Automatic Generation of Spider Maps for Providing Public Transports Information

Sara Santos[1(✉)], Teresa Galvão Dias[1,2], and Thiago Sobral[1,2]

[1] Faculty of Engineering of University of Porto, Porto, Portugal
{up201402814,tgalvao,thiago.sobral}@fe.up.pt
[2] INESC TEC, Porto, Portugal

Abstract. With the continuous growth and complexity of public transport systems, it is essential that the users have access to transport maps that help them easily understand the underlying network, thus facilitating the user experience and public transports ridership. Spider Maps combine elements from geographical and schematic maps, to allow answering questions like "From where I am, where can I go?". Although these maps could be very useful for travellers, they still are mostly manually generated and not widely used. Moreover, these maps have several design constraints, which turns the automation of the generation process into a complex problem. Although optimisation techniques can be applied to support the generation process, current solutions are time expensive and require heavy computational power. This paper presents a solution to automatically generate spider maps. It proposes an algorithm that adapts current methods and generates viable spider map solutions in a short execution time. Results show successful spider maps solutions for areas in Porto city.

Keywords: Spider maps · Schematic maps · Public transports · Automation

1 Introduction

Every major city has a complex public transport system that is part of everyday mobility of millions of citizens. These systems are vital for cities mobility and ought to be encouraged as an alternative to private transport. Thus, public transport maps provide simplified representations of the public transport networks, making them easy to interpret, facilitating the user experience and public transports ridership.

These maps are often represented by schematic maps, since they fulfilled the need for better and simpler representation of complex networks [6], presenting the readers the available services and navigation possibilities. A specific type of schematic map is designated spider map, used to represent complex areas, such

A. L. Martins et al. (Eds.): INTSYS 2019, LNICST 310, pp. 131–149, 2020.
https://doi.org/10.1007/978-3-030-38822-5_9

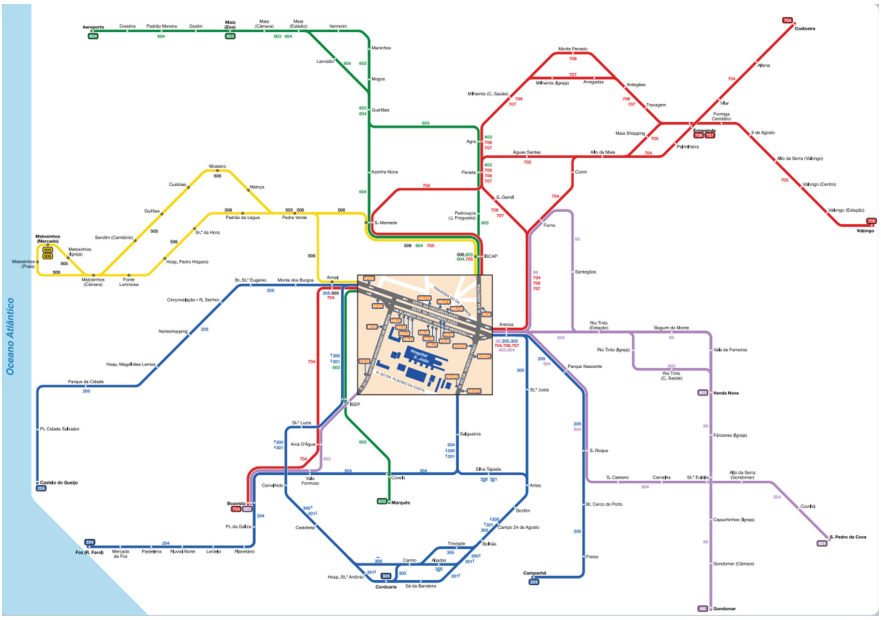

Fig. 1. São João hospital spider map [11]

as bus networks in city centres. For instance, Fig. 1 depicts a spider map created for the São João hospital area in Porto, Portugal.

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 the designer. There are several methods and techniques that can be applied to automate the spider map generation, but current solutions are complex and time expensive.

This work produces a solution able to generate a feasible spider map at run time, presenting an algorithm that modifies and adapts current techniques. The goal is to tackle the complexity of the problem and present viable solutions with short execution times and using less computational power. 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.

In Sect. 2 a brief overlook of relevant state-of-the-art methods and concepts are depicted, while Sect. 3 presents the problem approach and the developed methods. The last section concludes this paper and proposes future work and improvements.

2 Maps for Providing Public Transports Information

Transportation maps support complex public transport networks providing essential information (routes, stops and points of interest) for representing

the transport network [4]. An important process associated with these maps is schematisation, where certain aspects are giving emphasis and unimportant information is removed.

There are many methods for guiding this process. For instance, line generalisation methods, such as **simplification**, where some line points are removed, only maintaining those that ensure the overall line shape; **exaggeration**, that amplifies certain portions of objects; and **enhancement**, where certain features are emphasised to elevate the message [6]. Among other techniques is adapting the initial map (where points correspond to geographical locations) to a grid [5]. In this technique, line points are moved to grid intersections, while ensuring certain constraints, such as orientation and distance between points. The result is a map with a simpler overall shape, where incremental optimisation processes can be applied to improve the result.

Nonetheless, adapting maps to a grid can lead to very saturated areas, for instance, representing complex centre areas that have lines ending on city outskirts. Hence, Sarkar and Brown [10] proposed a method denominated fish-eye that applies different scales throughout the map, thus enabling magnification of crowded areas [2]. This is a *Focus+Context* technique of great value to the schematization process, since it emphasises important information, while maintaining the global context [9].

Spider maps combine elements from both geographical and schematic maps [1] and are used for represent the travel possibilities of complex public transport areas, for instance, bus networks in a city centre [7].

These maps are characterised by a central area called the hub, generally depicted by a rectangular shape which details a geographic map of the location, proving better spatial context [8]. From the hub emerges the schematic lines that represent the network routes. The location where lines emerge from the hub is not arbitrary and should considered route orientation and the stop location within the hub.

The lines in the spider map do not follow the geographic layout, since they are the result of several simplification and displacement operations, introducing the concept of *map point*. A **map point** is a relevant location in the map (e.g. stops or group of stops along the line routes), with coordinates associated to a map canvas that result from several operations during the map generation [9].

Spider maps have many elements in common with schematic maps, thus similar methods can be applied. However, they have extra constraints, e.g., determine the location where lines emerge from the hub, that makes the spider map generation process even more complex. Furthermore, overall topology should be assured, i.e., the general relations between map points should be maintained in order guarantee spatial awareness.

Spider maps' schematic lines are defined by a set of segments and map points, some of them shared by different lines. Shared segments should be drawn parallel separated by a distance greater than zero and each line has a colour associated. Moreover, angles between segments should be octilinear, i.e., should only follow horizontal, vertical or diagonal orientations (0, 45 or 90° angles) [9].

Though spider maps have a great potential for providing public transports information, there is not much information in literature about this type of maps and how to automate the generation. Most studies focus on the efforts made by Mourinho [9] in the development of techniques to automate the generation of spider maps.

In the proposed method, Mourinho [9] models the spider map as a graph with restrictions associated with points (vertexes of the graph) and lines (edges of the graph). The initial algorithm state is a map where map points and lines resemble the geographic location, then a multi-criteria algorithm is applied to determine the best location of each point, while ensuring a set of constraints and design guidelines. The solution successfully attained the proposed goals. However, this is a complex multi-criteria optimisation problem with great computational effort, since it aims at finding the best solution possible.

3 Automatic Generation of Spider Maps

3.1 Problem Definition

The spider map generation process is a complex problem, since these maps have several design constraints as depicted in Sect. 2. 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 Sect. 2. 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. 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 next section, Sect. 3.2, will describe the algorithm for generating a spider map solution, while Sect. 4 will present the prototype development. Results and evaluation of the developed solution will be analysed in Sect. 5.

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

3.2 Map Generation Algorithm

Beforehand, the algorithm needs as input the coordinates of the hub, that will allow querying the server for information relating the stops inside the selected area and the routes that belong to the spider map. The server will gather and process all the information needed and return to the client the set of stops inside the hub and the lines of the map. Each line is defined by a sequence of map points.

The spider map is modelled as a graph *G(V,E)* with vertexes V that represent the map points and edges E, the connection between two vertexes, translating the segments of routes defined by two map points. A graph representation was chosen since map points and route segments are shared between multiple lines, thus avoiding duplicated information and making it easier to ensure the relations between points.

Nonetheless, the map points returned from the server have as coordinates the latitude and longitude of the accurate geographical location. Hence, they need to be projected to the map canvas, which is defined as an SVG (Scalable Vector Graphics) using the tool *D3.js*. Thus, all the latitude and longitude coordinates are projected using the Mercator projection centred on the hub centre. The result is a map where the lines and map points are close to the geographical locations, but now map points have as coordinates x and y associated with the defined canvas.

The next step is to insert the hub and determine the location where the lines should emerge from. Therefore, the intersection points between the line segments and the hub boundaries are calculated and those will be the emerging points and all the other segments inside the hub are eliminated. The intersection represents an approximation of the orientation and path of the route, since the hub is a geographical representation of the area. The following step is to resize the hub, translating the lines to new locations considering the centre and the new hub dimensions and insert the hub image depicting the geographic location. Figure 2 exemplifies a hub in Casa da Música area in Porto, after the aforementioned operations.

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 or south and east or west of another point. When two points have an equal coordinate (x or y), they are called *in line* of each other.

(1) Grid Adaptation. The following algorithm step is to adapt the initial map to a grid, which will lead to an overall simpler shape, closer to fulfil the spider map restrictions. To build the grid, the bounds of the map are calculated, i.e., the maximum and minimum x and y coordinates are determined, that will correspond to the boundaries of the grid. Then, a grid is built with an initial cell size

Fig. 2. Hub example of Casa da Música area

length and all the grid intersection points are calculated. The grid intersection points will be the possible displacement options.

The next step is to move each map point to a neighbour grid intersection with the best score. Thus, for each map point the nearest grid intersection points are determine and a score that translates the quality of the displacement is calculated. The score combines the distance between the map point and the grid intersection and how well the topological relations are maintained if the map point is moved to that location. Thus, for each topological relation that is violated a penalty is given. However, if a topological relation changes to *in line* a smaller penalty is attributed, e.g., if point P1 is north of P2 and, with the displacement, P1 becomes in line with P2, a smaller penalty is credited than if P1 becomes south of P2. This loosen the topological constraints and will result in more straight lines after the grid adaptation process. The map point is moved to the grid intersection with the smaller score. Figure 3 depicts the grid adaptation process, illustrating initial locations in the left image and the displacement result in the right image.

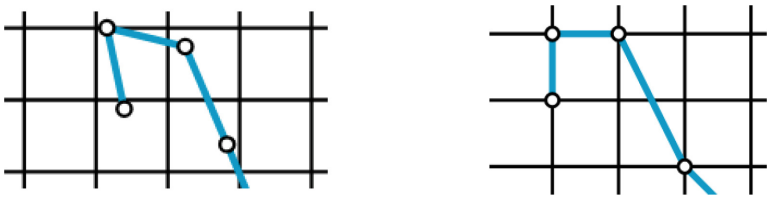

Fig. 3. Grid adaptation process

However, not all the nearest grid intersections are valid displacements. Grid intersection points that will cause hub occlusion, i.e., will intersect the hub, and that will cause line segments to overlap or pass through map points that do

not belong to that segment are removed as possible displacement locations. The addition of this restriction will lead to, in some cases, map points that will not have any possible displacements. When this happens, the graph is returned to the original state, the grid cell size is decrease and the grid adaptation process is restarted. By decreasing the grid cell, the granularity is increased which leads to more displacement options. This process is repeated until all points are displaced to a grid intersection or the grid cell size reaches a defined minimum. In this last case, the grid adaptation process may not be possible, thus a map solution will not be produced.

(2) Correcting Non-octilinear Angles. After the grid adaptation, the result will be a map with simpler line shapes that respect the topology relations, however some of the angles between segments will still not be octilinear. Thus, the next step is to identify the map points where the octilinear angle restriction is not satisfied and correct them. It is important to note that the non-octilinear angles resulting from the segments emerging from the hub are not taken into consideration in this step and will be corrected using a different approach.

Afterwards, for each map point identified with an incorrect angle, the algorithm will try to identify a near grid intersection that will correct the angle. Similar to the grid adaptation process, some of nearest grid intersection points will be removed as possible displacements. Grid intersections points are considered not valid if at least on of the following situations occur: (1) the displacement will cause octilinear angles to become non-octilinear; (2) the displacement will disturb the topological relations between points (changes to in line do not count as disturbance); (3) the displacement will cause occlusions, line overlapping or segments passing through map points that do not belong to that line. Figure 4 illustrates the correction of the non-octilinear angle (depicted in the left image) by displacing a point to a new grid location (right image).

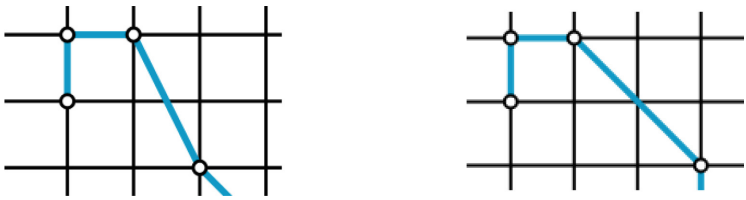

Fig. 4. Non-octilinear angle correction by displacement

However, 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 point will transform one segment in two new ones, which allows correcting the angles without any displacements. Moreover, a break point insertion also has restrictions, since it cannot cause occlusions and

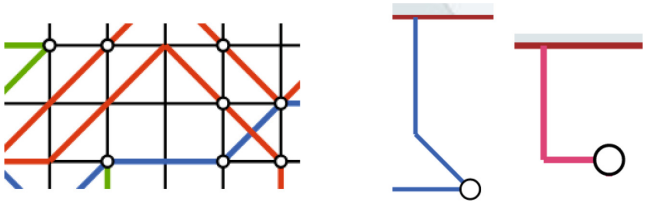

Fig. 5. Non-octilinear angle correction

line overlapping. In some cases, angles cannot be corrected with only a break point, thus two break points are introduced.

The non-octilinear angles of segments that emerge from the hub will be corrected using a different approach. If a segment that emerges from the hub has a non-octilinear angle with the next segment, a perpendicular line to the hub segment is inserted an then a break point is introduced to connect to the next map point. The break point is inserted in a grid intersection that will cause a 90° angle or, if not possible, a 45° angle. Figure 5 depicts the correction of non-octilinear angles by introducing two break points (left image) and the correction of hub related angles.

Even though the several angle correction steps aim at correcting non-octilinear angles through various approaches, in some cases it may not be possible to correct all angles, thus the generated map solution may have some errors.

(3) Draw the Spider Map. After correcting the angles, the final map point locations are determined, and the drawing process can begin. The first is to obtain and place the hub image, that is a geographical representation of the area. The image is obtained using the API Here[2] that returns an image of the geographical map giving a boundary box. Moreover, the stops are identified with markers.

Map points are drawn in the associated locations and do not need further processing. However, segments are shared between lines and need to be drawn parallel, thus making it necessary introducing an offset between shared segments.

In order to introduce an offset that will lead to parallel segments, it is necessary to calculate the slope of the line. Thus, identifying the correct orientation (vertical, horizontal or diagonal), is possible to introduce a correct offset to the x and y coordinates, just as illustrated in Fig. 6.

The final step is to draw the labels that identify the map points. Not all map points need to be labelled, only the last and the most important of each line. Nonetheless, the label's position needs to be determined. Thus, a score is calculated that translates how many occlusions will the label cause. For that, a bounding box of the label is placed at top, bottom, left and right of the corresponding map point and a penalty score is given for each line intersection. The chosen place will be the one with the smaller score.

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

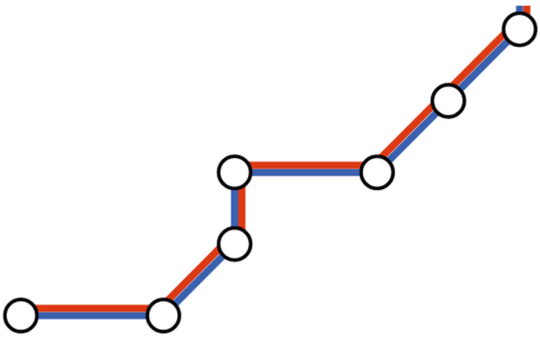

Fig. 6. Shared line segments

After this step, the generation process is finished and a valid spider map solution is presented to the user. Figure 7 depicts the algorithm generation process in a flowchart. Hence, a valid spider map solution is generated if all the aforementioned algorithm steps are successfully completed. In some cases the algorithm is not capable of producing a valid solution, for instance, if grid adaptation fails, no solution will be presented, or if not every non-octilinear angle is corrected, the spider maps will have errors.

The developed algorithm is integrated in the developed prototype depicted in Sect. 4 and results will be illustrated and evaluated in Sect. 5.

4 Prototype Development

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

4.1 Use Cases

The main use cases consist of selecting the desired hub area, and the generation of the corresponding spider map. Figure 8 illustrates the prototype use cases. Moreover, there is the ambition to integrate interaction and visualisation techniques to enhance the user experience.

In the first screen, a map of Porto city with interaction capabilities is presented to the user, 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 Fig. 9 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, the user can select one or combine several grid cells to create a personalised hub area. Figure 9 right shows an example of grid selection,

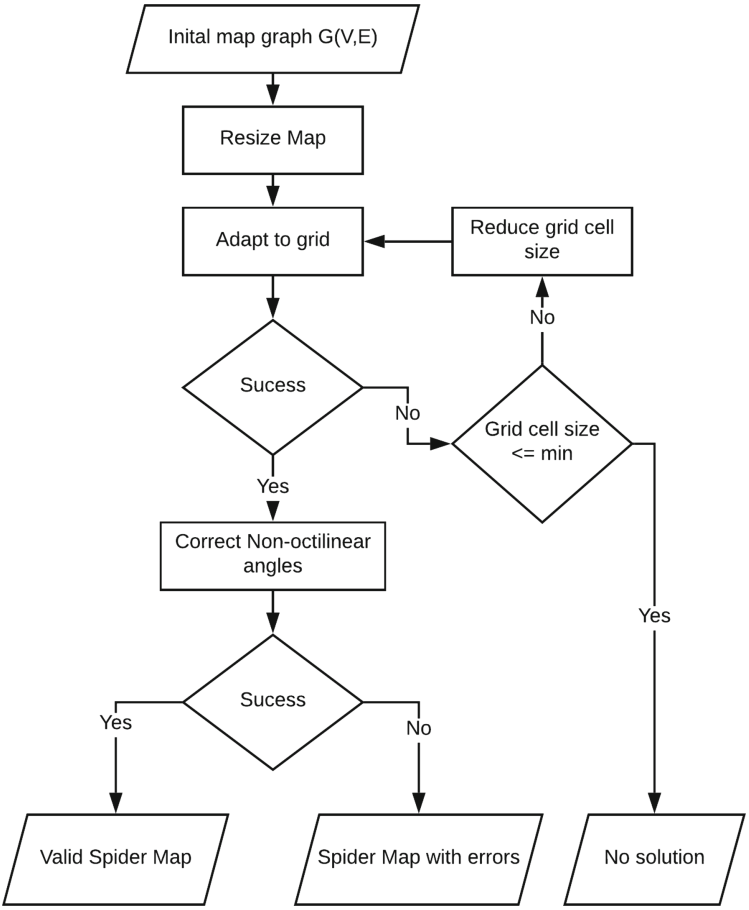

Fig. 7. Generate spider map algorithm

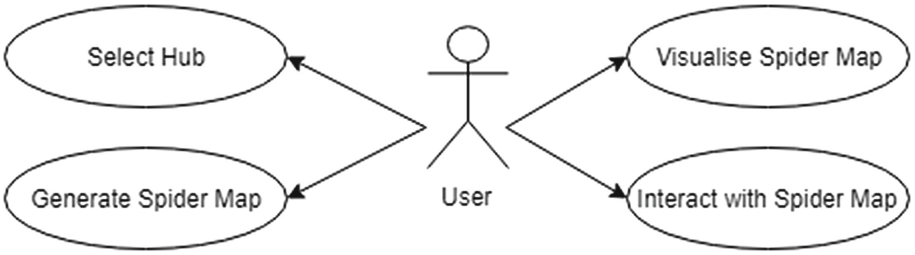

Fig. 8. Prototype use cases

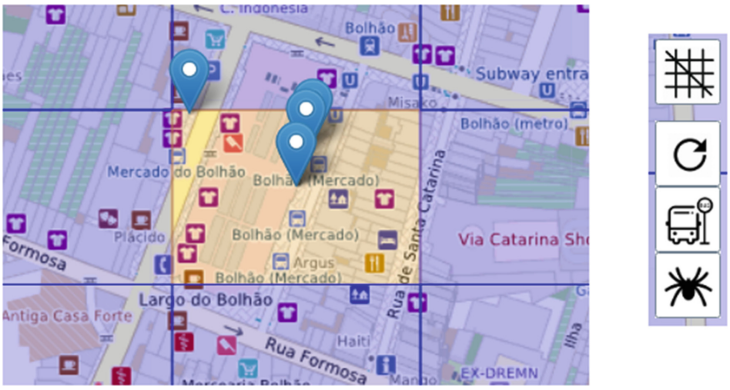

Fig. 9. Example of grid selection (left) and control buttons (right)

where selected cells are shown in orange and markers depict stops inside the hub 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 10 depicts an example of a portion of a spider map result where it is possible to check the additional information box when a map point is hover or clicked on.

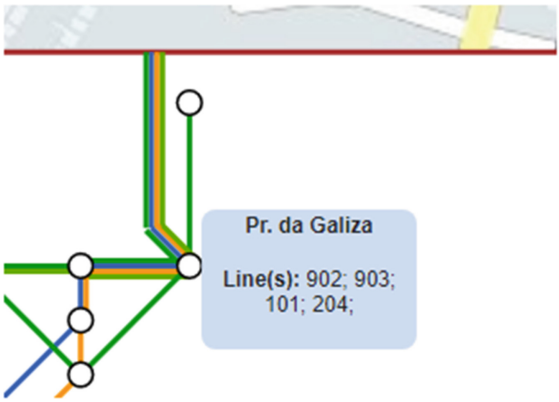

Fig. 10. Interaction with spider map: click on map points to obtain additional information

4.2 Architecture

The developed solution follows a simple two-tier architecture or client-server, illustrated in Fig. 11. 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 [3]. Thus, the server is responsible for dealing with all the necessary data operations, while the client is responsible for the spider map generation and rendering operations.

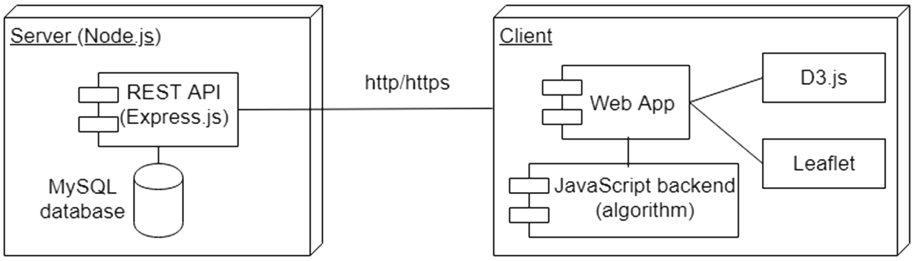

Fig. 11. Solution architecture

The server was implemented using *Node.js*[3] and *Express.js*[4] for the REST API. Moreover, the server establishes a connection with a *MySQL*[5] database that stores all the public transport data, that will be depicted in detail in the next section. Hence, the server is responsible for gathering and processing all the data needed for the spider map generation.

On the other hand, the client consists of a web application based on the two major use cases described in Sect. 4.1. For the geographic maps and hub selection *Leaftlet*[6] and *OpenStreetMap*[7] were used, while the spider map drawing and generation was developed using *D3.js*[8] and *Javascript* technologies.

4.3 Data Model

The data related to public transports of Porto was provided by OPT and stored in a MySQL database following the model depicted in Fig. 12.

Lines are made comprise of several stops that can belong to multiple lines. Moreover, routes are defined by a series of stops associated with an order and recorded in the Path table. The OPT data also identifies map points, which group several stops in a single point. Hence, when gathering the data for the

[3] https://nodejs.org.
[4] https://expressjs.com/.
[5] https://www.mysql.com/.
[6] https://leafletjs.com/.
[7] https://www.openstreetmap.org/.
[8] https://d3js.org/.

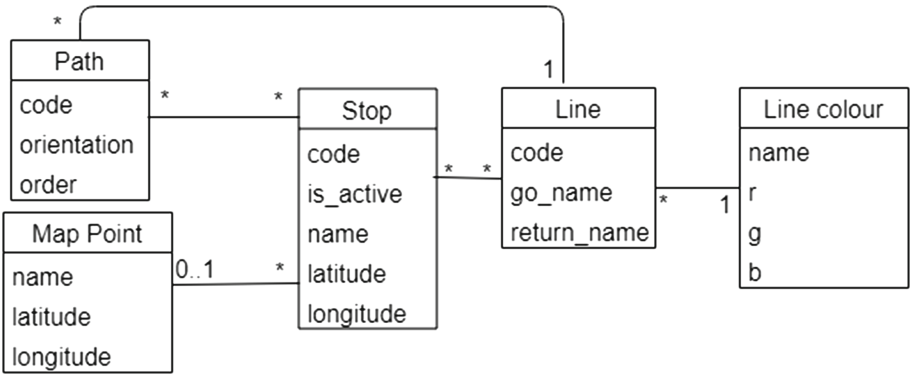

Fig. 12. Data model

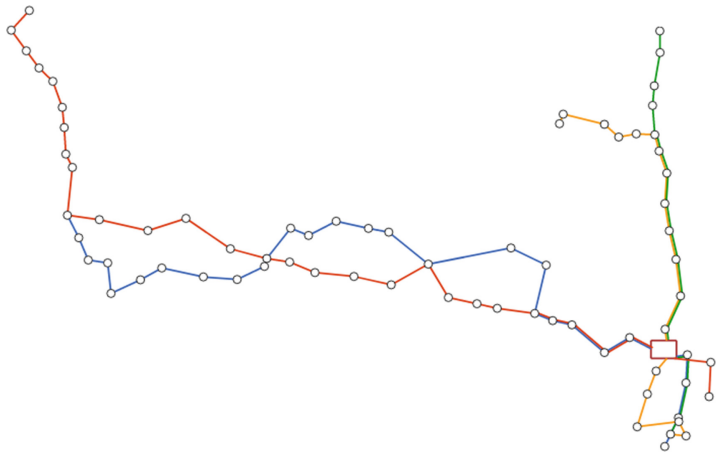

Fig. 13. Initial map state for *Praça da República* hub area

spider map generation, stops along the routes will be replaced or grouped by a map point record and those will be the considered map points during the map generation process.

Furthermore, stops and map points have associated geographical coordinates (latitude and longitude), which will represent the initial position of the points. Moreover, lines and stops also have other attributes associated, such as names and line colours.

5 Evaluation and Validation

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 in 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 Sect. 2.

5.1 Tests and Results

Performed tests aim at testing if the solution is capable of generated valid spider maps at real-time using the prototype develop to select the input hub area and generate and evaluate map results. Several tests were performed by choosing different hub areas as input and evaluating the results. 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 13 depicts the initial map state for *Praça da República* and Fig. 14 depicts the corresponding generated spider. Figure 15 also depicts a spider map result for *Aliados*, a busy centre area in Porto.

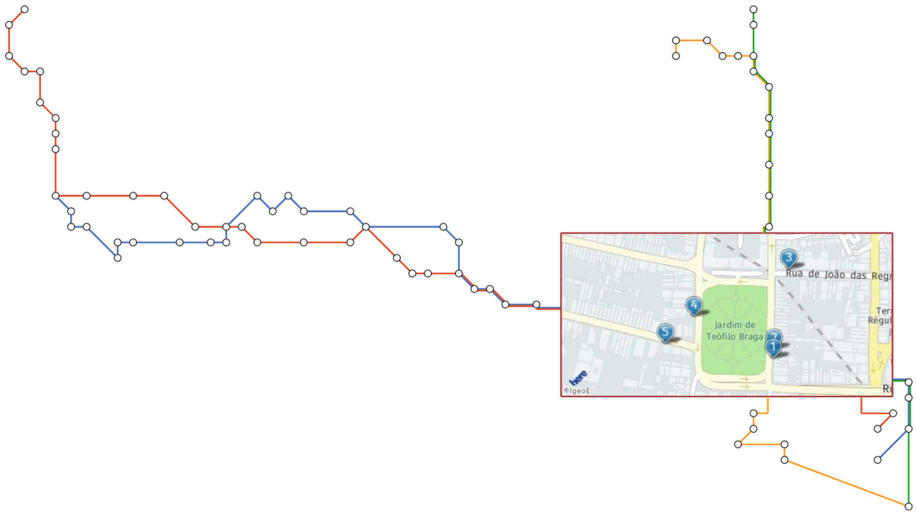

Fig. 14. Spider map result for *Praça da República* hub area

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

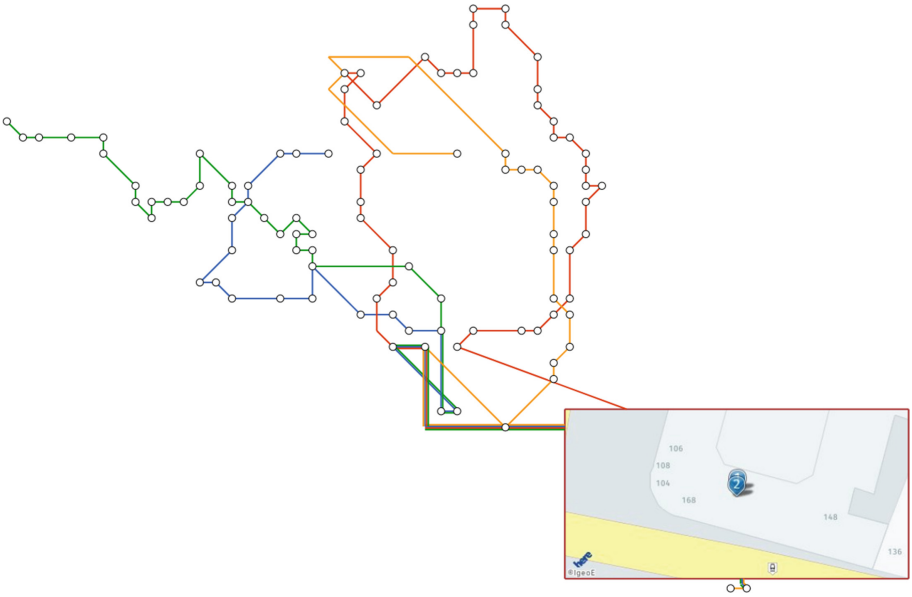

Fig. 15. Spider map result for *Aliados* hub area

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.

There is not much literature in automating the generation process of spider maps, and most of the efforts made in this area were through Mourinho's [9] work. However, in his work the goal was to find the optimal spider map solution, hence the quality was valued over fast results. Thus, the solution required great computational effort and long execution times. For instance, in tests accessing the quality versus the number of algorithm iterations, the average execution times were of 2797 s.

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 ms for complex centre areas. Table 1 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 2 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).

It can be concluded that the developed algorithm successfully produces results, i.e., the solution generates valid spider map results in 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.

5.2 Limitations and Future Work

The quality of the result depends on how and if all 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 assign a grid point to every map point.

Table 1. 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

Table 2. Outcome of performed tests

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

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

Nevertheless, introducing a restriction to the maximum number of lines solved 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. Therefore, reducing the cell size in each iteration to increase the grid granularity was proven successfully.

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. Nevertheless, 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 2. Thus, even for invalid results, the algorithm can present a solution that may not be completely correct (some angles may not be octilinear).

Some errors are the result of incorrect map point coordinates, that lead to incorrect projections, which subsequently cause the failure of grid 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 enhancements in the current state-of-the-art solution. 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.

Finally, 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 generated spider map.

6 Conclusions

Spider maps are a type of transportation map that presents all the public transport possibilities available in an area. These maps are very useful for passengers, but their production is still mostly manual. Some efforts have been made to automate the generation of these maps, but state-of-the-art solutions require great computational effort and long execution times to produce results. Hence, the proposed approach aims at developing a solution capable of automatically generate spider maps, tackling the complexity gap of current solutions, and thus possibly making an impact on the use of spider maps for providing public transport information.

The state-of-the-art review identified a gap in current solutions. The generation process of spider maps could be simplified, enabling the automatic creation of map results in real-time. Furthermore, human-computer interaction techniques can be applied to spider maps to enhance the user experience while manipulating such maps.

Henceforward, this work is focused on two goals: develop an algorithm that automatically generates spider maps results in real-time and considering the hub as input; and to develop a prototype that integrates the map generation algorithm, adding interactive capabilities to map results. The algorithm adapted techniques used in schematic maps generation, such as adapting a map to a pre-defined grid, and developed new processes that apply several operations to produce a spider map compliant to all the design restrictions. The prototype used the *D3.js*[9] 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.

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.

[9] https://d3js.org/.

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, and other algorithm improvements so it becomes more robust to complex network data. 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.

Acknowledgements. This work is 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 a Tecnologia within project PTDC/ECI-TRA/32053/2017 and POCI-01-0145-FEDER-032053.

Furthermore, special gratitude to OPT (http://www.opt.pt/) for supporting this project by providing all the data related to Porto's bus network, thus making it possible to evaluate the solution with real data.

References

1. Avelar, S., Hurni, L.: On the design of schematic transport maps. Int. J. Geogr. Inf. Geovis. **41**(3), 217–228 (2006). https://doi.org/10.3138/A477-3202-7876-N514
2. Baudisch, P., Good, N., Stewart, P.: Focus plus context screens - combining display technology with visualization techniques. In: Proceedings of the International Symposium on User Interface Software and Technology, UIST 2001, pp. 31–40 (2001). https://doi.org/10.1145/502348.502354
3. IBM: The Client/Server model (2019)
4. International Cartographic Association: History of ICA (2019). https://icaci.org/research-agenda/history/
5. Klippel, A., Kulik, L.: Using grids in maps. Theory and application of diagrams. In: Proceedings of First International Conference, Diagram 2000, Edinburgh, Scotland, UK, 1–3 September 2000, pp. 486–489 (2000)
6. Klippel, A., Richter, K.F., Barkowsky, T., Freksa, C.: The cognitive reality of schematic maps. In: Meng, L., Reichenbacher, T., Zipf, A. (eds.) Map-based Mobile Services: Theories, Methods and Implementations, pp. 55–71. Springer, Heidelberg (2005). https://doi.org/10.1007/3-540-26982-7_5
7. Maciel, F., Dias, T.G.: Challenging user interaction in public transportation spider maps: a cobweb solution for the city of Porto. In: Proceedings of IEEE Conference on Intelligent Transportation Systems, ITSC, pp. 181–188 (2016). https://doi.org/10.1109/ITSC.2016.7795551
8. Maciel, F.M.A.: Interactive spider maps for public transportation. Ph.D. thesis, Faculty of Engineering of University of Porto (2012)
9. Mourinho, J.: Automated generation of context-aware schematic maps: design, modeling and interaction. Ph.D. thesis, Faculty of Engineering of University of Porto (2015). https://hdl.handle.net/10216/79324
10. Sarkar, M., Brown, M.H.: Graphical Fisheye views of graphs. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 83–91 (1992). https://doi.org/10.1145/142750.142763
11. Sociedade de Transportes Colectivos do Porto: São João hospital spider map (2019). https://www.stcp.pt/fotos/spider_map