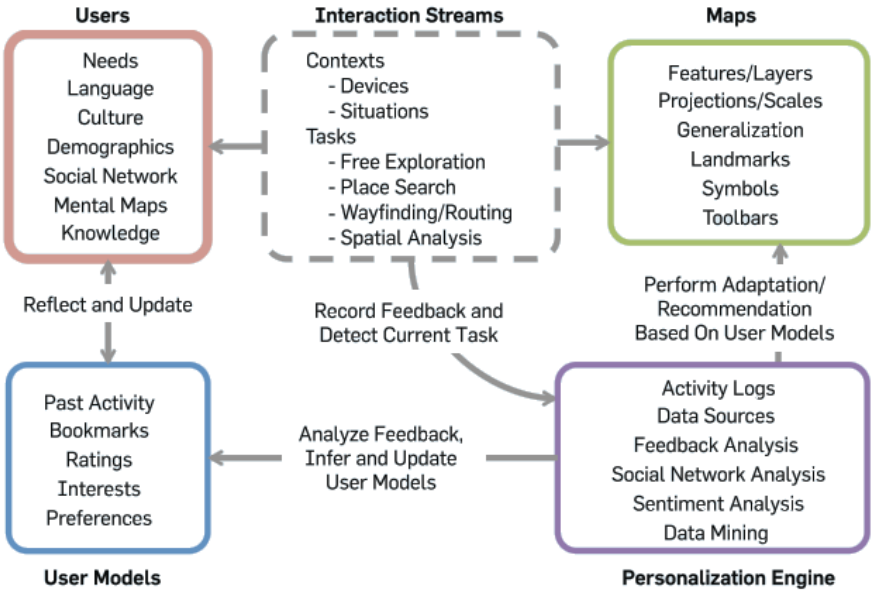

Figure 2.15: Conceptual framework for personalised maps [BB15]

about places. Finally, there is the challenge of cognitive map design. Alternative cartography representations can be validated using different concepts and principles from cognitive map design, where spatial cognition knowledge could be used for producing better personalisation and gain inside into human perception about the geographical environment [BB15].

This kind of complex map personalisation is still in very early stage, nonetheless there are several other concerns that relate to this problem. The main concern relates to user privacy, since this kind of personalised system relies on several surveillance-based techniques that collect maybe thousand of user related data. Also, there is the concern that personalised maps may lead to what Eli Pariser called a *"filter bubble"*, increasing cultural and social segregation between groups of users [Par11]. There is also the concern for the lack of transparency of the products. For instance, Google Maps uses black box models and personalisation cannot be turned off by users. Lastly, is the research concern of real data. Major companies, such as Google and Microsoft, have the advantage of having access to real user data, however most academic research is tested using *mock data*, which poses as a setback [BB15].

The transition from print static maps to digital maps enable GIS to overcome some of the print map limitations, nonetheless the fundamental problems of cartography remained unchanged. Geographical data is complex and dynamic and needs to be selected and processed using, for instance, scales, projections and generalisation techniques. Moreover, digital maps need to consider human-computer interaction and, when considering mobile computing, spatial information overload problems may arise, i.e., displaying too much information on the map [BB15].

2.5.4 Maps on mobile devices

Mobile systems introduced new ways for providing information by integrating location-based services and increasing the availability of services, which created the possibility of adapting contexts to the user's needs [HHP03]. Visualisations can aid problem understanding and users often rely on them to take better decision in less time. However, is not possible to approach mobile visualisation using the same techniques applied in desktop computer systems, since mobile devices have other constraints [KHPG11].

The biggest challenge of mobile devices is the screen size. Hence, displayed information needs to be limited, specifically in the case of maps, only few map objects and information can be visualised on a certain view [HHP03]. For instance, the position of labels in schematic maps need to be considered and certain routes can be partially display or simplified.

There are several differences between traditional visualisation context and mobile visualisation [KHPG11]. Additionally to screen size limitations, screen ratios are also different, mobile devices are less powerful and input methods differ from computers, e.g., mouse and keyboard are replaced with touch gestures. Thus, visualisation applications developed for computers typically do not perform well on mobile devices. Furthermore, mobile devices have an external factor, since they are sensitive to environment conditions, e.g., the signal quality.

The same concepts described in Human-Computer Interaction section (Section 2.5.1) can be applied on mobile systems design: achieve usability by applying user-centred development [HHP03]. Applying generic heuristics is imprecise, since they are abstract and do not represent the specific guidelines or complex aspects and interactions of the user interface. Even though a restrict set of general rules does not exist, it is important to ensure that applications and systems provide good visualisation and interaction, since undisciplined design may lead to certain failures and lack of usability [KHPG11].

The main task related to map applications is scroll and zoom, as it allows the user to explore different parts of the visualisation. However, since the full view usually does not fit the screen, users need to use these actions to obtain the global spatial context which is cognitively complex and forces user to lose the global context when visualising a detailed scale [KHPG11]. Thus, two different types methods were designed considering their applicability in the mobile context.

First type is designated *Overview+Detail* where two different views, one for context and another for detail, are provided simultaneously. For instance, a simple example of design that follows this approach is showing the context view occupying all the screen and a small rectangle in the bottom with the detailed view of the selected area.

On the other hand, *Focus+Context* approach establishes context and detail simultaneously within the same view. The best know example is *Fish-eye* views that magnify the objects in the point of greater focal attention and progressively decreases the size with distance.

In [HHP03], Heidmann et al. performed an experiment with volunteer users with the goal of identifying good design techniques for designing map mobile applications and establish good interaction techniques. The experiment consisted on two major phases: first, questioning and

interviewing potential users for gathering system requirements, and a second phase, a usability test, where users evaluated a prototype. Furthermore, the second phase had also two iterations. The first iteration involved 15 participants that, after being introduced to the basic concepts of the prototype usage, were asked to perform several tasks related to the navigation with different map views, changing displayed information and manipulating map objects. The goal of these tasks aimed at testing the usability of hidden layers and how symbols can be used to express words. The second phase had 11 participants and, similar to the first phase, volunteers were asked to complete certain tasks using the improved system that resulted from phase one feedback. The developed prototype was based on HTML technologies and tests were performed on a device similar to a Compaq ipaq Personal Digital Assistant (PDA).

From the described experiment, the authors in [HHP03] were able to demonstrate the advantages and disadvantages of the proposed design techniques. For instance, the general solution for not clutter maps with labels is to only give an abstract overview in small scales and offer a more detailed view in bigger scales, which can be complex. Other technique is to hide labels in tooltips that pop up when a map object is selected. Experiment results demonstrate that hidden labels are a good solution for some tasks and were easily adopter by the users. However, in tasks where a general overview of the spatial configuration was required this technique lack usability.

Another investigated technique was the application of abstraction and simplification methods. For instance, in this experiment a simple layout of a building was showed, and a tour path displayed in the screen. In small scale, instead of showing detailed route within hall, it was only showed the sequence of halls and users could retrieve more information by clicking on the symbols that represented each wall. This method was easily understood by the users.

Furthermore, it was concluded that is important that not only the user is able to visualise the spatial data but also should be able to browse, manipulate or interact with it. This can achieve by enabling selection of certain objects to retrieve more information or choose which objects to display on the map by a list selection. However, this interaction must be provided such that it is intuitive for the user to perform this type of actions. The final technique evaluated was performing zooming through a slider control. The concern with this method was how to eliminate the inconsistencies of changing the scale, and how to do it without confusing the users. In this experiment, the participants generally neglected the use of zoom to obtain more information, instead preferred clicking on map object.

Even though this experiment was performed on PDA devices, and nowadays mobile devices offer bigger screens with more resolution and capable of dealing with multi-touch, this experiment demonstrated some useful results in using certain techniques. However, it does not consider new possibilities that some recent devices provide, for instance, zoom can be performed by gliding two fingers, which can be more efficient than using a slider control.

2.5.5 Interaction on mobile devices

Current mobile device introduced a new form of interaction: gesture-based interaction. A gesture is considered a movement of one or more fingers on the screen, hence this new method of

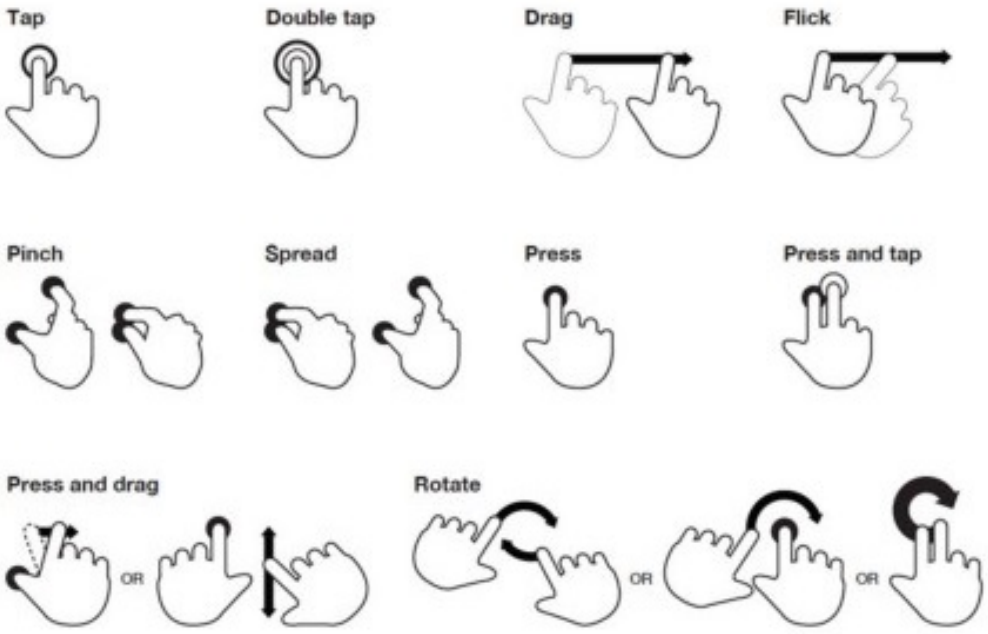

Figure 2.16: Gestures on multi-touch devices [CAR19]

interaction uses gestures as its method of user input [KRR13].

Smartphones provide a new set of interaction possibilities by introducing multi-touch and several gestures [MD16]. Nowadays, these mobile devices have larger screens than previous devices that enabled more complex interactions.

However, the introduction of complex gestures conducted to new design concerns [KRR13]. For instance, screen occlusion of some area when performing some gesture may affect the task being performed. Also, using a finger to perform a precise gesture can be difficult and, when the screen is big, gestures between extremities may be exhausting to the user. A list of the most common gestures used in devices with multi-touch screen is depicted in Figure 2.16.

2.6 Conclusions

Transportation maps are an important support for public transport, providing passengers an easy way of understanding the underlying network and wayfinding in cities. Spider maps are a type of transportation map that combine elements from both geographic and schematics maps. They provide all the travel possibilities of an area and are typically used in busy city centre where the networks are usually denser.

Even though these maps have great potential for providing public transport information, they are not widely used in comparison with other types of transportation maps. This may be due to the fact that spider maps are mostly manually generated and still rely on the expertise of the designer

and stakeholders. Notwithstanding, there are several techniques applied to line generalisation in schematic maps that can be adjusted to assist the spider map generation process.

Moreover, there is not much literature focused on spider maps, and the majority of the efforts made for automating the spider map generation process were done by João Mourinho [Mou15]. Mourinho's solution is able to successfully produce spider maps, however, the goals focus on the quality over performance, making this a complex solution with great computational effort.

Nowadays the use of digital devices is widespread, which makes it fundamental to study how humans interact with such devices. Human-Map interaction on digital devices is fundamentally based on and can apply the same concepts and techniques studied and developed in the field of human-computer interaction. Hence, to better understand the process of interacting with maps, it is essential to study this field and how interaction design can be applied. Smartphones with touch screens and location-based services opened new possibilities for interaction, however new concerns and limitations arise caused by the smaller screens that these devices tend to have in comparison with traditional personal computers.

Chapter 3

Designing a solution for generating interactive spider maps

Spider maps have great potential for providing public transport information, since they present all travelling possibilities of an area and provide better spatial awareness than other transportation maps by combining elements from both geographical and schematic maps [MD16].

However, these maps are still not widely used in comparison with other types of transportation maps. For instance, the traditional metro schematic maps. This may result from the complexity of the generation process and the fact that they are still done manually, relying on the expertise of the designer [Mou15]. Therefore, there is the ambition of developing a solution that can automate the spider map generation process and thus make an impact on the use of spider maps for proving public transports information.

This chapter depicts the development of a solution that approaches the automatic generation of spider maps. Section 3.1 describes the proposed problem and defines the challenges and goals of this project. On the other hand, Section 3.2 depicts the development of the algorithm to automatically generate spider maps, while Section 3.3 describes the prototype development which integrates the algorithm with map interaction. Lastly, Section 3.4 evaluates the developed solution and conclusions are presented in Section 3.5.

3.1 Problem definition

The spider map generation process is a complex problem, since these maps have several design constraints as depicted in Section 2.4. Additionally, the process is mostly done manually, relying on the expertise of the map maker. Even though some current solutions can automatically generate spider maps, they are complex and time expensive for producing results.

Thus, the objective of the proposed solution is to develop an algorithm capable of producing a spider map by creating, adapting and modifying techniques. The solution must take as input

the spider map hub area selected by the user and generate as result a viable spider map. A result is considered viable if it satisfies the design restrictions of spider maps aforementioned in Section 2.4. The goal is to develop a prototype that integrates the developed algorithm capable of producing spider map results in short execution times, since it will affect the prototype usability.

Along with automating the spider map generation process, the prototype should also integrate interaction and visualisation techniques, taking advantage of the benefits of digital maps over the traditional form and thus potentially achieve better usability and learnability. Such techniques can be integrated before generating the map, for instance, during the hub selection process, and when visualising the map result, e.g. different levels of zoom and clickable items for additional information.

The developed prototype is focused on Porto city and all the public transport data was provided by OPT[1]. The user is presented a geographic map of Porto for choosing the hub area that will be used as input for generating the spider map. The target is to produce a viable map that can be aesthetically improved in a post-processing phase.

3.2 Map generation algorithm

The algorithm comprises a sequence of steps that apply displacement operations, to ensure conformance to the spider map restrictions. The major restrictions are octilinear angles and maintain the topological relations, hence the biggest challenge of the algorithm is to find a location for every map point that ensures octi-linearity, while maintaining the topological relations between points.

Beforehand, the algorithm needs as input the coordinates of the hub, defined by the top left and bottom right corners. These geographical coordinates, i.e., a set of latitude and longitude, will allow querying the server for all the data needed for the spider map generation process. Thus, the server will provide all information related to stops inside the hub and the lines that will belong to the spider map. The lines are defined by a sequence of map points, already established by the database and these will be the points taken into consideration in the algorithm. The goal of the algorithm is to find a location for all map points so that lines follow the spider map design constraints.

Nonetheless, the map points returned from the server are defined by geographic coordinates of their accurate location. Hence, map points need to be projected onto the map canvas, being defined by an x and y instead of latitude and longitude. The map canvas is defined as an SVG using the *D3.js* tool and map points coordinates are projected using the Mercator projection centred at the hub centre. The projection process uses the *D3.js Geo* plugin that provides map projection features.

At this stage, map points have a similar location to their geographic representation, so lines also follow an approximation to the real geographic form. The next step is to insert the hub, defined by the four corners coordinates that represent the boundary area. These coordinates have also been projected so they are defined by an x and y in the canvas. After the hub insertion, the

[1] http://www.opt.pt/

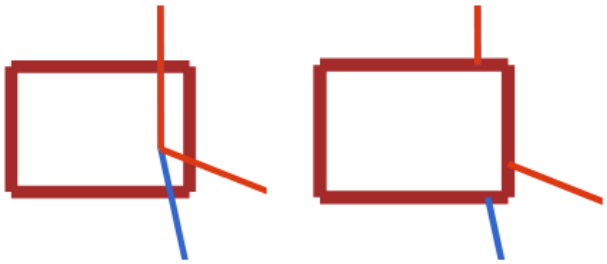

Figure 3.1: Determine where lines should emerge from the hub

following step is to determine where lines should emerge from the hub and eliminate line segments inside the hub. Therefore, the emerging points, i.e., the points where lines emerge from the hub, will be determined by the intersection of the line segment with the hub. This intersection point represents an approximation of the orientation and path of the line, since the hub is a geographical representation of the area and map points are still located at their original positions, i.e., a close representation of the geographic location. Additionally, all segments positioned inside the hub are eliminated. Figure 3.1 illustrates this process: the left image exemplifies the map state after hub insertion and the right image demonstrates the calculation of the emerging points and elimination of segments inside the hub.

Map points can be shared by multiple lines and lines can even share segments, thus duplicated information may exist. Hence, the spider map is modelled as a graph *G(V,E)*, where *V* represents the vertexes, i.e. the map points, and *E* the edges, i.e. route segments that represent the connection between two map points. Each vertex and edge may belong to one or multiple lines, thus avoiding having duplicated map points or segments. Vertexes have *x* and *y* coordinates, a list of lines they belong to, a name representative of the stop or area and an attribute that records if the vertex is an emerging point. It is important to register which vertexes are hub emerging points, since they should not be moved during the algorithm process. On the other hand, edges have two vertexes associated and a list of lines. The order of the map points in the lines is also recorded as well as the colour that lines should be drawn.

Spider maps have associated distortion, since their points do not represent geographical locations, but the product of multiple operations. Furthermore, dimensions are altered, i.e., the hub area is usually augmented and distances between map points are reduced, resulting on a compression effect centred on the hub. However, at this point in the algorithm process, the map dimensions still resemble the real dimensions: the hub is small, and lines are very spread out. Hence, the next step is to resize the hub, increasing its dimensions and translating the lines accordingly. The result of this operation causes the distortion and compression effect aforementioned.

The resized hub dimension was set to 300 by 300 hundred pixels, however, hub selection may not follow this aspect ratio. Hence, the final size of the hub is recalculated so the original aspect ratio is preserved. For instance, if the original hub width is 200 pixels and the original height 100 pixels, the resized hub will have 600 pixels of width and 150 pixels of height.

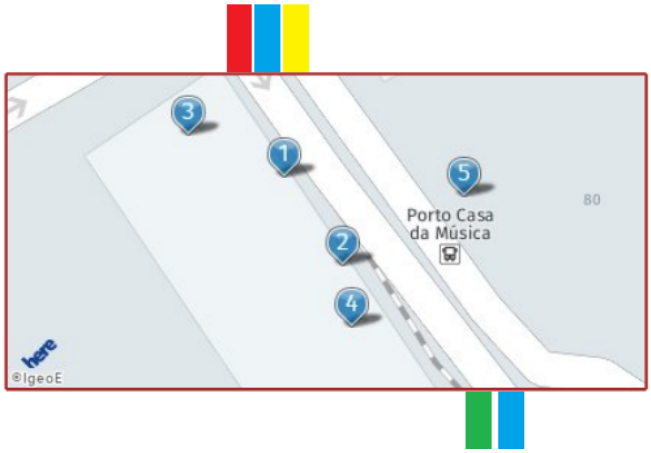

Figure 3.2: Hub of Casa da Música area. Lines emerge from the hub at an approximate location of their trajectory

Nevertheless, after resizing and translating operations, some of the lines may end up intersecting the hub. Thus, map points inside the hub are identified and the maximum distance to the hub boundary is calculated. Then a translation operation corresponding to this distance value is applied, pushing the line out of the hub. At this stage, the map is similar to the original, but with distortion and with lines closer to the hub boundaries.

The hub is a portion of a geographic map that depicts the area associated with the spider map. To obtain the geographic map image Here API[2] was used, providing services that return a map image of the specified area. The image already has the correct size, i.e., same size as the hub, thus it is placed on the hub coordinates. Figure 3.2 illustrates an example of a hub of Casa da Música surroundings. The markers represent the stops in that area.

Furthermore, before beginning the displacement operations to satisfy the spider map restrictions, a matrix containing the topological relations between points is built. It is important to build the matrix before the generation process starts, since at this stage all the points relate to each other close to their real geographical location. Thus, for each map point is calculated the relation to every other map point. A map point can be north of (No) or south of (So) and east of (Eo) or west of (Wo) another point. When two points have an equal coordinate (x or y), they are defined as *in line* of each other. Table 3.1 depicts the topological matrix of points P1, P2 and P3 relations.

After this step is completed, the displacement operations begin in order to find a location for every map point that satisfies spider maps restrictions. First, a grid adaptation operation is done with the intent of simplifying the overall shape of lines. Next, all angles are ensured to be octilinear and then the spider map is displayed, following the draw rules. The next sections depict these algorithm steps, that will displace map points trying to generate a viable map solution.

[2]https://developer.here.com/

Table 3.1: Example of topological relation matrix between points P1, P2 and P3

-	**P1**	**P2**	**P3**
P1	-	inLine, Wo	No, Wo
P2	inLine, Eo	-	No, inLine
P3	So, Eo	So, inLine	-

3.2.1 Grid adaption

The first step of the algorithm is to adapt the current map to a predefined grid, by assigning a grid intersection point to every map point. This step simplifies the overall shape of lines, leading to a closer solution where spider map restrictions are followed.

The first task is to build the grid over the map, thus the maximum and minimum x and y values of the map points are determined which represent the bounds of the map and, subsequently, the boundaries of the grid. Then, the grid is built with an initial grid cell size of 20 pixels by 20 pixels. It is important to note that the cell size will affect the complexity of the algorithm, since grid cells with smaller cell size lead to finer grid granularity, which increases the search for possible displacements. On the other hand, it may not be possible to adapt a map to a grid if cells have a large size, given that possible displacements will be scarce. After several tests with different sizes, this initial cell size was chosen since results showed that most maps could successfully adapt to a grid with this cell size, without the need of repeating the process by adapting the cell size. The final step in building the grid is to determine the grid intersection points. These points represent all the possible displacement for map points during the grid adaptation process and the subsequent algorithm steps.

Therefore, for every map point the nearest grid intersection points are calculated, i.e., the 16 surrounding and closest grid intersection points are determine. However, not every nearest grid intersection point is a valid displacement. A grid intersection point is considered for a valid displacement if it causes no hub occlusions, i.e., does not cause any segments to intersect with the hub; does not lead to any segment overlapping, i.e., segments do not pass through map points that do not belong to that segment; and the grid intersection point is free, i.e., it does not have a map point assigned.

The grid intersection point selected for the displacement is the one with the smallest score, which represents the attribution of less penalties. The score combines the distance from the map point to the grid intersection point being evaluated (points with greater distance will be more penalised) and a score that translates how well topology relations are maintained, by giving a penalty to every topological violation. A displacement causes a topological violation if it changes the relation between two points. For instance, having P1 south of and east of P2, a displacement that leads to P1 being north of or west of P2 is considered to cause a topological violation. Smaller penalties are given to displacements that cause relations to change to in line. This trade off by loosening the topological constraints leads to simpler line shapes, where lines become straighter which causes less non-octilinear angles. If no topological violation occurs, then the topological

Figure 3.3: Line shape before and after the grid adaptation process

score given is zero.

However, in some cases it may not be possible to adapt the map to the grid with the current grid cell size. This means that some map points may not have any possible valid displacements. Hence, the grid cell size is decreased, the map points coordinates are restored to their original locations and the grid adaptation process is restarted. Decreasing the cell size leads to a finer grid granularity, which in turn leads to more possible displacements. This processed is repeated until the grid adaptation is successfully completed or the grid cell size reached a defined minimum. In this last case, the grid adaptation process may not be possible, thus a map solution will not be produced.

Figure 3.3 depicts the grid adaptation process, illustrating initial locations in the left image and the displacement result in the right image. In the figure is possible to note that lines have a simpler shape and some of the angles already comply with the octilinear angle restriction. Furthermore, this algorithm step is described in the flowchart illustrated in Figure 3.4

3.2.2 Correct non-octilinear angles

After the grid adaptation process is finished, the result is a map with simpler line shapes and where map points respect the topology relations. However, some of the lines may still not follow octilinear angles. Just as mentioned in Section 2.4, one of the spider maps restrictions is that angles should only be of 0, 45 or 90 degrees, i.e., only octilinear angles. Hence, the next algorithm step is to identify and correct non-octilinear angles.

The first step is to identify all the map points where two segments form a non-octilinear angle. Map points corresponding to hub emerging points are not taken into consideration, since they will be approached using a different method to ensure the lines also form octilinear angles when intersecting the hub boundaries.

Afterwards, for each map point identified with an incorrect angle, the algorithm will try to identify a grid intersection point which displacement will correct the angle. The process is similar to the nearest grid intersection points search in grid adaptation, where the closest grid points are identified and invalid displacements are removed from possible options. A grid intersection is considered not valid for non-octilinear angle correction if:

1. The displacement will cause octilinear angle to become non-octilinear;

2. The displacement will disturb topological relations between points (changes to in line are not considered as disturbance);

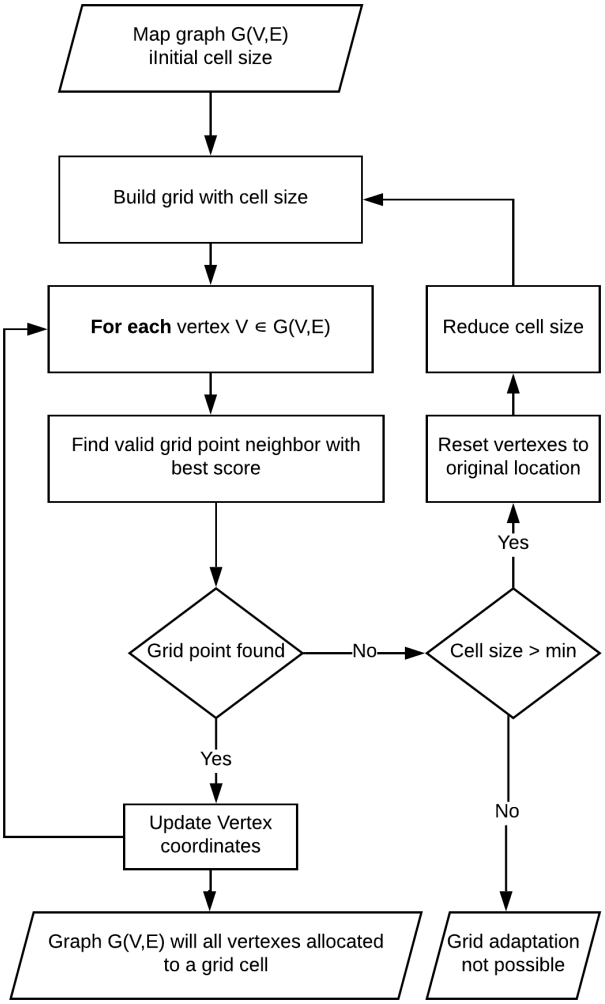

Figure 3.4: Algorithm steps and decisions for grid adaptation

3. The displacement will cause occlusions with the hub, line overlapping or lead to segments passing through map points that do not belong to that segments;

4. That grid point already was a map point associated;

5. The displacement will cause the angle to remain non-octilinear.

Just as in grid adaptation, scores are calculated for every valid option and the map point is displaced to the best scored grid intersection point, i.e., the grid intersection point with the smallest score. Figure 3.5 exemplifies the correction of a non octilinear angle by displacing a map point to another grid intersection point. Moreover, Figure 3.6 depicts how the algorithm tries to correct the identified non-octilinear angles by displacing vertexes.

Nonetheless, some map points will not have any possible valid displacements that will correct the non-octilinear angles, thus making them candidates for a break point introduction. A break

Figure 3.5: Line shape before and after the correction of a non-octilinear angle by map point displacement

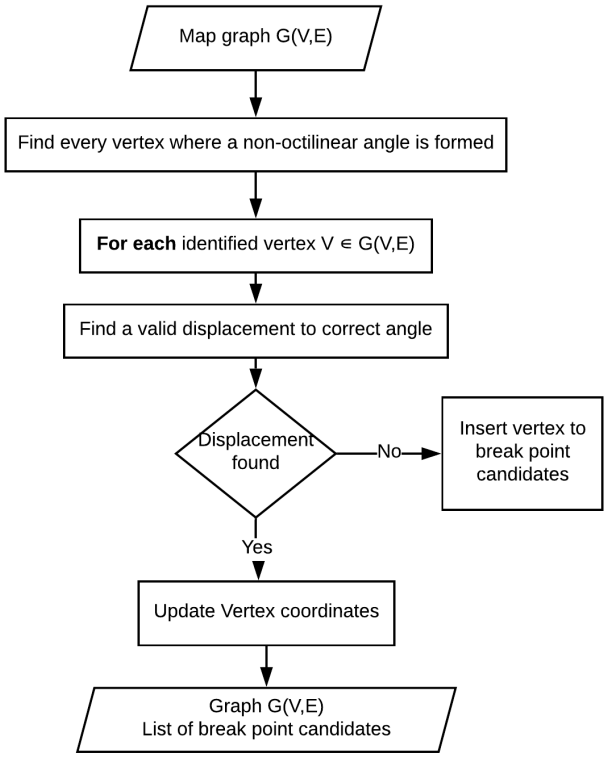

Figure 3.6: Algorithm steps and decisions for correcting a non-octilinear angle with a displacement

point is a map point introduced in one of the segments of the incorrect angle to correct the non-octilinear angle without displacing any map point. This new map point is added to the graph representation and marked as being a break point. Break points are introduced to correct angles and have no other meaning to the spider map, so they need to be represented differently.

To introduce a break point, grid intersections surrounding the identified segments are searched and checked if the displacement will correct the angle. Similar to the previous operations, a displacement is valid if causes no occlusions, and no segments overlap. A break point introduction will transform one segment in two new ones, allowing angles to be corrected. Figure 3.7 illustrates a result of a break point introduction, while Figure 3.8 depicts the algorithm steps involved in this operation.

Figure 3.7: Non-octilinear angle correction with segment break point

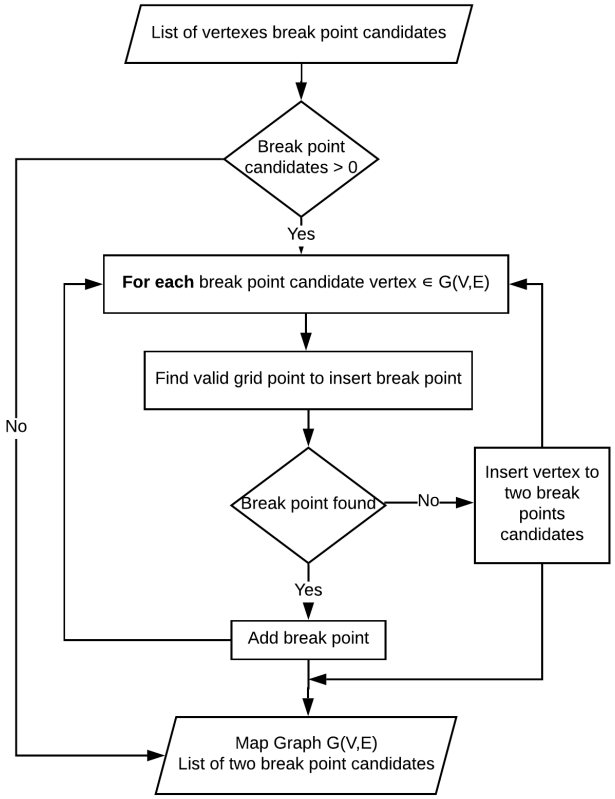

Figure 3.8: Algorithm steps for correcting a non-octilinear angle with a break point

Although introducing a break point will correct most of the remaining non-octilinear angles, in some cases, mostly in very dense areas, it is not possible to find a valid location to introduce the break point. Hence, two break points are introduced to correct these last cases. The introduction of two break points is very similar to inserting a single one. However, the goal is to find the best combination of two valid grid points that can correct the angle. The complexity of this problem is restrained by limiting the grid points search to the nearest grid points and by eliminating all the invalid grid locations. Figure 3.9 depicts non-octilinear angle correction by inserting two break points. Moreover, the algorithm steps related to this operation are depicted in Figure 3.10.

Moreover, lines emerging from the hub should also make an octilinear angle with the hub boundaries and, just as aforementioned, the correction of these angles is treated separately from the

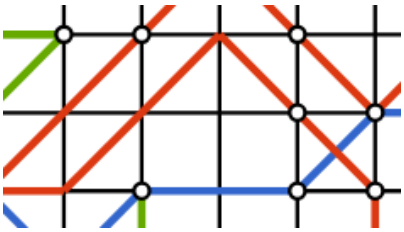

Figure 3.9: Non-octilinear angle correction with two break points

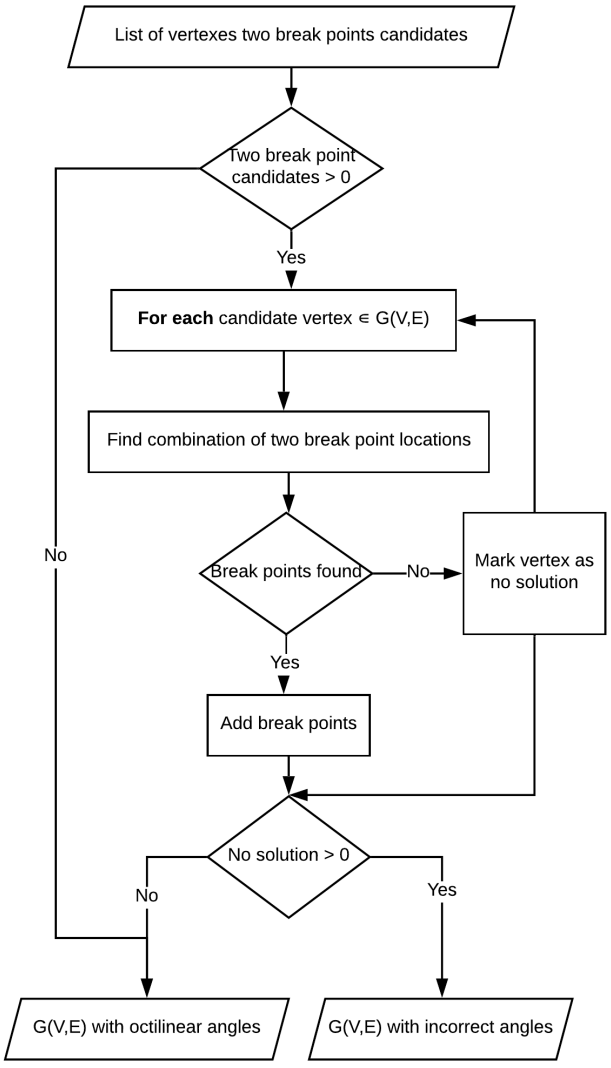

Figure 3.10: Algorithm steps for correcting a non-octilinear angle with two break points

remaining map points. To correct the angles from hub emerging segments is established that those segments should make a 90^o angle with the hub boundary. Then, a break point is introduced in that segment, so the corresponding angle is 90^o or, if not possible, 45^o degrees. Figure 3.11 depicts the

correction of angles from hub emerging segments, while Figure 3.12 depicts the algorithm step to correct the angles of emerging hub edges.

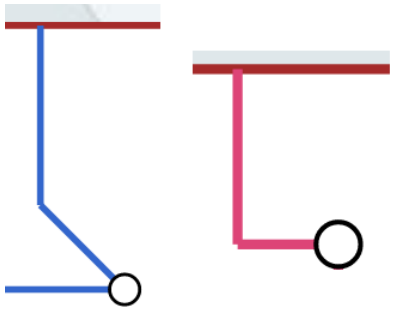

Figure 3.11: Correction of non-octilinear angles of hub emerging segments

3.2.3 Draw spider map

After correcting the angles, the final map point locations are determined, and the drawing process can begin. The hub is ready, thus the spider map drawing process will focus on drawing the lines, relevant map points (map points related to break points are not drawn) and introduce interaction capabilities in the spider map.

Map points are drawn in the associated locations and do not need further processing. However, some segments can shared between lines and need to be drawn parallel by introducing an offset. Thus, it is necessary to calculate the slope of the line and identify the correct orientation (vertical, horizontal or diagonal) so the offset is correctly added to the x and y coordinates, just as illustrated in Figure 3.13. For instance, vertical segments will only need an offset in the x coordinate, while horizontal segments will have the offset on the y coordinate.

Through *D3.js Behaviour* plugin is possible to introduce interaction to the spider map, by making gestures do zoom, navigate and click on map points to check additional information. The interaction capabilities will be depicted in Section 3.3.3.

Thereupon, the map generation algorithm can be divided into smaller steps described in the previous sections. This steps displace map point in an attempt to find a spider map solution. After the rendering process, the generation process is completed and a spider map solution is exhibited. Test and results to the developed algorithm are described in Section 3.4. Figure 3.14 depicts all the algorithm steps and Appendix A presents the algorithm pseudo-code.

3.3 Prototype development

For the purpose of testing how the developed algorithm performs in real usages, a prototype was created integrating the algorithm depicted in Section 3.2 and taking advantage of digital map characteristics by combining visualisation and interaction techniques.

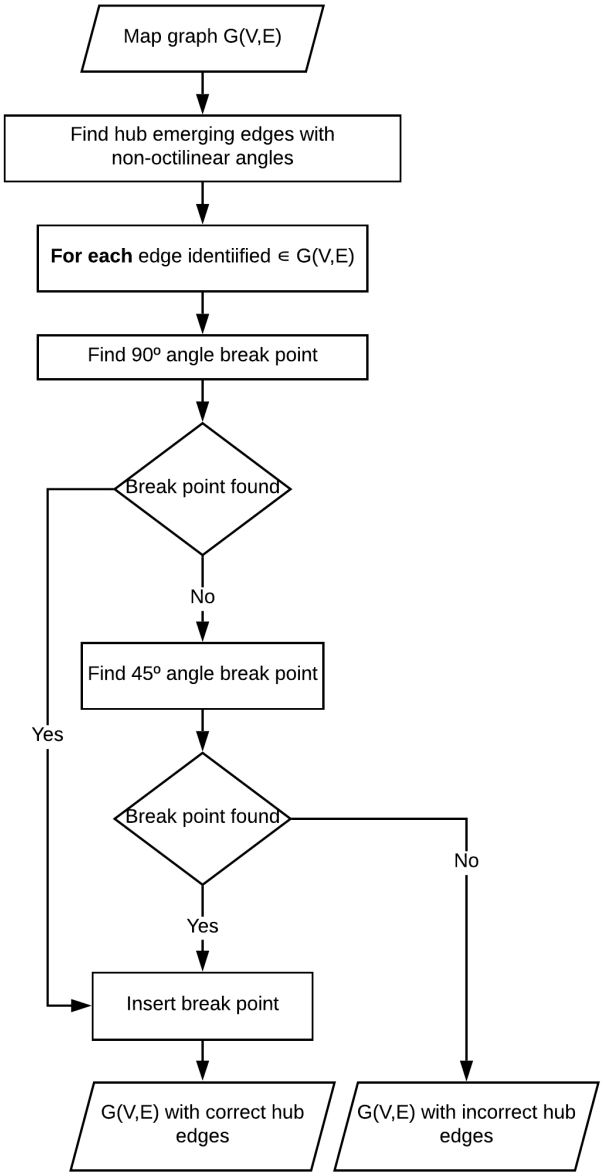

Figure 3.12: Algorithm steps for the correction of non-octilinear angles of hub emerging segments

3.3.1 Architecture

The developed prototype follows a two-tier or client-server architecture, as illustrated in Figure 3.15. This architecture style is commonly used in distributed systems to separate operations into the client and server, where the server provides services to the client [IBM19]. Hence, the prototype client is responsible for all rendering and visualisation operations and implements the algorithm, whereas the server processes all the necessary data providing services to the client through a REST API. The client and server communicate via the HTTP protocol.

The client consists of a web application with two screens that implement the two major use

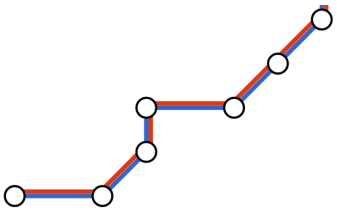

Figure 3.13: Shared line segments drawn parallel

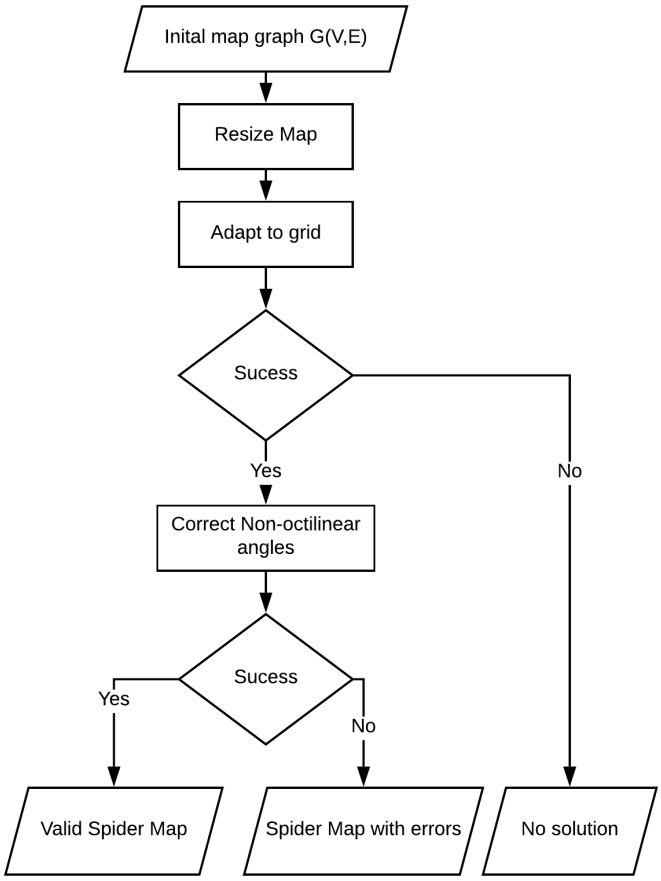

Figure 3.14: Algorithm for spider map generation

cases: select the hub area; view and interact with the spider map result. This will be depicted in more detail in the following Section 3.3.3. The algorithm was implemented using *JavaScript* and the spider map rendering operations were done using *D3.js*[3] (Data-Driven Documents) library. Furthermore, *Leaflet*[4] was used to display an interactive Porto map, enabling the user to select the hub. To retrieve a hub image from the selected coordinates, Here API was used since it provides

[3]https://d3js.org/
[4]https://leafletjs.com/

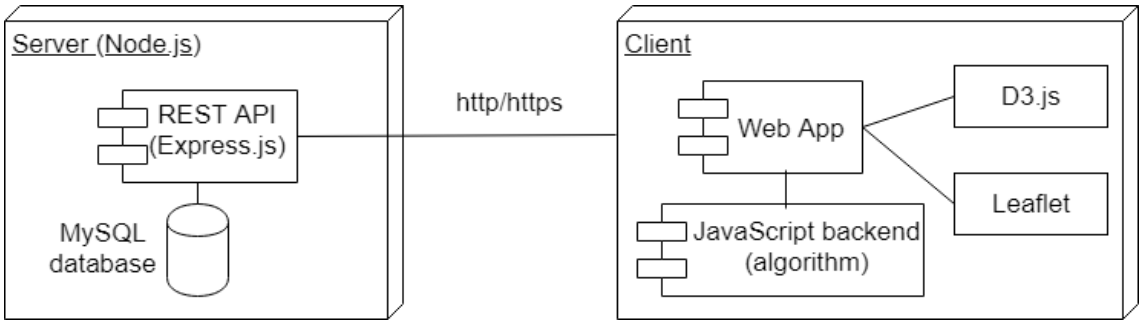

Figure 3.15: Client-Server Architecture followed by the developed prototype

endpoints that return a geographic map image defined by a boundary box, in this case, the hub.

Leaflet[5] is an open-source JavaScript library for interactive maps, chosen for being lightweight, recommended for mobile devices and because it can be extended by several available plugins that facilitate the creation of personalised interaction with maps. On the other hand, *D3.js* is also a *JavaScript* library for manipulating documents based on data, presenting a tool to display it using HTML, SVG and CSS. *D3.js* is a powerful and versatile tool that has an extended data manipulation features, such as geographical projections, scales and interaction and visualisation tools. Even though *D3.js* has a superior learning curve than other similar tool, e.g. *Cytoscape.js*[6] or *Vis.js*[7], the flexibility and numerous components make this tool favourable for the proposed prototype goals.

On the other hand, the server was implemented using *Node.js*[8] and *Express.js*[9]. *Node.js* was chosen, since it presents a reliable and simple way of setting up a server and *Express.js* is used for routing and creating a REST API. Moreover, the server establishes a connection with a *MySQL*[10] database that stores all the public transport data, that will be depicted in Section 3.3.2.

The developed REST API created using *Node.js* and *Express.js* follows a simple structure based on three components: routes, controllers and models. The models define the objects and their methods, accessing the database to retrieve the desired information. On the other hand, the controllers define the methods for accessing the models [MDN19]. Routes represent endpoints that will be associated to a controller, responsible for receiving the request and sending the answer.

The implemented API has two main models, routes and stops, with several endpoints for handling and retrieving information associated with each one of the models. Table 3.2 depicts the API endpoints and present a brief description of their use. The main use of the API is to send the information required for the spider map generation, hence the only type of operations implemented were of retrieval type.

[5]https://leafletjs.com/
[6]https://cytoscape.org/
[7]http://visjs.org/
[8]https://nodejs.org
[9]https://expressjs.com/
[10]https://www.mysql.com/

Table 3.2: API Endpoints

Endpoints	Parameters	Description
/stops	-	Get all stops
/stops/hub	topLong; topLat; bottomLong; bottomLat	Get stops inside the hub defined by the top left and bottom right corners geographic coordinates (longitude and latitude)
/stop	stopId	Get stop by ID
/stop/lines	stopId	Get routes of all of the lines that go through a stop
/line/stop	stopId	Get all lines that serve a stop
/line	lineId	Get line by ID
/line/code	code	Get line by code
/line/route	lineId	Get line route by line ID

3.3.2 Data model

All the data related to the Porto public transports network was provided by OPT and stored in a *MySQL* database modelled as Figure 3.16 depicts. The database stores all the data related to stops, lines and routes required for the spider map generation.

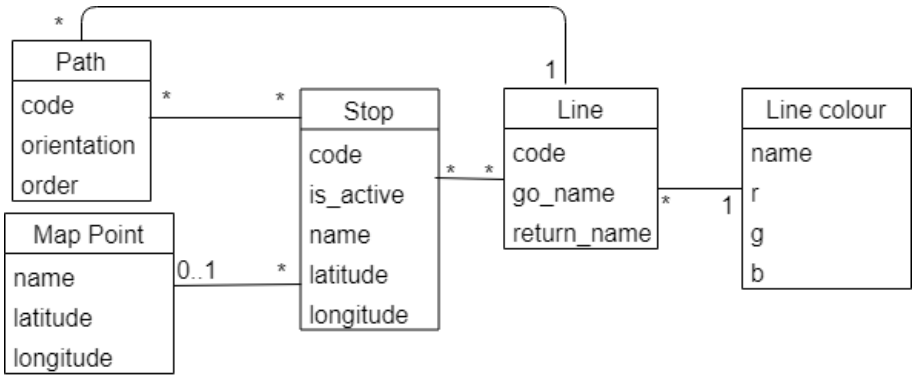

Figure 3.16: Data Model

Lines are characterised by a code, name and have a line colour associated. On the other hand, stops are also defined by a code, name and geographic coordinates (longitude and latitude). Lines are made of a set of multiple stops and each stop may belong to many lines. However, routes are defined by the table *Path* that define the sequence of stops identified by the attribute *order*. The data provided already defines map points that may represent groups of stops. Henceforward, when a stop is associated with a map point it should be replaced when forming the route of a line.

3.3.3 Use cases

The prototype aims at validating the developed algorithm and test how it performs in real uses. Hence, the main use cases are select the hub area for the algorithm and generate the spider map.

Figure 3.17 illustrates the prototype use cases. Moreover, there is the ambition to integrate interaction and visualisation techniques to enhance the usability and learnability. Also, the aforementioned use cases can correspond to several user stories, depicted on Table 3.3 that depict ways that the user can interact with the system.

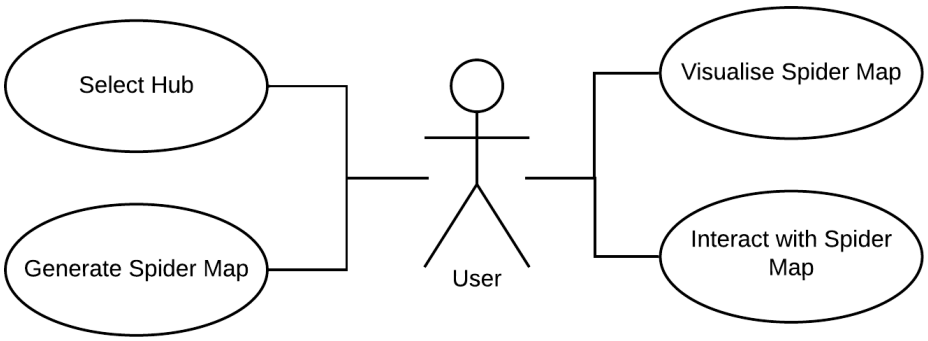

Figure 3.17: Use cases

Table 3.3: User Stories

User Story	**Description**
US01	As a User, I want to see and navigate a map of Porto city area
US02	As a User, I want to see a pre-defined grid that marks possible hub placements
US03	As a User, I want to choose one or multiple grid cells that define the hub area
US04	As a User, I want to clear current grid selections
US05	As a User, I want to check the stops inside the hub selection
US06	As a User, I want to check additional information about stops
US07	As a User, I want to generate a spider map given the hub selected in the grid
US08	As a User, I want to visualise the spider map resulted from my hub selection
US09	As a User, I want to check information about the hub stops and lines that belong to the spider map
US10	As a User, I want to interact with the spider map by zooming, moving and clicking on elements for additional information

In the first screen the user is presented a map of Porto city with interaction capabilities, i.e., the user can zoom and navigate through the map. Furthermore, in the top right corner the user can access control buttons illustrated in Figure 3.18 left. In this controls users can show/hide the pre-defined grid, check stops inside the grid selection and finally generate the spider map.

The pre-defined grid lays over the Porto city corresponding to the boundaries of the available data. Then users can select one or combine several grid cells to create a personalised hub area.

Figure 3.18 right shows an example of grid selection, where selected cells are shown in orange and markers depict stops inside the hub selection. Furthermore, the grid cells size was chosen considering the amount of stops inside the area. Larger hubs will have more stops, which increases the complexity of the spider map generation, but also spider maps with too large hub loose their meaning and usability. Moreover, a pre-defined grid was chosen over, for instance, over allowing the user to draw the hub in the map, since it controls the area where data is available and the hub size by defining a minimum (a single grid cell selection).

After the user chooses the desired hub and selects *"Generate Spider Map"*, the algorithm takes the hub coordinates as input and generates a spider map result. In the next screen the user can visualise and interact with the map result. The user can navigate, zoom and click on map points to check additional information. All these interaction features were developed using *D3.js Behaviour* plugin that allow to catch and handle interaction events. Figure 3.19 depicts an example of a portion of a spider map result where is possible to check the additional information box when a map point is hover or clicked on.

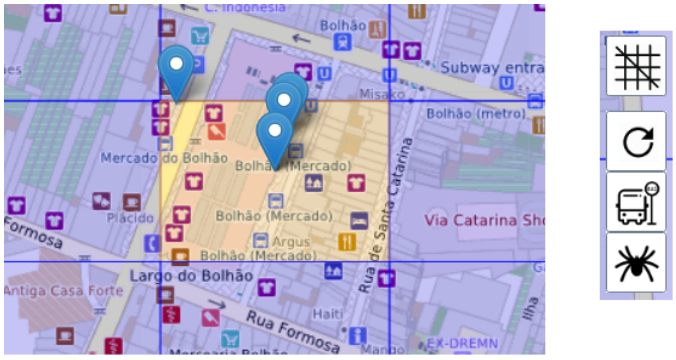

Figure 3.18: Example of grid selection (left) and map controls (right)

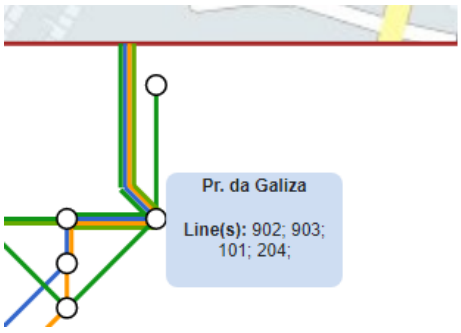

Figure 3.19: Interaction with spider map: click on map points to obtain additional information

3.4 Evaluation and Validation

Throughout this section the developed solution will be tested and results will be depicted, demonstrating the contribution to the identified gap in state-of-the-art solutions. Henceforward, spider maps are still mainly manually produced and not widely used compared to other transportation maps.

Current solutions are complex and take very long to produce spider map results. Hence, the ambition is to tackle the complexity of the generation process of spider maps and develop a solution capable of automatically generate spider maps at real-time. Thereby, the two variables taken into consideration during the evaluation and validation are if the map is correctly generated, i.e., the spider map follows the establish design rules, and the execution time needed to produce the result. A result is considered valid if it complies with the spider map restrictions aforementioned in Section 2.4.

3.4.1 Tests and results

Performed tests aim at testing if the solution is capable of generating valid spider maps at real-time, using the prototype develop to select the input hub area, generate and evaluate map results. Several tests were performed by choosing different hub areas as input and evaluating the results.

The goal is to generate spider maps at real-time and thus simplify the generation process of these maps. Contrary to, for instance, Mourinho's[Mou15] work, whose goal was to find the optimal solution, the goal of this work is to find a valid map solution and reduce execution time and complexity of the generation process.

Furthermore, a result is considered good if grid adaptation was successful, lines emerge from the hub at the determined location (i.e., there are no lines placed inside the hub), all angles are octilinear and the spider map is drawn according to the set of rules (lines have an assign colour and parallel segments are drawn octilinear). Even though tests were only performed for Porto's bus network, the number of possible hub inputs is very extensive. Hence, the tests focused on testing areas where the network is denser, i.e., areas served by many public transports' lines like city centres. In Porto, some of the busiest areas are *Aliados*, *Casa da Música*, *Hospital São João*, *Castelo do Queijo* and others.

Tests demonstrated that the developed algorithm produces successful spider map results for the city of Porto. Figures 3.20 depicts the initial map state for *Praça da República*, a busy centre in Porto, and Figure 3.21 depicts the corresponding generated spider. In Appendix B several spider map results are illustrated, for several areas of Porto city and with different complexities. The complexity of the generation process and, subsequently, the spider map is directly related to the number of map points, i.e., the complexity increases as the number of map points also increases, since more displacement operations and angle corrections will be needed to generate a valid map. Hence, to control the continuous increase in complexity, a limit to the number of lines in the spider map was set, as well as a limitation on the hub size. This prevents the user to select very large areas for the algorithm, preventing the exponential increase in complexity.

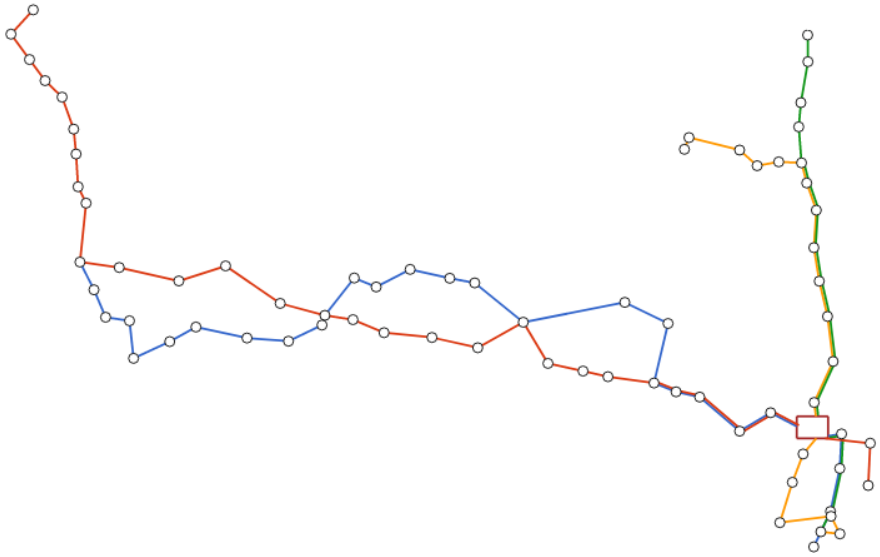

Figure 3.20: Initial map state for *Praça da República* hub area

The work developed by Mourinho aims at automating the spider map generation process and find the best map solution, while this work goals is to simplify the generation process and find a valid map solution. Several tests were performed by Mourinho for accessing the quality of the solutions over parametrization. For instance, tests for accessing the quality of solution versus the number of iterations produced results with average execution times of 2797 seconds.

Even though is not possible to establish a direct comparison with the tests performed by Mourinho, it is possible to conclude that the developed solution was able to produce results faster. The developed solution produced spider maps under 500 milliseconds for complex centre areas. Table 3.4 depicts tests results for valid solutions, describing the number of map points, the numbers of different lines of the map, hub area and the execution time (ET) in milliseconds. In addition, Table 3.5 depicts the success of test results, identifying how frequently a valid solution was obtained, the number of times where a solution was not possible and the number of incorrect solutions (i.e., spider map results that contain some non-octilinear angles).

As conclusion is possible to affirm that the developed algorithm successfully produces results, i.e., the solution generates valid spider map results at real time, taking significantly less time compared to state-of-the-art solutions. Thus, this work successfully tackles the complexity of the spider map generation process and contributes to the identified gap of current work.

3.4.2 Limitations

The quality of the result will depend how and if the stages of the algorithm are successful. In the grid adaptation stage, the algorithm will adapt the cell size until the initial map is successfully adjusted to the grid; however, in some cases, grid adaption may not be possible. In very dense areas, a vast number of map points compete for a grid allocation. Thus, even by increasing the grid granularity, it may not be possible to attribute a grid point to every map point. Moreover,

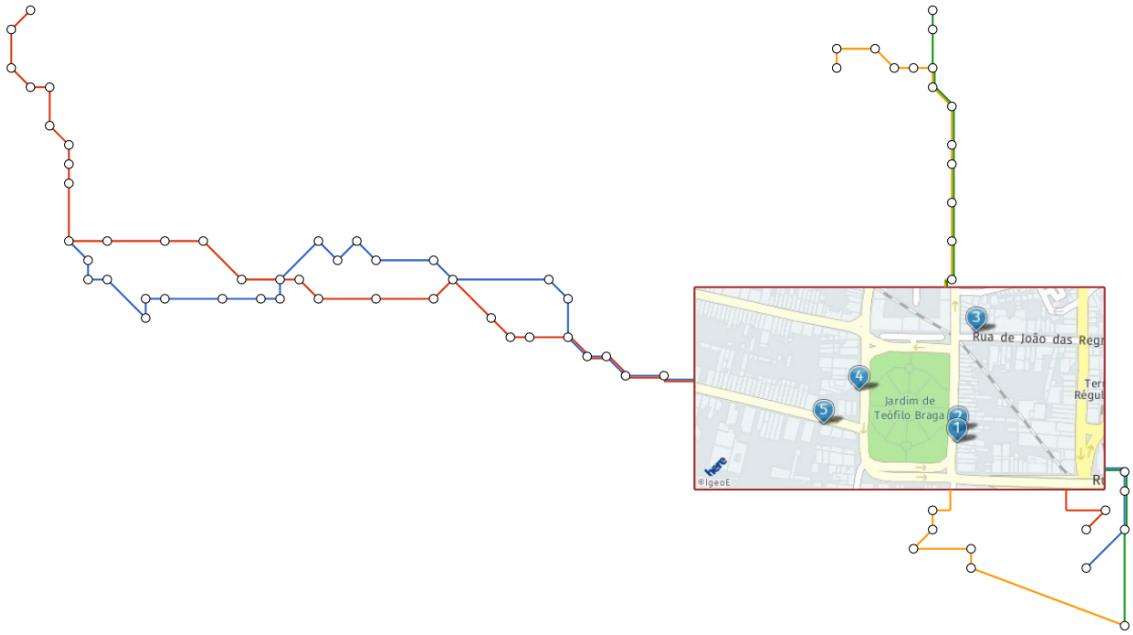

Figure 3.21: Spider Map result for *Praça da República* hub area

since the subsequent algorithm steps depend on the success of grid adaptation, a solution will not be found.

Nevertheless, introducing a restriction to the maximum number of lines contained the occurrence of this problem, and tests showed that the grid adaptation process is successfully completed even in complex areas, and with just one or two iterations.

The next algorithm step that will influence the quality of the solution is the correction of non-octilinear angles. In the developed solution, the algorithm has several iterations that aim correcting the non-octilinear angles through several approaches. The first approach is identifying a valid grid allocation to displace the identified map points and correct the angle. Howbeit, in some cases is not possible to find a valid displacement that corrects the angle, thus the next iterations try to correct the remaining non-octilinear angles by inserting one or two break points. The integration of different approaches to correct identified non-octilinear angles was effective in producing valid spider map results.

Notwithstanding, in some cases the algorithm may not produce a valid spider map (i.e., some angles may not be corrected) or, in the worst-case scenario, not produce a solution. Most invalid algorithm results derive from the non-octilinear angle correction, not the grid adaption as depicted in Table 3.5. Thus, even for invalid results, the algorithm can present a solution that may not be completely correct (some angles may not be octilinear).

Many of these cases arise from errors in the data, perhaps the result from migrating the data from the OPT database to the implemented one. Some errors are the result of incorrect map point coordinates, that lead to incorrect projections, which subsequently cause the failure of grid

Table 3.4: Tests results for generated spider maps

No. map points	No. Lines	Hub Area	ET (ms)
153	6	Castelo do Queijo	844.29
153	6	Castelo do Queijo	710
32	1	Av. Boavista	122.21
153	6	Casa da Música	419.23
144	1	Casa da Música	298.64
10	1	Aliados	35.04
108	4	Aliados	387.71
130	4	Trindade	664.88
52	2	Boavista	215.83
88	2	Hosp. São João	272.87
106	4	Hosp. São João	432.53
102	4	Av. Boavista	5445
32	1	Av. Boavista	102.67
105	4	Praça da República	482.9
105	4	Aliados	496.27
27	1	Bolhão	95.45
39	1	Campo Lindo	177.45
66	3	Marquês	244.62
177	6	Marquês	599.46
37	6	Passeio Alegre	105.96
51	6	Foz do Douro	159.23
33	6	Ramalde	102.33
79	6	Parque Real	194.93

adaptation or angle correction.

On the other hand, circular lines are viewed as a special case, since they sometimes lead to particular results. For instance, results with circular lines often cross themselves, which may be valid according to spider map restrictions, but is not very aesthetically pleasing. Also, the rescaling the hub operation may lead to undesired distortion, that in some cases may preclude the success of the non-octilinear angle correction.

Even though some limitations were identified and the algorithm may be improved so it becomes more robust to certain cases, results have proven that the developed solution was successful and provides advancements in the current state-of-the-art solutions. The solution is able to produce viable spider map solutions at real-time and taking in consideration the hub area as user input. Furthermore, the prototype demonstrates the the successful integration of the algorithm with the advantages of digital maps by incorporating visualisation and interaction techniques.

Furthermore, the spider map solutions can be aesthetically improved in a post-processing stage, with more line simplifications. Nonetheless, the developed solution provides advances in the simplification of the generation process of spider maps, thus potentially making an impact on the use of spider maps in providing public transports information. Through the developed prototype, the user is able to choose a desire hub area and visualise all the travel possibilities by the

Table 3.5: Outcome of performed tests

Solution	**No. of results**
Valid solution found	22
No solution found	3
Solution with errors	5

generated spider map.

3.4.3 Future Work

Performed tests demonstrated that the developed solution contributed to simplifying and automating the spider map generation process, however, the complexity of this problem opens the possibility for several improvements.

The main goal throughout the development of the presented solution was to generate a valid spider map, i.e., a map that follows the design restrictions set for spider map. Thus, even though valid spider map results were obtained, some results may not be very aesthetically pleasing. This implies that lines may follow octi-linearity, but their shape can be complex. Thus, future work may focus on improving the aesthetically value of the spider map results, by adding more simplification to the overall shape of lines.

Furthermore, in the previous section several limitations were identified. Thus, to make the developed algorithm more robust, the aforementioned points should be addressed. For instance, when it is not possible to correct some non-octilinear angle, the algorithm may adapt the grid or try to displace other map points, so an angle correction is possible.

To conclude, future work may be done relating to the integration of new interaction and visualisation techniques. The developed prototype already implements main techniques, such as navigation, zoom and interaction with some map elements. However, other techniques may be explored, which may enhance the obtained results. Such techniques can be different information for different levels of zoom or add interactive elements to the hub, e.g., markers for stops.

3.5 Conclusions

The generation of spider maps is mainly a manual process and current state-of-the art solutions are complex, requiring great computational effort and taking long execution times to produce map results. Hence, the proposed problem aims at developing a solution capable of automatically generate spider maps, tackling the complexity gap of current solutions.

The developed algorithm takes as input the hub coordinates and applies several map points displacement operations to achieve a valid spider map solution. Summarising the process, all the map data is modelled as a graph and all the subsequent operations are performed over this data. Afterwards, the map is adapted to a defined grid and non-octilinear angles are corrected over different processes. Finally, the map is draw and interaction capabilities are integrated using *D3.js*.

Even though the algorithm does not guarantee that a valid solution is always generated, performed tests show the developed solution was successful in contributing to the identified gap. Spider maps were automatically generated during execution times shorter than other state-of-the-art solutions and the developed prototype successfully integrates visualisation and interaction techniques. On the other hand, invalid solutions may arise for incorrect data retrieve from the database, and future work may approach this issue and improvements can be done to make the algorithm more robust to greater map complexities.

Chapter 4

Conclusions

Interactive Spider maps for providing public transport information aim at enhancing the traditional approach to support passengers by taking advantage of the spider map capabilities for providing network information.

Spider maps have great potential for providing public transport information, since they portray all the travelling possibilities of an area. They combine elements from both geographic and schematic maps and offer better spatial context as they have a central area, called the hub, that depicts the geographic location of the map. Schematic lines emerge from the hub, depicting the trajectory of routes.

However, these maps are not widely used for providing public transports information, which may be derive from the generation process still being mostly manual. Some works ambitioned and were successful to automate the generation process of spider maps. However, the developed solutions are complex and require great computational effort to produce results. On the other hand, digital maps open new possibilities for integration with interaction and visualisation techniques, enhancing their information value with customised and dynamic ways to understand and learn map information.

The state-of-the-art revision identified a gap in current solutions. The generation process of spider maps could be simplified, enabling the automatic creation of map results at real-time, thus, possibly making an impact on the use of spider maps for providing public transport information. Furthermore, human-computer interaction techniques can be applied to spider maps to enhance the interface and learnability of the maps.

Henceforward, this work is focused on two goals: develop an algorithm that automatically generates spider maps results at real-time and considering the hub as input; and develop a prototype that integrates the map generation algorithm and add interactive capabilities to map results. The algorithm adapted techniques used in schematic maps generation, such as adapt a map to a pre-defined grid, and develop new processes that apply several operations to produce a spider map

compliant to all the design restrictions. The prototype used the *D3.js*[1] tool to assist the visualisation and interaction with the map. *D3.js* is a powerful open-source tool that provides several data manipulation operations and integrated interaction events, ideal for the prototype goals.

Throughout the validation and evaluation process, the objective was to test if the solution could produce valid spider map solutions at real-time, reducing the execution time needed to produce map results. The prototype and tests focused on Porto bus network and all the data was provided by OPT[2].

Performed tests showed that the solution is successful and can produce map results in shorter execution times than state-of-the art solutions. Furthermore, the prototype developed validated that the algorithm can be successful integrated in a web application that provides an interface for passengers to interact and customise the map generation.

Future work may improve the map aesthetics in a post-processing phase by applying more simplification to the spider map schematic lines, which will increase the quality of the solution. Moreover, the algorithm may not produce results in certain cases. This problem can derive from incorrect network data stored in the database and the greater complexity of some areas, which leads complex spider maps (maps with a vast number of map points, increasing the number of operations needed to produce a valid solution). Hence, the algorithm can be improved to become more robust to complex areas and adapt itself when a certain solution is not possible.

Notwithstanding, the developed solution contributed to the identified gap in state-of-the-art solution, producing spider map solutions at real-time and considering user input. Even though results may not have the quality as the ones presented in, for instance, Mourinho's work, it may be more valuable for passengers to present a valid customised solution at real-time, than to value quality. Furthermore, the solution may present value for providing public transports information, since it enables users to select a desire area and examine all the travel possibilities of the network by interacting with the spider map automatically generated through the prototype.

[1] https://d3js.org/
[2] http://www.opt.pt/

References

[AA91] Nuray M Aykin and Turgut Aykin. Individual differences in human-computer interaction. *Individual differences in human-computer interaction*, 20(3):373–379, 1991.

[AA99] Gennady L. Andrienko and Natalia V. Andrienko. Interactive maps for visual data exploration. *International Journal of Geographical Information Science*, 13(4):355–374, 1999.

[Abd16] Rifaat Abdalla. Mobile GIS and Location-Based Services (LBS). In *Introduction to Geospatial Information and Communication Technology (GeoICT)*, pages 83–103. Springer International Publishing, Cham, 2016.

[AH06] Silvania Avelar and Lorenz Hurni. On the Design of Schematic Transport Maps. *The International Journal for Geographic Information and Geovisualization*, 41(3):217–228, 2006.

[AM00] Silvania Avelar and Matthias Muller. Generating Topologically Correct Schematic Maps. In *9th International Symposium On Spatial Data Handling*, 2000.

[An 19] An Taisce- The National Trust for Ireland. An Taisce published 'Schematic Map' of Dublin Transport Proposals. http://www.antaisce.org/articles/taisce-published-schematic-map-dublin-transport-proposals, 2019.

[Ave02] Silvania Avelar. Schematic Maps on Demand: Design, Modeling and Visualization. *Unpublished Ph. D. Dissertation, Swiss Federal Institute of Technology*, (14700):142, 2002.

[BB15] Andrea Ballatore and Michela Bertolotto. Personalizing Maps. *Communications of the Acm*, 58(12), 2015.

[BCE+16] Nigel Bevan, Jim Carter, Jonathan Earthy, Thomas Geis, and Susan Harker. New ISO Standards for Usability, Usability Reports and Usability Measures. volume 9731, pages 268–278, 2016.

[BGS01] Patrick Baudisch, Nathaniel Good, and Paul Stewart. Focus Plus Context Screens - Combining Display Technology with Visualization Techniques. *Proceedings of the International Symposium on User Interface Software and Technology (UIST'01)*, pages 31–40, 2001.

[CAR19] CARRIE COUSINS. How to implement gestures into your mobile design. https://thenextweb.com/dd/2015/11/09/how-to-implement-gestures-into-your-mobile-design/, 2019.

REFERENCES

[CBKF00] Hernan Casakin, Thomas Barkowsky, Alexander Klippel, and Christian Freksa. Schematic Maps as Wayfinding Aids. pages 54–71, 2000.

[CEN19a] CEN. SIRI. http://www.transmodel-cen.eu/standards/siri/, 2019.

[CEN19b] CEN. Transmodel. http://www.transmodel-cen.eu/, 2019.

[CEN19c] CEN. Transmodel- purpose. http://www.transmodel-cen.eu/overview/purpose-of-the-transmodel/, 1 2019.

[Cou07] National Research Council. Chapter 3: Research Priorities. In *A Research Agenda for Geographic Information Science at the United States Geological Survey*, page 74. The National Academies Press, Washington, DC, 2007.

[DP11] David H Douglas and Thomas K Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. In *Classics in Cartography*, chapter 2, pages 15–28. John Wiley & Sons, Ltd, 2011.

[Eng19] English Oxford Living Dictionaries. "public transport". https://en.oxforddictionaries.com/definition/public_transport, 1 2019.

[GB13] Judith Gelernter and Shilpa Balaji. An algorithm for local geoparsing of microtext. *GeoInformatica*, 17(4):635–667, 2013.

[Gen19] General Transit Feed Specification (GTFS). http://gtfs.org/, 1 2019.

[Goo19] Google. GTFS- Overview. https://developers.google.com/transit/gtfs, 1 2019.

[HHP03] Frank Heidmann, Fabian Hermann, and M Peissner. PD126 Interactive maps on mobile, location-based systems: design solutions and usability testing. *Human - Centered Computing: Cognitive, Social and Ergonomic Aspects*, (August):1258–1262, 2003.

[Hun78] Earl Hunt. Mechanics of verbal ability. *Psychological Review*, 85(2):109–130, 1978.

[IBM19] IBM. The Client/Server model, 2019.

[In 19] In the Book. London Tube Map. https://www.inthebook.com/en-gb/literary-tube-map/, 6 2019.

[Int19] International Cartographic Association. History of ICA. https://icaci.org/research-agenda/history/, 2019.

[JDWH02] Longin Jan, Latecki Daniel Wildt, and Jianying Hu. Extraction Of Key Frames From Videos By Optimal Color Composition Matching And Polygon Simplification. 2002.

[JSA13] Krzysztof Janowicz, Simon Scheider, and Benjamin Adams. A geo-semantics flyby. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8067 LNAI:230–250, 2013.

[KHPG11] J. Kohlbrecher, S. Hakobyan, J. Pickert, and U. Grossmann. Visualizing energy information on mobile devices. *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS'2011*, 2:817–822, 2011.

[KK00] Alexander Klippel and Lars Kulik. Using grids in maps. *Theory and application of diagrams. First International Conference, Diagram 2000, Edinburgh, Scotland, UK, September 1-3, 2000 Proceedings*, pages 486–489, 2000.

[KRBF05] Alexander Klippel, Kai Florian Richter, Thomas Barkowsky, and Christian Freksa. The cognitive reality of schematic maps. *Map-based Mobile Services: Theories, Methods and Implementations*, pages 55–71, 2005.

[KRR13] Jens Kolb, Benjamin Rudner, and Manfred Reichert. Gesture-Based Process Modeling Using Multi-Touch Devices. *International Journal of Information System Modeling and Design*, 4(4):48–69, 2013.

[Kup05] Axel Kupper. *Location-based Services: Fundamentals and Operation*. John Wiley & Sons, Inc., USA, 2005.

[Loh89] David F Lohman. Human Intelligence: An Introduction to Advances in Theory and Research. *Review of Educational Research*, 59(4):333–373, 1989.

[Lov05] Steve Love. *Understanding mobile human-computer interaction*. Elsevier Butterworth-Heinmann, 2005.

[Mac12] Francisco Miguel Amaro Maciel. *Interactive Spider Maps for Public Transportation*. PhD thesis, Faculty of Engineering of University of Porto, 2012.

[MD16] Francisco Maciel and Teresa Galvão Dias. Challenging user interaction in public transportation spider maps: A cobweb solution for the city of Porto. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, pages 181–188, 2016.

[MDN19] MDN. Express.js REST API. https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Express_Nodejs/roteamentos, 6 2019.

[MM12] Wojciech Mokrzycki and Samko M. New version of Canny edge detection algorithm. pages 533–540. 2012.

[Mou15] João Mourinho. *Automated Generation of Context-Aware Schematic Maps: Design, Modeling and Interaction*. PhD thesis, Faculty of Engineering of University of Porto, 2015.

[OG15] Patrick Olivier and Klaus-Peter Gapp. *Representation and Processing of Spatial Expressions*. Psychology Press, 1st edition, 2015.

[Par11] Eli Pariser. *The Filter Bubble: What the Internet Is Hiding from You*. Penguin Group , The, 2011.

[Por07] Thomas Porathe. User-Centered Map Design. 2007.

[PP08] Thomas Porathe and Johannes Prison. Design of human-map system interaction. *Proc. CHI '08, ACM Press*, page 2859, 2008.

[PRS+94] Jenny Preece, Yvonne Rogers, Helen Sharp, David Benyon, Simon Holland, and Tom Carey. *Human-Computer Interaction*. Addison-Wesley Longman Ltd., Essex, UK, UK, 1994.

REFERENCES

[Rei04] Tumasch Reichenbacher. Mobile Cartography – Adaptive Visualisation of Geographic Information on Mobile Devices. *Fakultat fur Bauingenieur– und Vermessungswesen*, PhD(June):189, 2004.

[RRL12] João Tiago Ribeiro, Rui Rijo, and António Leal. Fast Automatic Schematics for Public Transport Spider Maps. *Procedia Technology*, 5:659–669, 2012.

[Rub94] Jeffrey Rubin. *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[Saa99] Alan Saalfeld. Topologically Consistent Line Simplification with the Douglas-Peucker Algorithm. *Cartography and Geographic Information Science*, 26(1):7–18, 1999.

[SB92] Manojit Sarkar and Marc H. Brown. Graphical Fisheye views of graphs. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 83–91, 1992.

[Soc19] Sociedade de Transportes Colectivos do Porto. São João hospital spider map. https://www.stcp.pt/fotos/spider_map, 2019.

[SS] Remote Sensing and Environment Science. Visualizing Schematic Maps Through Generalization Based on. *Archives*, pages 379–384.

[SV04] Jochen Schiller and Agnès Voisard. Introduction. In Jochen Schiller and Agnès Voisard, editors, *Location-Based Services*, The Morgan Kaufmann Series in Data Management Systems, pages 1–5. Morgan Kaufmann, San Francisco, 2004.

[Tra19] Transports of London. Buses from Baker Street and Marylebone. http://content.tfl.gov.uk/bus-route-maps/baker-street-and-marylebone-010417.pdf, 2019.

[UXL19] UXLab. HUMAN COMPUTER INTERACTION (HCI) – And how it influences UI & UX Design. http://theuxlab.weebly.com/blog/human-computer-interaction-hci-and-how-it-influences-ui-ux-design, 2019.

[VCW02] Michael P Verdi, Steven M Crooks, and David R White. Learning Effects of Print and Digital Geographic Maps. *Journal of Research on Technology in Education*, 35(2):290–302, 2002.

[Vis89] M Visvalingam. Cartography, GIS and Maps in Perspective. *The Cartographic Journal*, 26(1):26–32, 1989.

[VSV19] VSV. SIRI- specification. https://www.vdv.de/siri.aspx, 1 2019.

Appendix A

Algorithm pseudo-code

A.1 Resize initial map

Algorithm 1: Resize initial map

Input : Hub corners coordinates; map graph G(V,E)
Output: Resized hub coordinates and graph G(V,E)

*calculate hub **width**, **height** and **centre** from coordinates*;
newWidth ← 300 * width / height;
newHeight ← 300 * height / width;

scalingFactorX ← newWidth / width;
scalingFactorY ← newHeight / height;

for ***each*** *v ∈ G(V,E)* **do**
    if *v is hub emerging point* **then**
        calculate vector from hub centre to v;
        calculate new v coordinates applying a translation of the obtained vector with the scaling factor;
    else
        identify the corresponding line hub emerging point;
        apply same translation of the identify hub emerging point;
    end
end
lines ← `GetLines(E)`;
for ***each*** *line* ∈ lines **do**
    if *line is inside* **then**
        calculate maximum distance to hub boundaries;
        apply translation to every line map point;
    end
end

A.2 Adapt to Grid

Algorithm 2: Adapt map to grid

```
Input  : Map graph G(V,E)
Output: Map graph G(V,E) adapted to grid
for each v ∈ G(V,E) do
    if v not hub emerging point then
        calculate nearest grid intersection points;
        for each nearest grid point do
            if grid point is valid then
                calculate grid point score;
            end
        end
        if valid displacements found then
            update v coordinates to grid point with best score;
        else
            if cellSize < min then
                solution not found;
            else
                decrease cell size;
                reset graph to original locations;
                restart GridAdaptation;
            end
        end
    end
end
```

A.3 Correct non-octilinear angles

Algorithm 3: Correct non-octilinear angles with displacement

```
Input  : Map graph G(V,E)
Output: Break point candidates
incorrectVertexes ← FindNonOctilinearAngles(V);
for each v ∈ incorrectVertexes do
    if v not hub emerging point then
        calculate nearest grid intersection points;
        for each nearest grid point do
            if grid point is valid then
                calculate grid point score;
            end
        end
        if valid displacements found then
            update v coordinates to grid point with best score;
        else
            add v to break point candidates breakPointCandidates;
        end
    end
end
```

Algorithm 4: Correct non-octilinear angles with break point insertion

```
Input  : Map graph G(V,E), break point candidates breakPointCandidates
Output: Two break point candidates
for each v ∈ breakPointCandidates do
    if v not hub emerging point then
        calculate nearest grid intersection points;
        for each nearest grid point do
            if grid point is valid then
                calculate grid point score;
            end
        end
        if valid grid intersection gridPt found then
            add break point gridPt to V;
            transform edge E(pt1,pt2) into E1(pt1,gridPt) and E2(gridPt,pt2);
        else
            add v to two break point candidates twoBreakPointsCandidates;
        end
    end
end
```

Algorithm 5: Correct non-octilinear angles with two break points insertion

```
Input  : Map graph G(V,E)
Output: Map graph G(V,E)
for each v ∈ twoBreakPointsCandidates do
    if v not hub emerging point then
        calculate valid nearest grid intersection points;
        for each nearest grid point do
            calculate two grid point combination;
            calculate score;
        end
        if valid two points grid intersection combination gridPt found then
            add break points gridPt1 gridPt2 to V;
            transform edge E(pt1,pt2) into E1(pt1,gridPt1), E2(gridPt1,gridPt2) and
              E3(gridPt2,pt2);
        else
            G(V,E) with non-octilinear angles;
        end
    end
end
```

A.4 Correct hub emerging segments

Algorithm 6: Correct non-octilinear angles of hub emerging segments

```
Input  : Map graph G(V,E)
Output: Map graph G(V,E) with correct hub emerging segments
for each v ∈ G(V,E) do
    if v is hub emerging point then
        find grid intersection for 90^o angle;
        if grid intersection not found then
            find grid intersection for 45^o angle;
        end
        insert break point at grid intersection found;
    end
end
```

A.5 Generate spider map

Algorithm 7: Generate Spider Map

Input : Hub coordinates

Output: SVG of spider map

Retrieve route information from database;

build map graph G(V,E);

adaptToGrid();

if *grid adaptation successful* **then**

    correct non-octilinear angles;

else

    reset graph G(V,E) to original coordinates;

end

drawSpiderMap();

Appendix B

Spider Map results

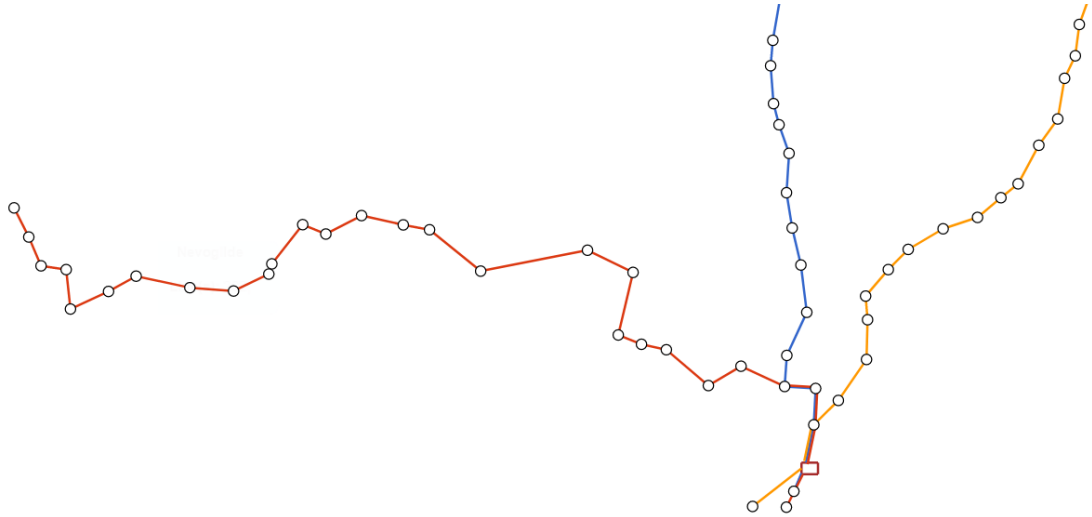

Figure B.1: Initial map for result in Figure B.2

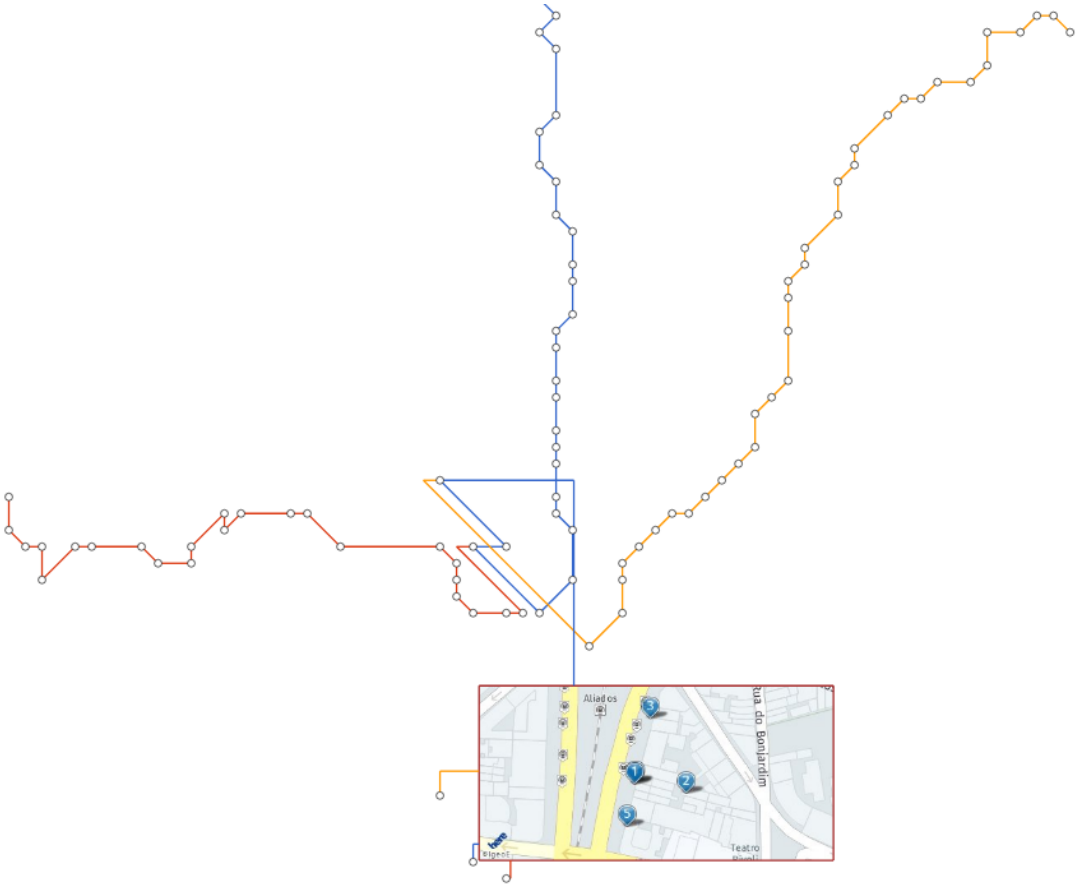

Figure B.2: Spider Map result for an *Aliados* hub area

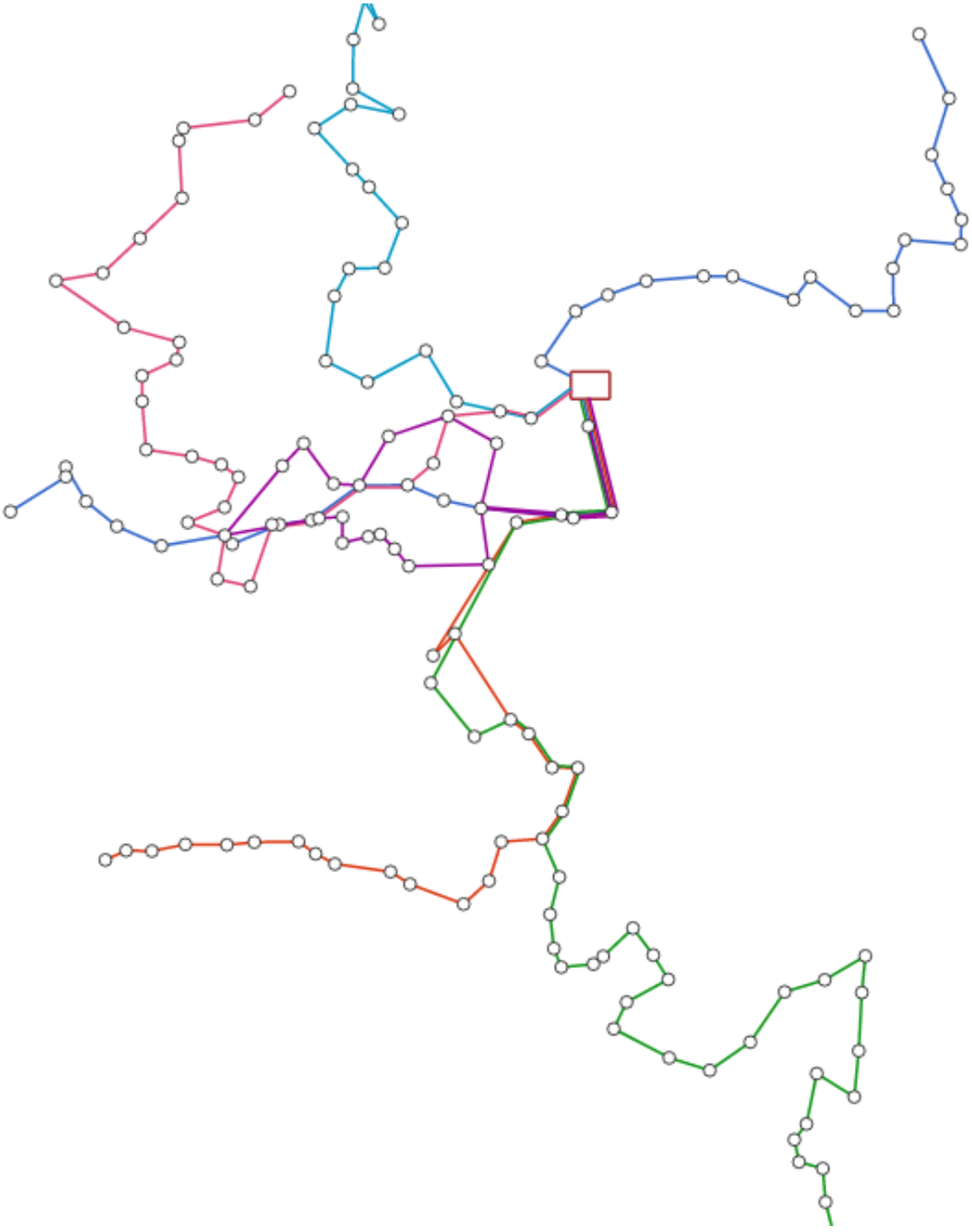

Figure B.3: Initial map for result in Figure B.4

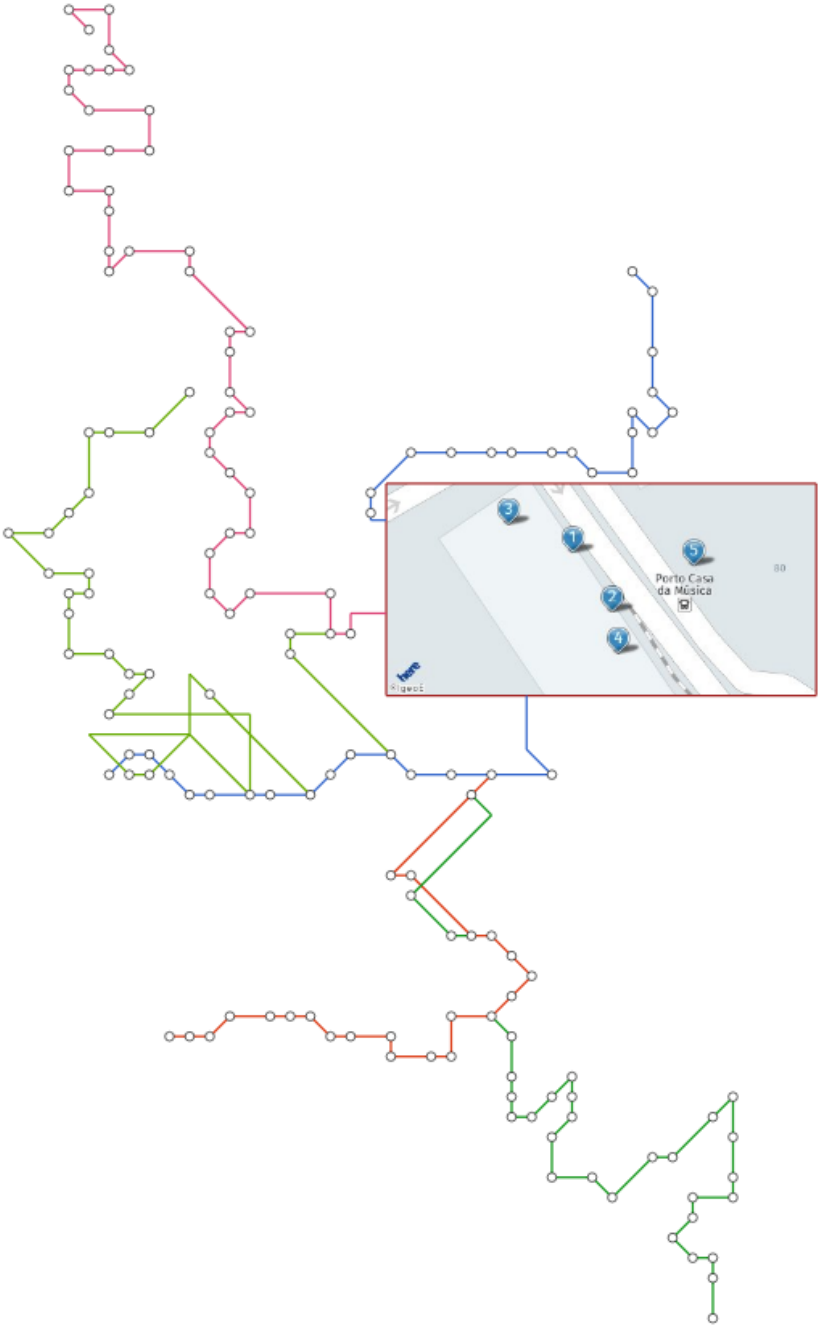

Figure B.4: Spider Map result for *Casa da Música* hub area

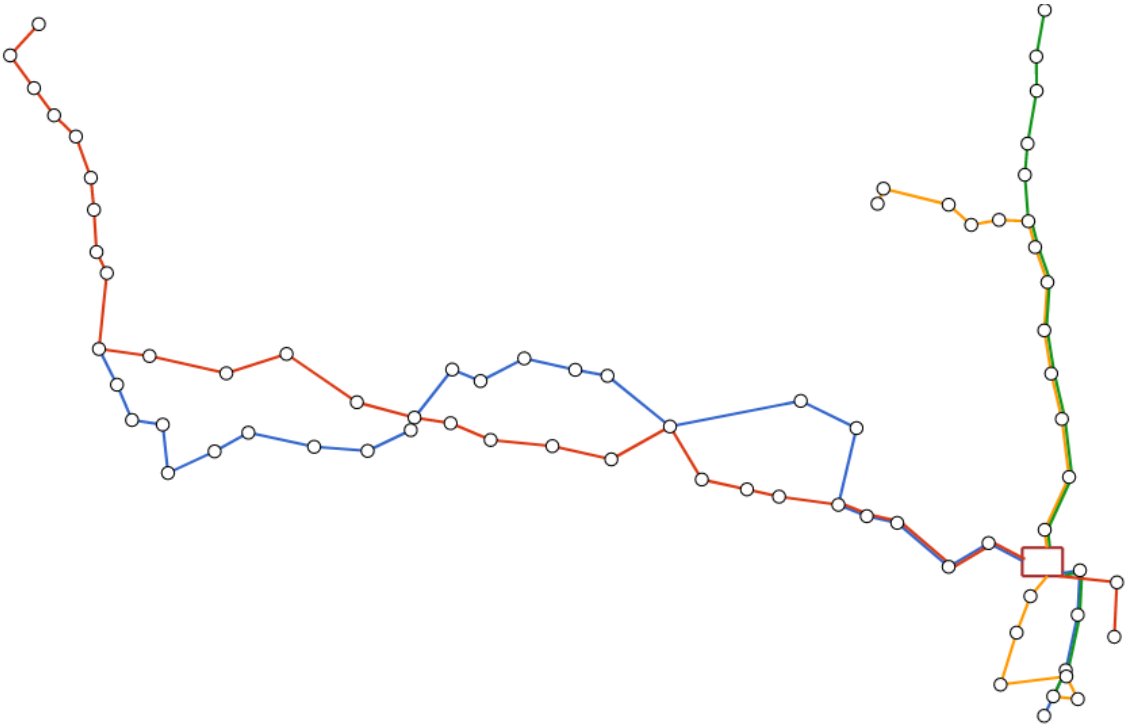

Figure B.5: Initial map for result in Figure B.6

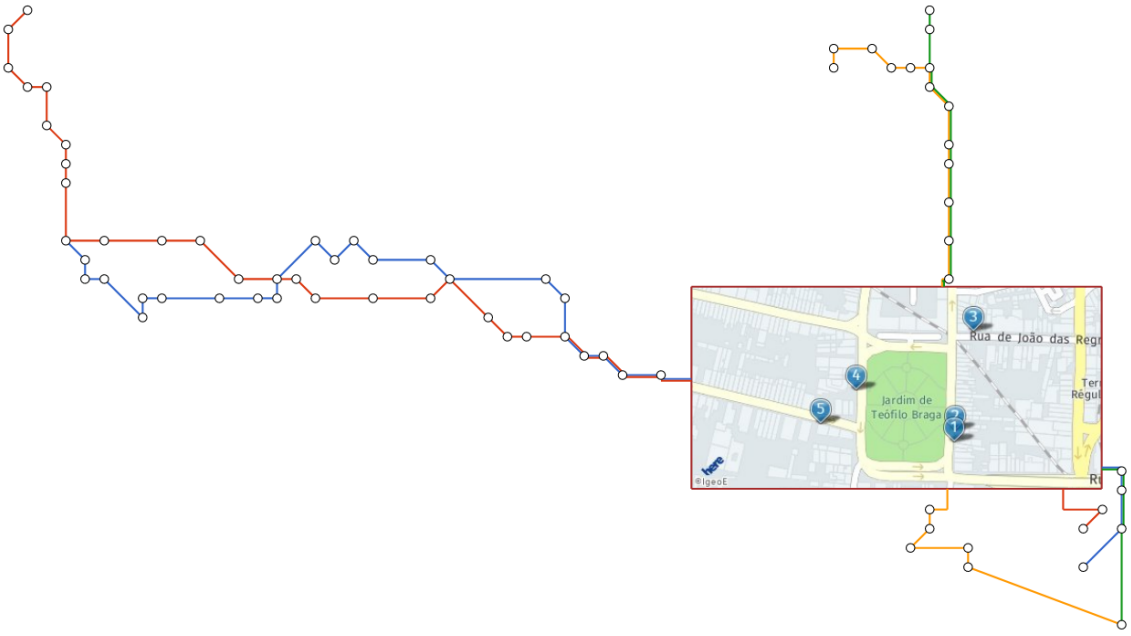

Figure B.6: Spider Map result for *Praça da República* hub area

Figure B.7: Initial map for result in Figure B.8

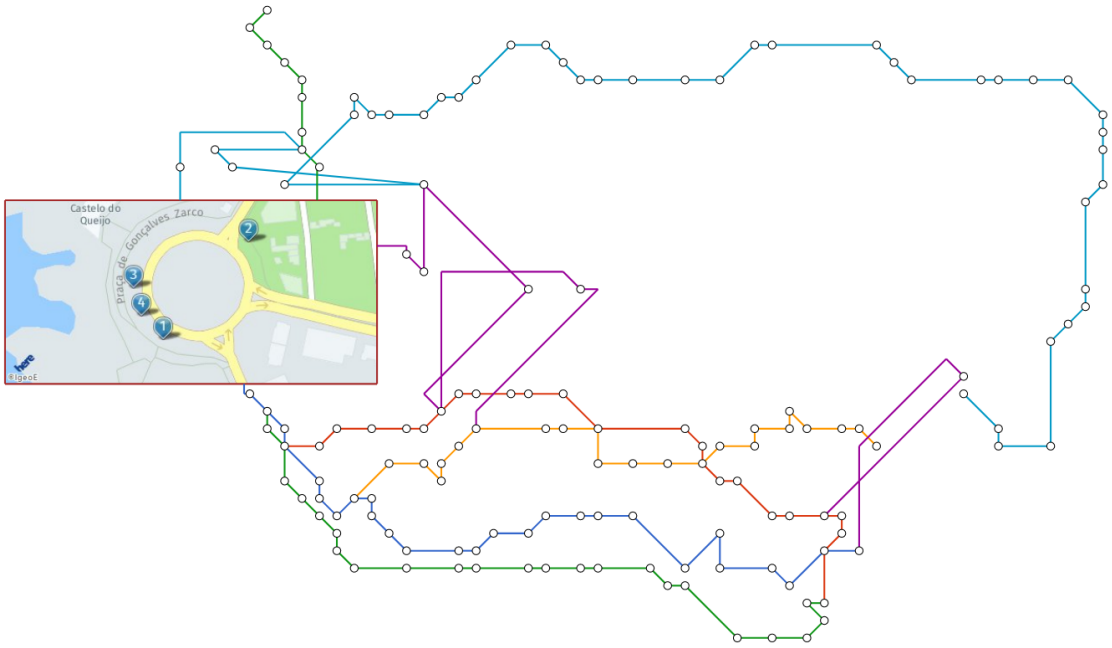

Figure B.8: Spider Map result for *Castelo do Queijo* hub area

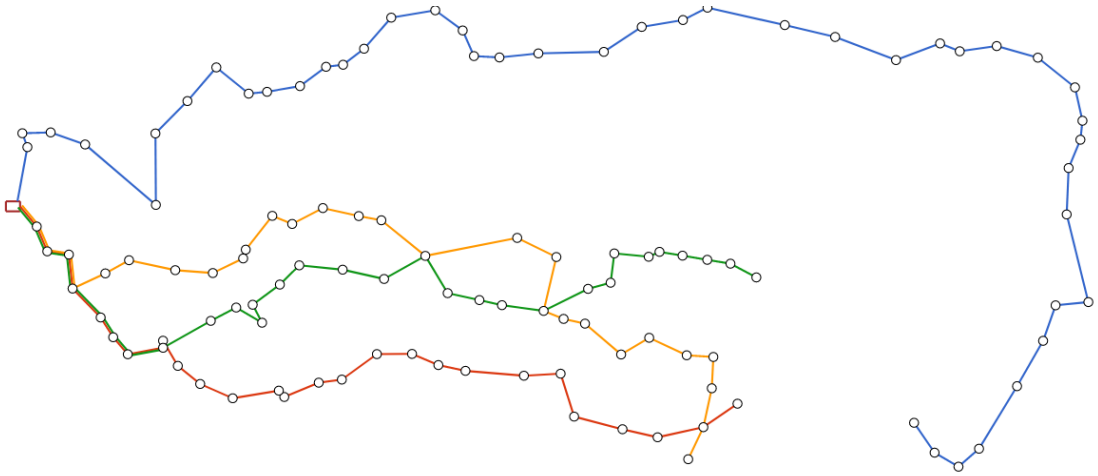

Figure B.9: Initial map for result in Figure B.10

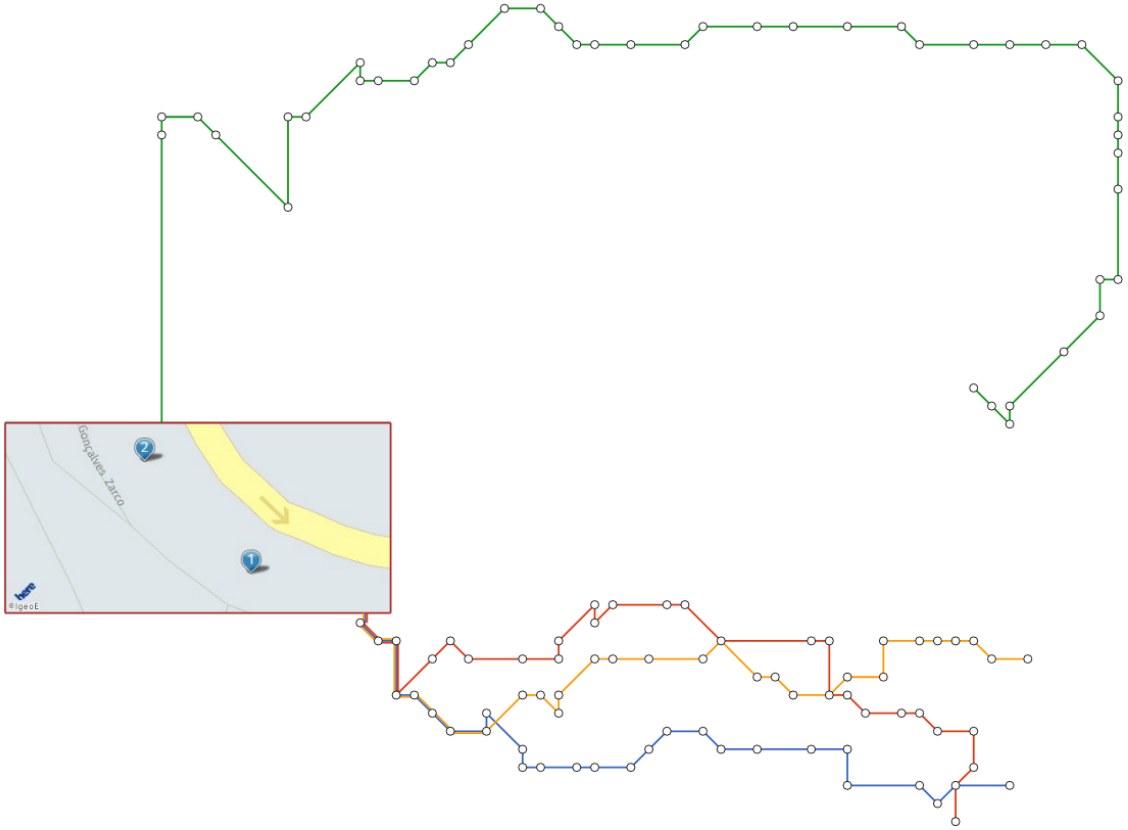

Figure B.10: Spider Map result for *Castelo do Queijo* hub area

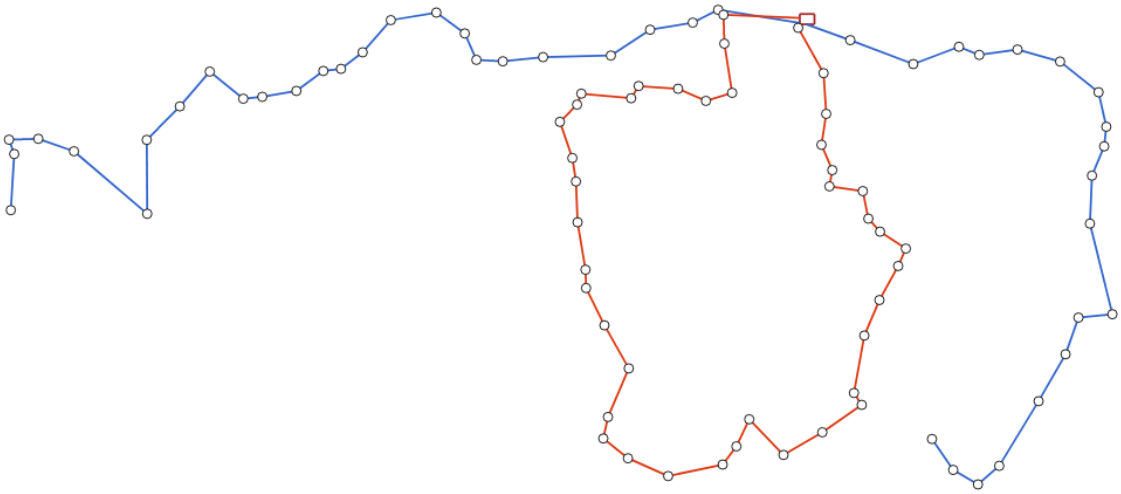

Figure B.11: Initial map for result in Figure B.12

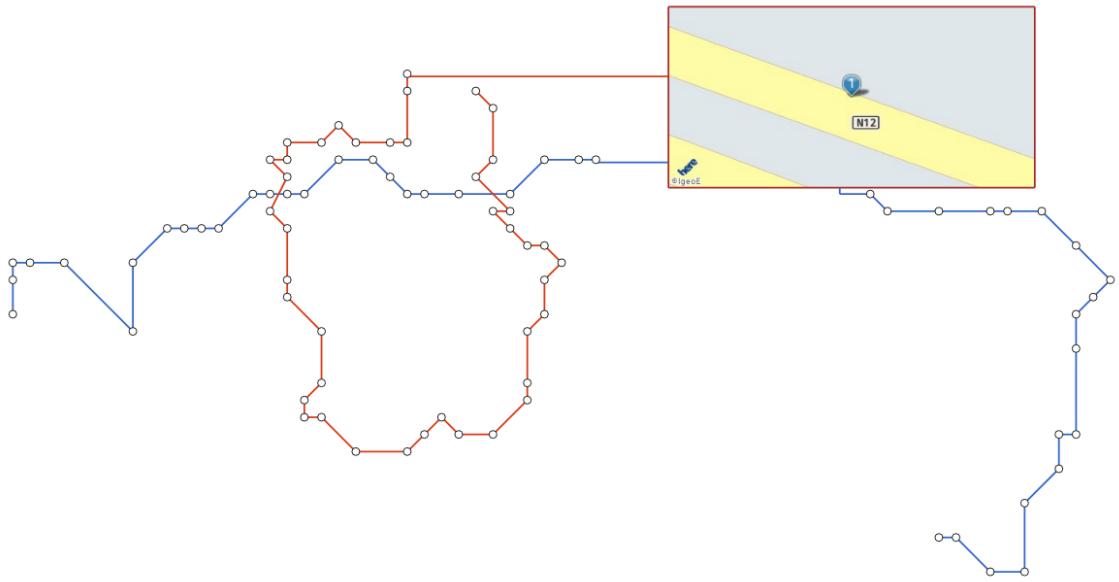

Figure B.12: Spider Map result for *Hospital São João* hub area

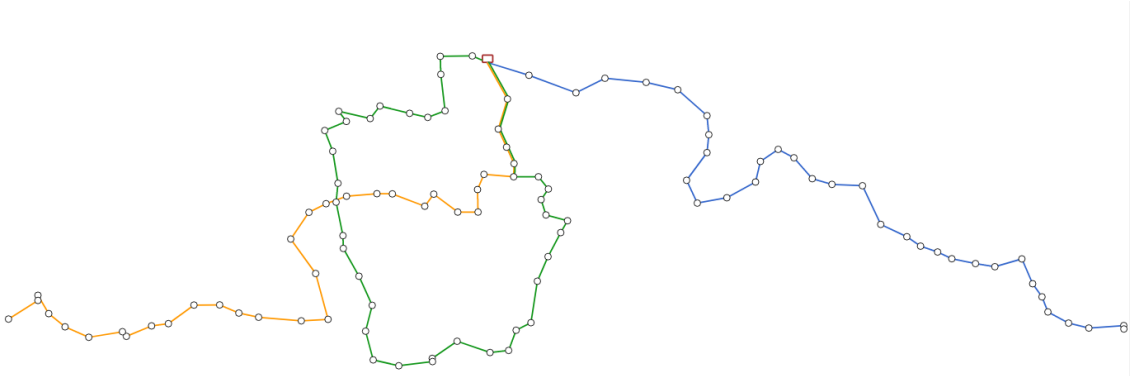

Figure B.13: Initial map for result in Figure B.14

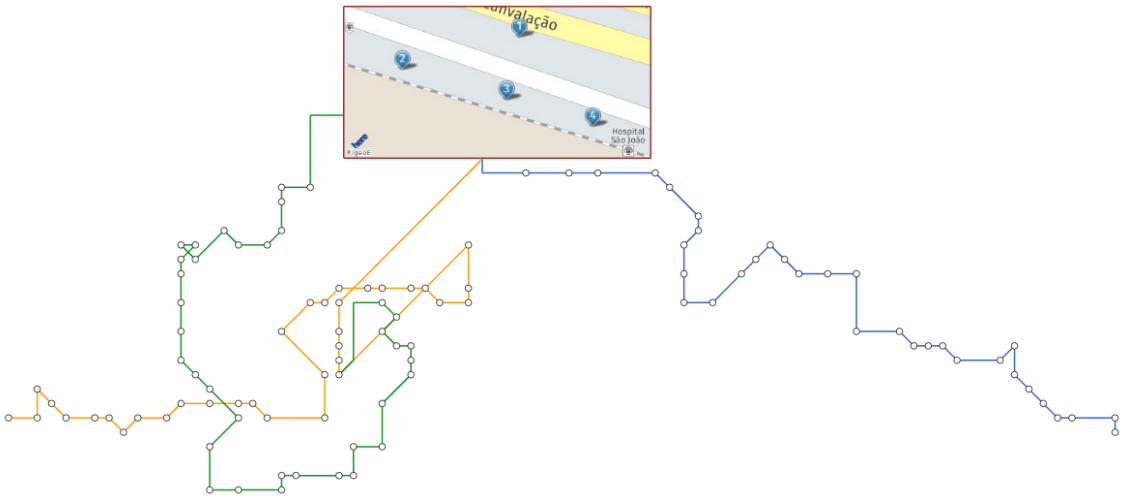

Figure B.14: Spider Map result for *Hospital São João* hub area

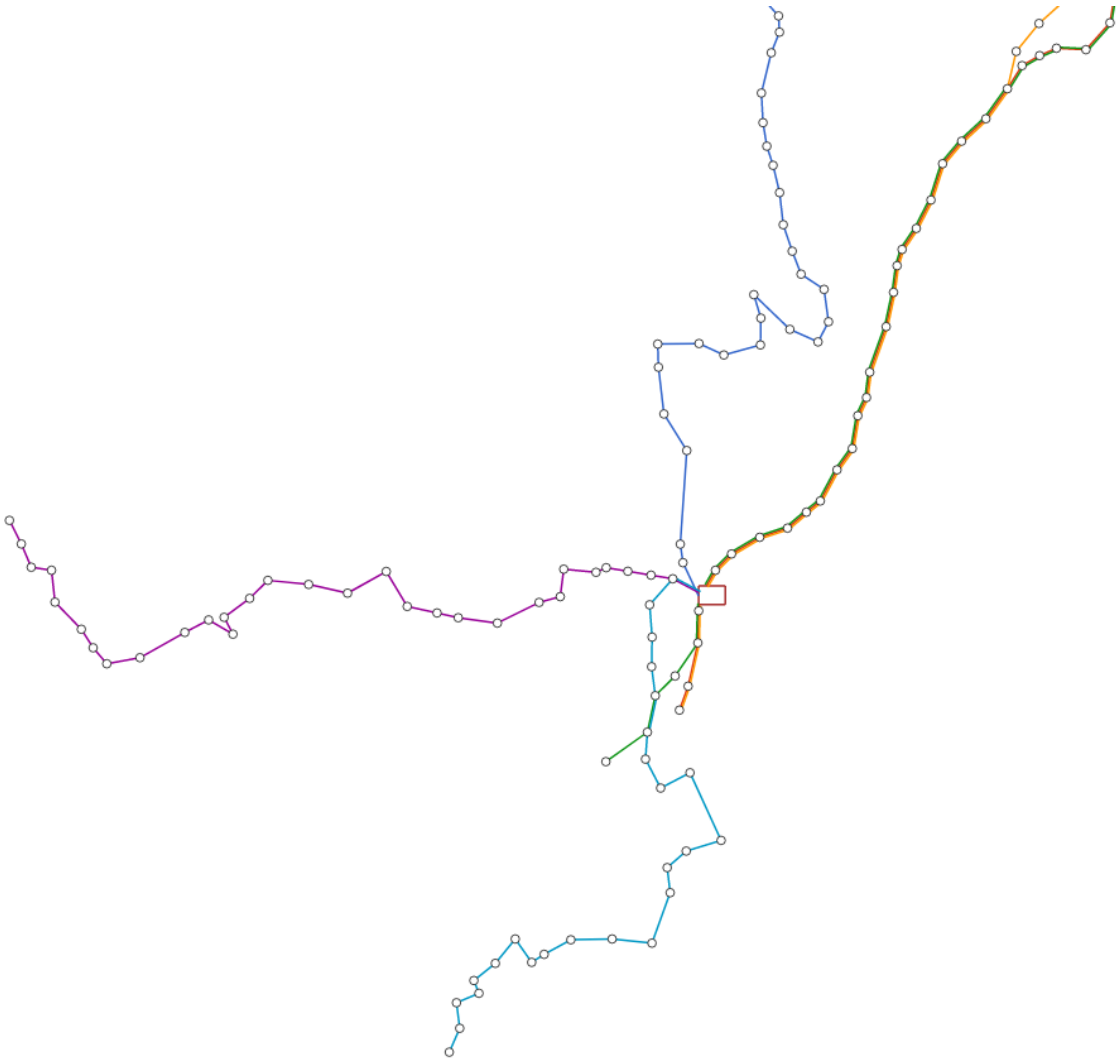

Figure B.15: Initial map for result in Figure B.16

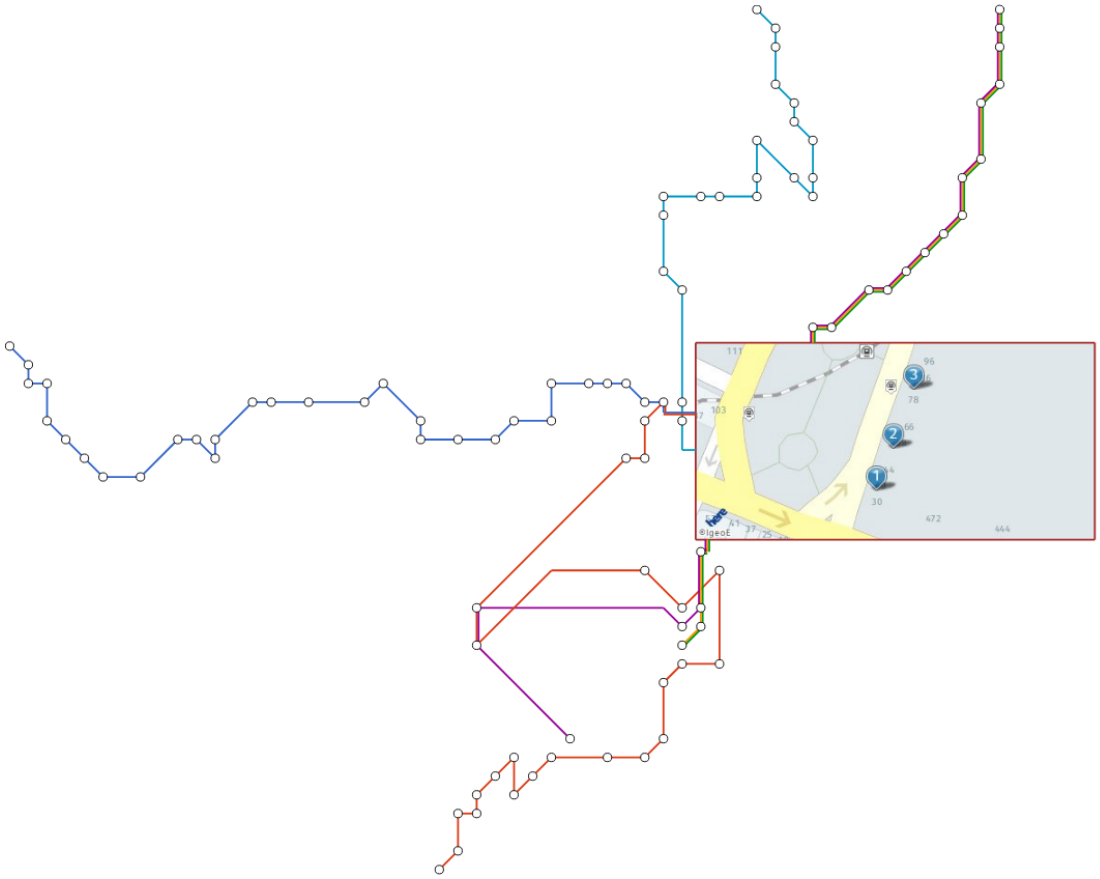

Figure B.16: Spider Map result for *Marquês* hub area

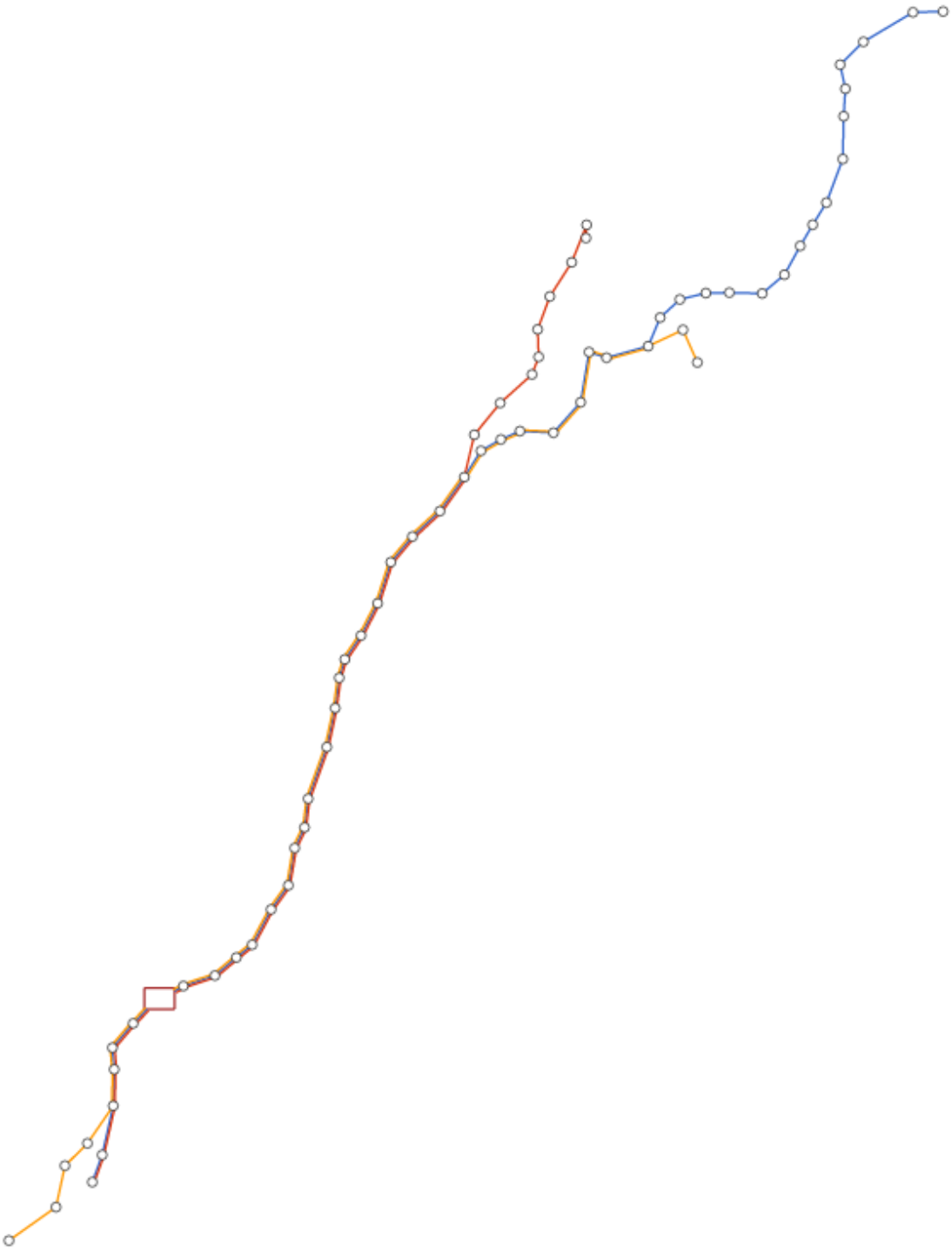

Figure B.17: Initial map for result in Figure B.18

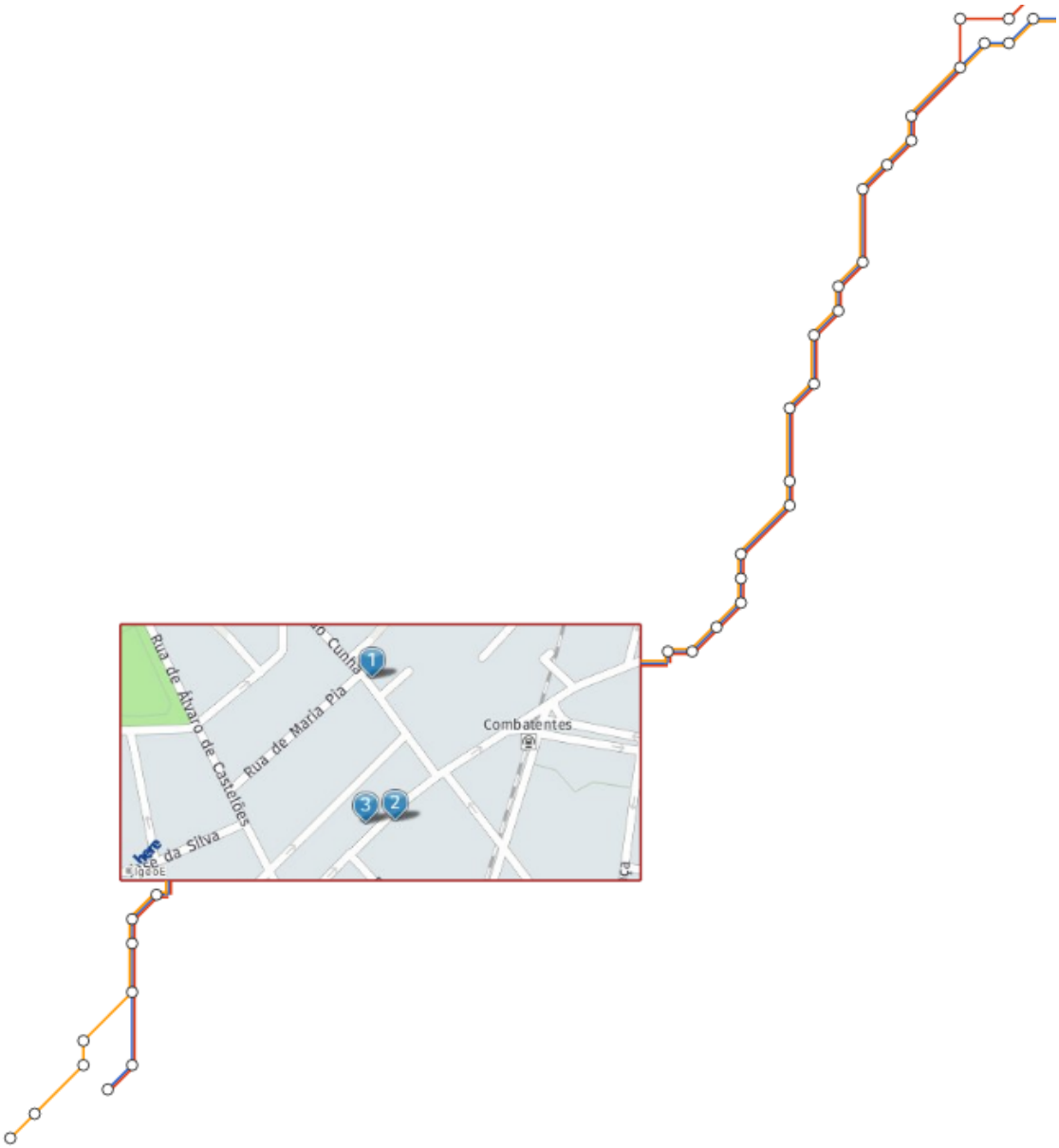

Figure B.18: Spider Map result for *Marquês* hub area

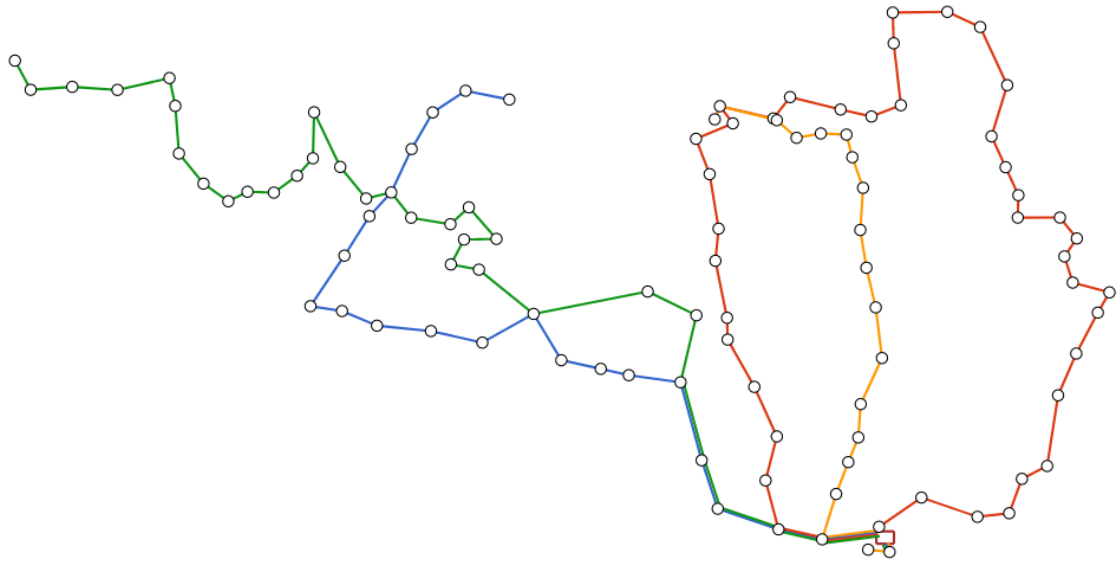

Figure B.19: Initial map for result in Figure B.20

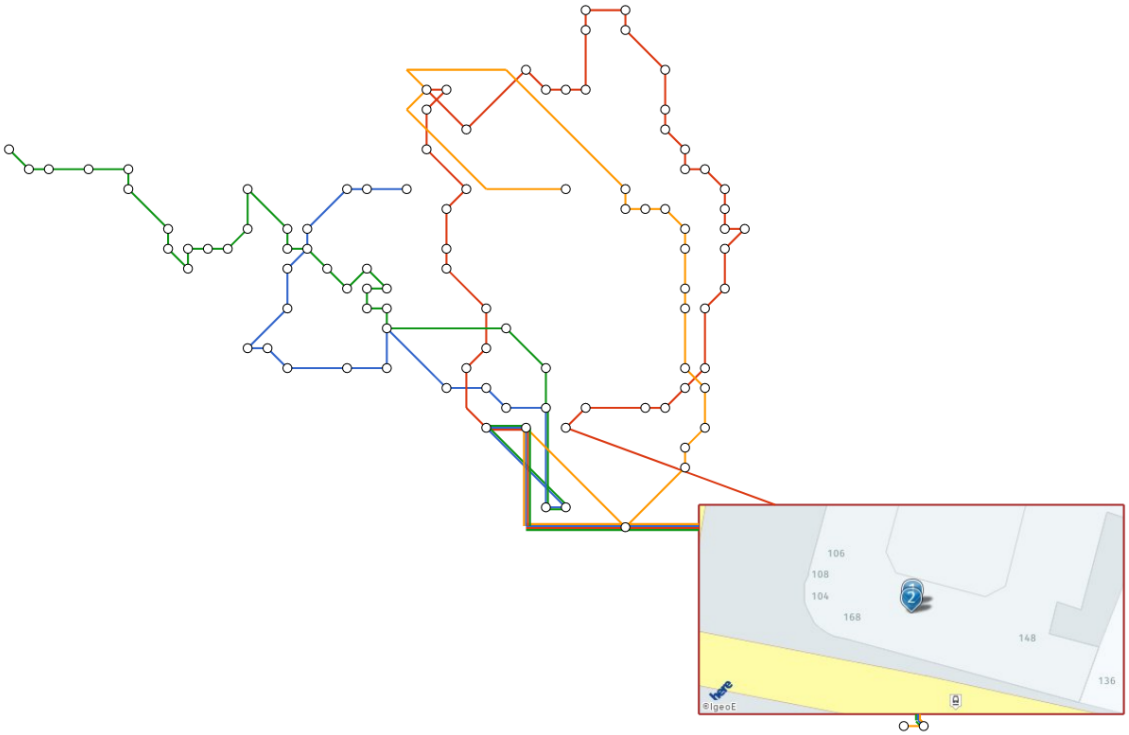

Figure B.20: Spider Map result for *Aliados* hub area

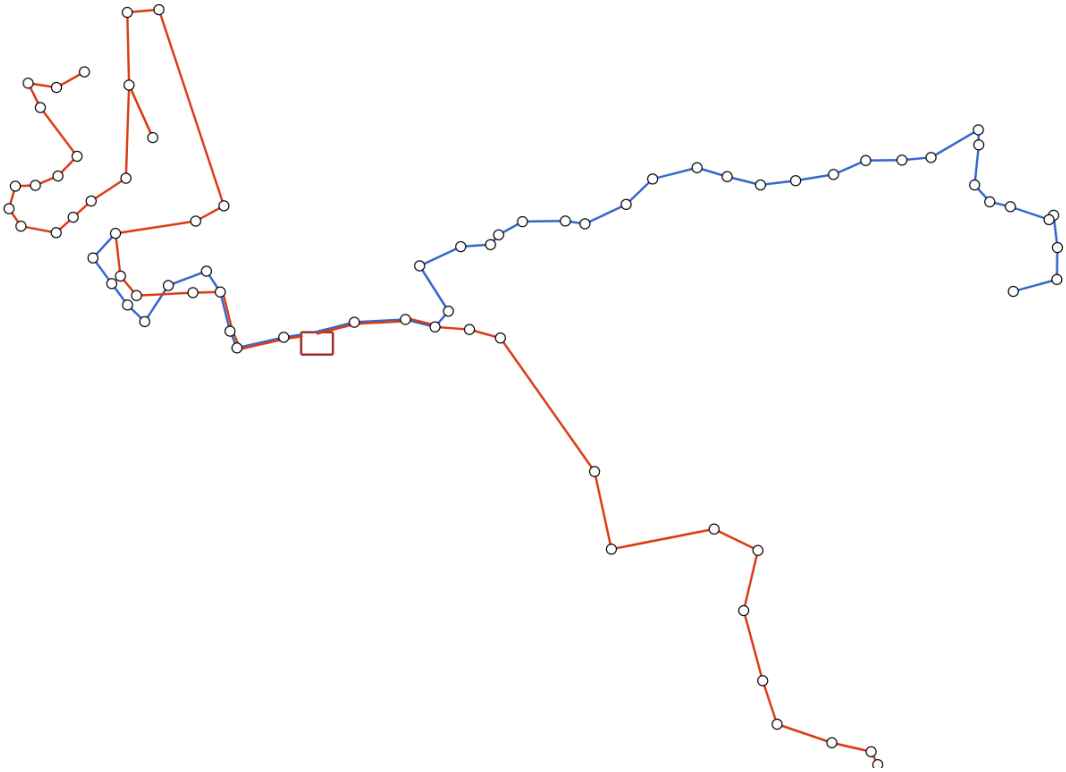

Figure B.21: Initial map for result in Figure B.22

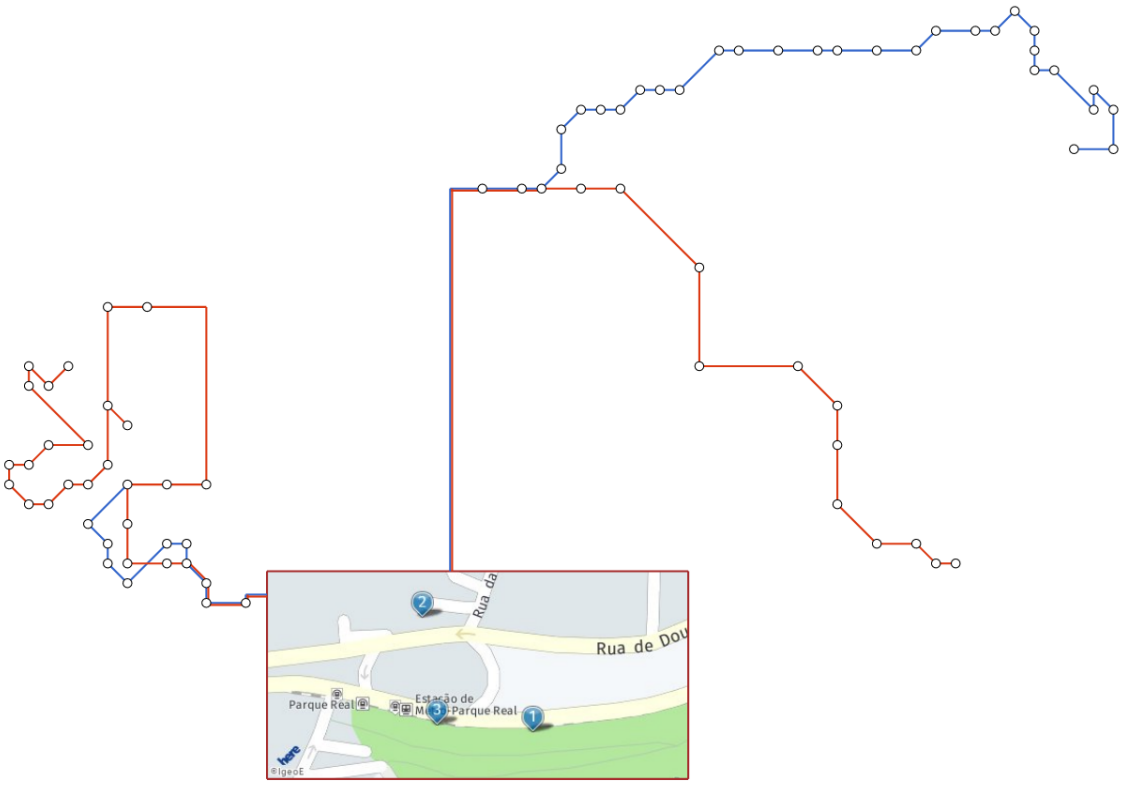

Figure B.22: Spider Map result for *Parque Real* hub area