Embedding travel time cues in schematic maps

Cijsouw, M.

Award date:
2015

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Algorithms & Visualization Research Group

Embedding Travel Time Cues in Schematic Maps

Master's Thesis

Author: M. (Michel) Cijsouw

Supervisors: dr. H.J. (Herman) Haverkort
dr. M.A. (Michel) Westenberg

Eindhoven, August 2015

Abstract

A metro map is a type of diagram in which shapes of lines are simplified and geographic distances are often heavily distorted in order to provide a clear overview of a complex network. Because of these distortions, information about the original (real-world, geographically correct) map is lost. As a result, travelers could take unnecessary long routes when traveling from A to B. In order to tackle this problem we introduce several visual cues that aid the user in finding the fastest route from A to B.

Preface

First of all, I would like to thank dr. H.J. (Herman) Haverkort, dr. M.A. (Michel) Westenberg and dr.ir. H.M.M. (Huub) van de Wetering for providing me the opportunity to work with the Algorithms & Visualization group at the Technical University of Eindhoven. A special thanks to my supervisors dr. H.J. (Herman) Haverkort and dr. M.A. (Michel) Westenberg who have guided me through this project and provided me constant feedback during our weekly meetings. I would also like to thank R. (Roel) van Happen, a fellow student who did a project on a related topic and was also present in our weekly meetings, where he was able to provide feedback on my work. Furthermore, I thank my friends and family for providing me moral support and constant encouragement throughout the project.

Contents

Chapter 1

Introduction

Urban transportation networks are often complex in densely populated cities. Travelers may use so-called *metro maps* in order to find their way through these networks. A metro map is a type of diagram in which shapes of lines are simplified and geographic distances are often heavily distorted in order to provide a clear overview of the network. Metro maps can be divided into several subgraphs in which every subgraph is a so called *metro line*. These subgraphs are typically distinguishable by color, and traveling over such a subgraph never requires passengers to switch trains. In our project we do not allow cycles in these subgraphs. In 1933, Beck published a revolutionary London Underground Tube Map [1] that fully adheres to the notion of a metro map diagram. He believed that travelers were not too bothered about geographical accuracy, and were more interested in how to travel from one station to another, and where to change trains. The principles of Beck's map have been used in many other cities and countries, and over the years people (such as Roberts [2, 3]) have been improving the readability and efficiency of these maps. Over the past two decades, several algorithms were presented that generate such maps automatically [4, 5, 6].

1.1 Problem statement

Imagine that you are at station A, and you wish to travel to your destination D. Since there is no direct connection between A and D you would need to travel via an intermediate station: either B or C. Now imagine that in a schematic map the routes $\langle A \to B \to D \rangle$ and $\langle A \to C \to D \rangle$ are visually equal in distance, while in reality one of the routes is 10 minutes faster. This is an example of bad design, ideally the route that is visually the shortest should in reality also be the shortest route. An important trait of transportation networks is that transfers from one train to another take time, which means that sometimes taking a (small) detour is faster than taking the actual shortest path on the map. Furthermore, one can

imagine that the perceived time not only depends on the total length of the lines between A and B, but may depend on more factors such as line thickness, amount of line bends or other visual cues that aid the user in recognizing the fastest route between two stations. The question that now arises is: what metro map layout modifications and cues can we introduce in order to aid the user as best as possible in finding the fastest route between two stations?

1.2 Approach

When generating a metro map layout with modifications and extra visual cues, two different modification types can be distinguished. One modification type deals with adapting the input graph (i.e., change the location of the vertices) such that the graph adheres to a set of metro map design rules (see rules R1-R7 in Section 4.1). We call this a **schematization** of the network. Once we obtained a schematized graph we move on to the second modification type: **visualization**, in which we are interested in visualizing the schematized graph in such a way that the user is aided as best as possible when planning a route from A to B. In our approach this entails the introduction of varying line thicknesses such that important lines are rendered thicker than unimportant lines. We distinguish important lines from unimportant lines by measuring how often an edge is part of a fastest route, as described by our shortest paths algorithm in Section 3.3. Furthermore, we introduce dotted lines and arrow hints to aid the user in the right direction.

1.2.1 Schematization

Nöllenburg and Wolff published a paper [5] on automated schematic graph drawing using a *mixed-integer program* (MIP). Their method is able to generate (labeled) metro maps for small and medium-size metro networks that are of high visual quality and that can compete with official maps. We chose to use their algorithm as a basis for our program, because their algorithm provides great flexibility and linear programs of several thousands of variables can be solved rapidly by modern MIP solvers such as CPLEX [7]. We extended this schematization algorithm such that important and unimportant lines are taken into account in the constraints of the program. We integrated the following extensions:

- Important lines should be drawn with a minimal amount of line bends. Unimportant lines are allowed to be more winding.
- Hard constraint H3 (see Section 4.1) states that for each edge e, the line segment $\Gamma(e)$ must have length at least ℓ_e. This minimum length factor ℓ_e is given for each edge by the input graph G. We modified the ℓ values such that they take the edge thickness into account. Lines with thick edges nearby will allocate more space in order to prevent that

a part of the drawing Γ will be too cluttered. Conversely, lines with only thin edges in their neighborhood do not require extra space allocation.

1.2.2 Visualization

In traditional metro maps the thickness of lines is consistent throughout the entire map. Although consistency may be beneficial for the readability of the map, we could introduce design rules that take advantage of varying line thicknesses. Consider Figure 1.1, and imagine that we want to travel from station A to station D:

Case (a). Since both routes $\langle A \to B \to D \rangle$ and $\langle A \to C \to D \rangle$ are visually equal in distance, it is unclear which of these two routes we should take. In practice (without any knowledge of the city or metro network) we would choose an arbitrary route, which could result in a needlessly longer travel time.

Case (b). The edges between stations are now annotated with weights, where the weight denotes the travel time in seconds from one endpoint to another. This gives the user enough information to plan his route, simply by adding up the weights of all possible routes. The route with the shortest total weight is the route with the shortest travel time. However, we do not want to put this tedious task on the user.

Case (c). The thickness of an edge corresponds to the importance of the edge in the network. If the user is aware of the fact that important lines (lines that are used most often in fastest routes) are drawn thicker than unimportant lines (lines that are rarely used in fastest routes), he or she could scan any given map at a glance and immediately determine the fastest route based on the thickness of the edges. Whether the introduction of this design rule actually improves the usability of the map is investigated in Chapter 7.

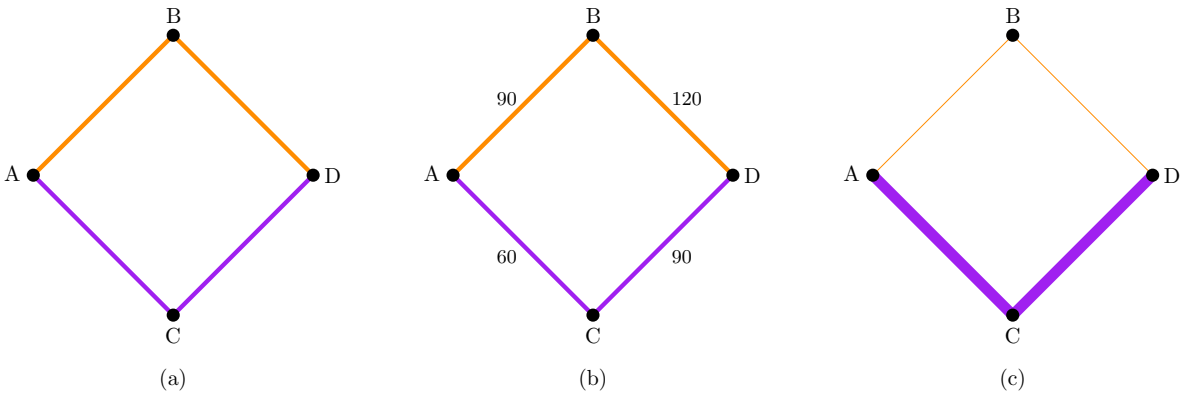


Figure 1.1: Different line thicknesses may give a better understanding of the map. Thicker lines denote faster routes.

Inspired by so-called *isochrone maps*, we have implemented an option to compute the visual cues relative to a given *starting station* instead of every pair of stations. In other words, instead of computing cues for every pair $\{(u, v) \mid u, v \in V\}$, we compute only for $\{(u, v) \mid v \in V\}$ where $u \in V$ is the starting station. The idea is that these maps provide more relevant information when standing at the starting station, e.g., the metro map could be a poster on a wall somewhere inside the station. If a transportation network consists of n stations, we could generate n different maps and display the maps in the stations that correspond to the starting station. Apart from varying line thicknesses we introduce more visual cues that can only be added when the starting station is known:

- Lines are replaced by dotted lines if they are not important for travelers at a particular starting station;
- Lines are annotated by arrow hints such that they point travelers to the right direction.

Although information may be more relevant, we do realize that a different map for every single station could be confusing for travelers, so we need to apply heavy modifications with caution. We believe that changes in the visual cues are acceptable, but heavy changes in the schematization part confuses travelers too much.

1.3 Outline

First, we discuss related work and earlier publications on the subject of metro map schematization, visualization and usability testing in Chapter 2. Chapter 3 describes how we collected the required data for our metro maps. In Chapter 4 we describe the schematization algorithm and our extensions, followed by our visualization algorithms in Chapter 5. Then we present our methods for our experiment in the form of an online user study in Chapter 6, followed by our results and discussions on the hypotheses in Chapter 7. Finally, we present our conclusions in Chapter 8.

Chapter 2

Related work

This chapter consists of three sections. In Section 2.1 we discuss earlier work that is done on automatically generating schematic metro maps. Then we discuss earlier work on the visualization of travel times in metro maps in Section 2.2. Finally, Section 2.3 describes existing papers on usability testing in metro maps.

2.1 Schematization

Automatic Metro Map Layout using Multicriteria Optimization. Stott et al. [6] described an automatic metro map layout system based on multicriteria optimization. A number of metrics are defined, which are used in a weighted sum to attempt to measure the aesthetic quality of the diagram. The station criteria are:

- *Angular resolution.* The angles of incident edges at each station should be maximized.
- *Edge length.* The edge lengths across the whole map should be approximately equal to ensure regular spacing between stations.
- *Balanced edge length.* The length of edges incident to a particular station should be similar.
- *Line straightness.* Edges that form part of a line should, where possible, be collinear either side of each station that the line passes through.
- *Octilinearity.* Each edge should be drawn horizontally, vertically or diagonally at 45deg.

Apart from criteria that must be optimized, there is also a set of rules (hard constraints):

- *Bounding area restriction.* Restrict the movement of stations to be within a certain bounding area so that the final diagram will fit on the target display.
- *Relative position.* Enforce the relative position between adjacent stations.
- *Occlusions.* Avoid the introduction of occlusions of other edges and stations to ensure that a station is not moved so that it is not lying on top of any other station or edge, and that edges do not cross other edges or lie on top of any other station.
- *Edge ordering.* Preserve the (circular) ordering of edges incident to a station.

They used a hill climbing method that takes all criteria and rules into account. They take an input graph and snap all stations to a grid, such that there is at most one station at any point. For each station they calculate the *fitness* of the diagram (determined by a fitness function, a function that weighs all criteria and is aimed at reflecting how aesthetically pleasing the map is). They then search the points around a rectangle centered on the station at a given distance. Next, they move the station to the location that improves the fitness the most. This is performed for all stations in the diagram until no more improvements can be made.

Figure 2.1 shows a normalized published (official) map, Figure 2.2 shows an undistorted (geographically correct) map, and Figure 2.3 shows an automatically-drawn map of Mexico City generated with their layout method.

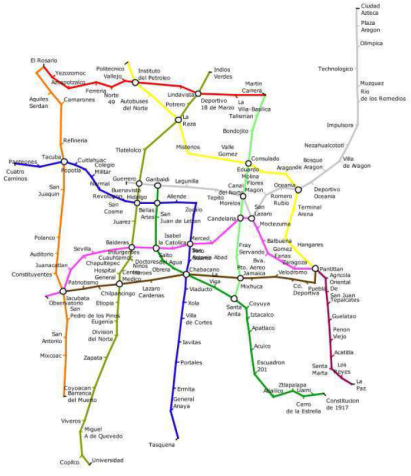


Figure 2.1: Mexico City normalized published map.

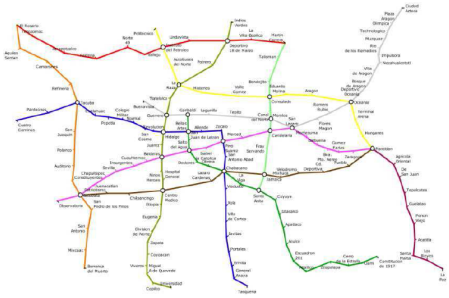


Figure 2.2: Mexico City undistorted map.

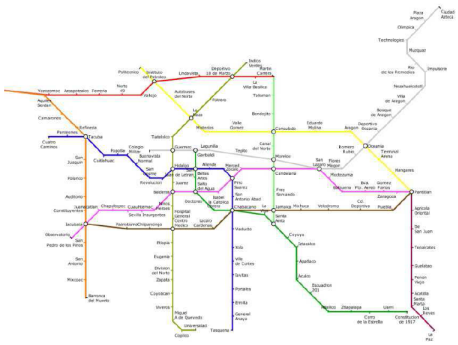


Figure 2.3: Mexico City automatically-drawn map.

The Metro Map Layout Problem. Hong et al. [8] used the following qualitative criteria for a good metro map layout:

C1: Each line is drawn as straight as possible.

C2: No edge crossings are present.

C3: No overlapping of labels occurs.

C4: Lines are mostly horizontal or vertical, with some at 45 degrees.

C5: Each line is drawn distinctly, with a unique color.

They tried five different metro map layout methods using various combinations of spring algorithms. The tools that they used are GEM [9], a modified version of PrEd [10] and a magnetic spring algorithm [11]. Each method can be briefly described as follows:

1. Method 1: The GEM algorithm.
2. Method 2: Simplify the metro map graph using a preprocessing step and use the GEM algorithm with edge weight.
3. Method 3: Simplify the metro map graph and use the GEM algorithm without edge weight. Then they use the modified PrEd algorithm with edge weight.
4. Method 4: Simplify the metro map graph and use the GEM algorithm without edge weight. Then they use the modified PrEd algorithm with edge weight, plus orthogonal magnetic spring algorithm.
5. Method 5: Simplify the metro map graph and use the GEM algorithm without edge weight. Then they use the modified PrEd algorithm with edge weight, plus orthogonal magnetic spring algorithm, plus 45 degree magnetic field forces.

Figures 2.4 through 2.8 show the resulting layouts of the different methods.

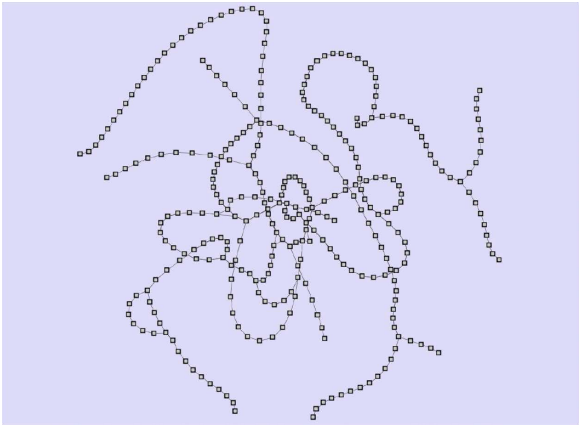


Figure 2.4: Sydney Cityrail NSW network produced by Method 1.

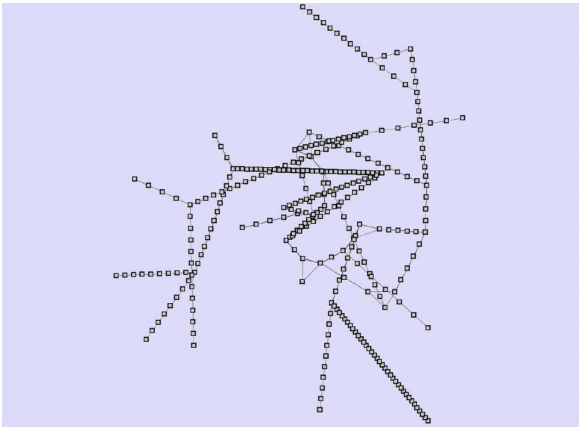


Figure 2.5: Sydney Cityrail NSW network produced by Method 2.

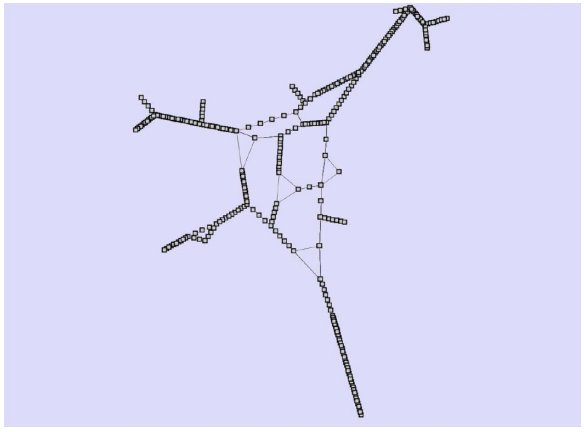


Figure 2.6: Sydney Cityrail NSW network produced by Method 3.

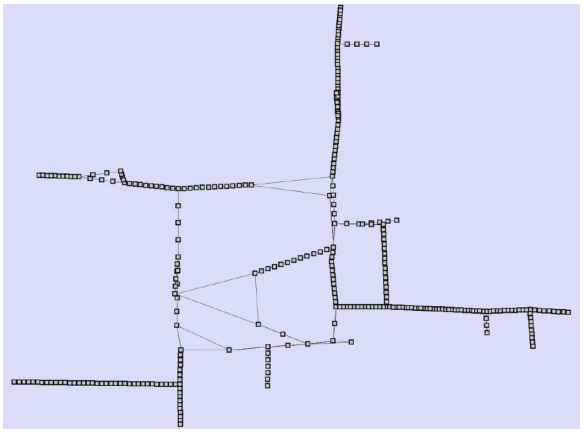


Figure 2.7: Sydney Cityrail NSW network produced by Method 4.

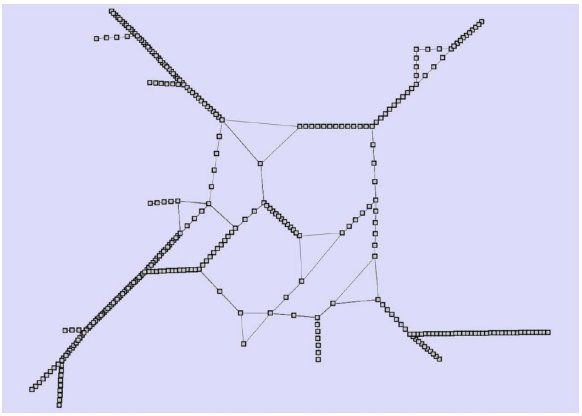


Figure 2.8: Sydney Cityrail NSW network produced by Method 5.

Octilinear Force-Directed Layout with Mental Map Preservation for Schematic Diagrams. Chivers and Rodgers [4] presented an algorithm that uses modified versions of two force based techniques: a spring embedder [12] and a magnetic spring model [11]. Consider the different stages in Figure 2.9. At the start only spring embedder forces are applied (S_1) — this stage runs until the energy level of the schematic falls below a set threshold; then there is a switchover in S_2 until only octilinear forces are applied (S_3). They then perform a post-processing step to straighten periphery line sections.

Furthermore, they used Delaunay triangulations to constrain node movement by proximity during optimization and thus help preserve the users' mental map. The generated triangulation edges are used during layout generation as a frame to hold nodes in place. Each edge is modeled as a Hookean spring, with a length at rest equal to its initial length, and a spring strength equal to a user defined value. This spring strength value can be varied in order to affect the level of mental map preservation.

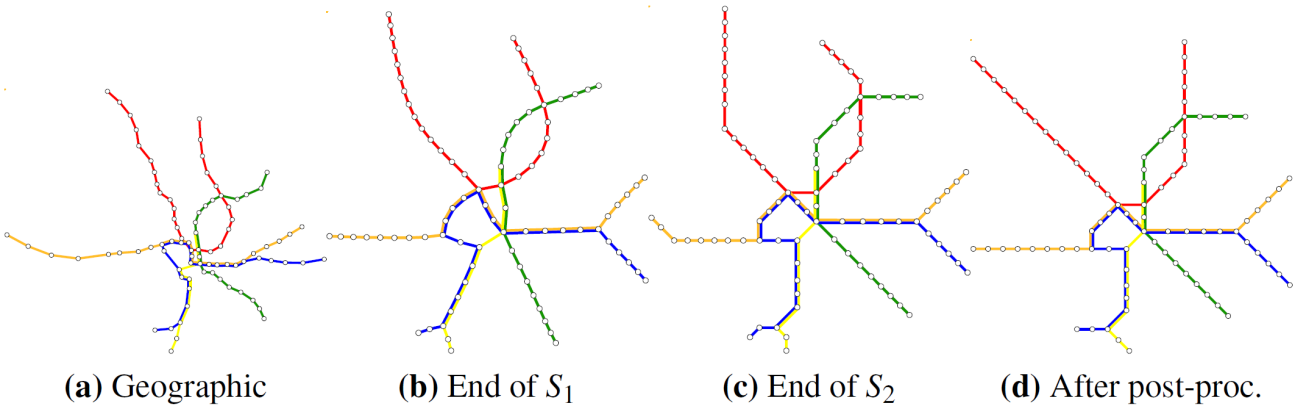


Figure 2.9: Washington metro map at key stages throughout the layout method.

Drawing and Labeling High-Quality Metro Maps by Mixed-Integer Programming. Nöllenburg and Wolff published a paper [5] on automated schematic graph drawing using a *mixed-integer program* (MIP). Their method is able to generate (labeled) metro maps for small and medium-size metro networks that are of high visual quality and that can compete with official maps. As described earlier, this algorithm provides a basis for our own schematization program. In Chapter 4 we describe their method in more detail.

What's Your Theory of Effective Schematic Map Design? Roberts [13] conducted an Internet-based survey in which he puts nine different specially-designed London Underground maps to the test. These nine maps are shown in Figure 2.10. People are asked to rate each of the nine designs, identifying maps that they believe are *hard to use*, *easy to use* or *neutral* compared with the rest and, subsequently *attractive*, *unpleasant*, or *neutral* compared with the rest.

The responses of the first 100 people to complete the survey are summarized in Tables 2.1 and 2.2. The octilinearity bias is noteworthy, with usability ratings disproportionately high compared with equivalent multilinear and curvilinear designs. Interestingly, the bias seems to be towards linear maps in general, with an overall belief that straight lines improve usability no matter what their angles. He reaches this conclusion by comparing attractiveness ratings with usability ratings. The linear maps are overall always rated as being more usable than they are attractive, and the curvilinear maps are always rated as being more attractive than they are usable. Furthermore, he states that *usability* is uncorrelated with map *engagement*: a person's opinions concerning the usability and attractiveness of a design. Map engagement is determined by subjective measurements such as evaluation questionnaires or choices between different designs. The lack of association between usability and engagement means that people can prefer maps that are difficult to use, and reject maps that are easy to use.

Usable?	Geographical	Compact	Stylised
Multilinear	24% Easy 28% Neutral 48% Hard	18% Easy 27% Neutral 55% Hard	33% Easy 39% Neutral 28% Hard
Octolinear	42% Easy 34% Neutral 24% Hard	37% Easy 32% Neutral 31% Hard	89% Easy 7% Neutral 4% Hard
Curvilinear	13% Easy 29% Neutral 58% Hard	4% Easy 25% Neutral 71% Hard	17% Easy 29% Neutral 54% Hard

Table 2.1: Usability ratings of the first 100 people to complete the Internet-survey.

Attractive?	Geographical	Compact	Stylised
Multilinear	9% Attr 32% Neutral 59% Unpl	8% Attr 20% Neutral 72% Unpl	23% Attr 35% Neutral 42% Unpl
Octolinear	33% Attr 42% Neutral 25% Unpl	36% Attr 37% Neutral 27% Unpl	75% Attr 21% Neutral 4% Unpl
Curvilinear	22% Attr 40% Neutral 38% Unpl	25% Attr 22% Neutral 53% Unpl	34% Attr 34% Neutral 36% Unpl

Table 2.2: Attractiveness ratings of the first 100 people to complete the Internet-survey (Attr = attractive, Unpl = unpleasant).

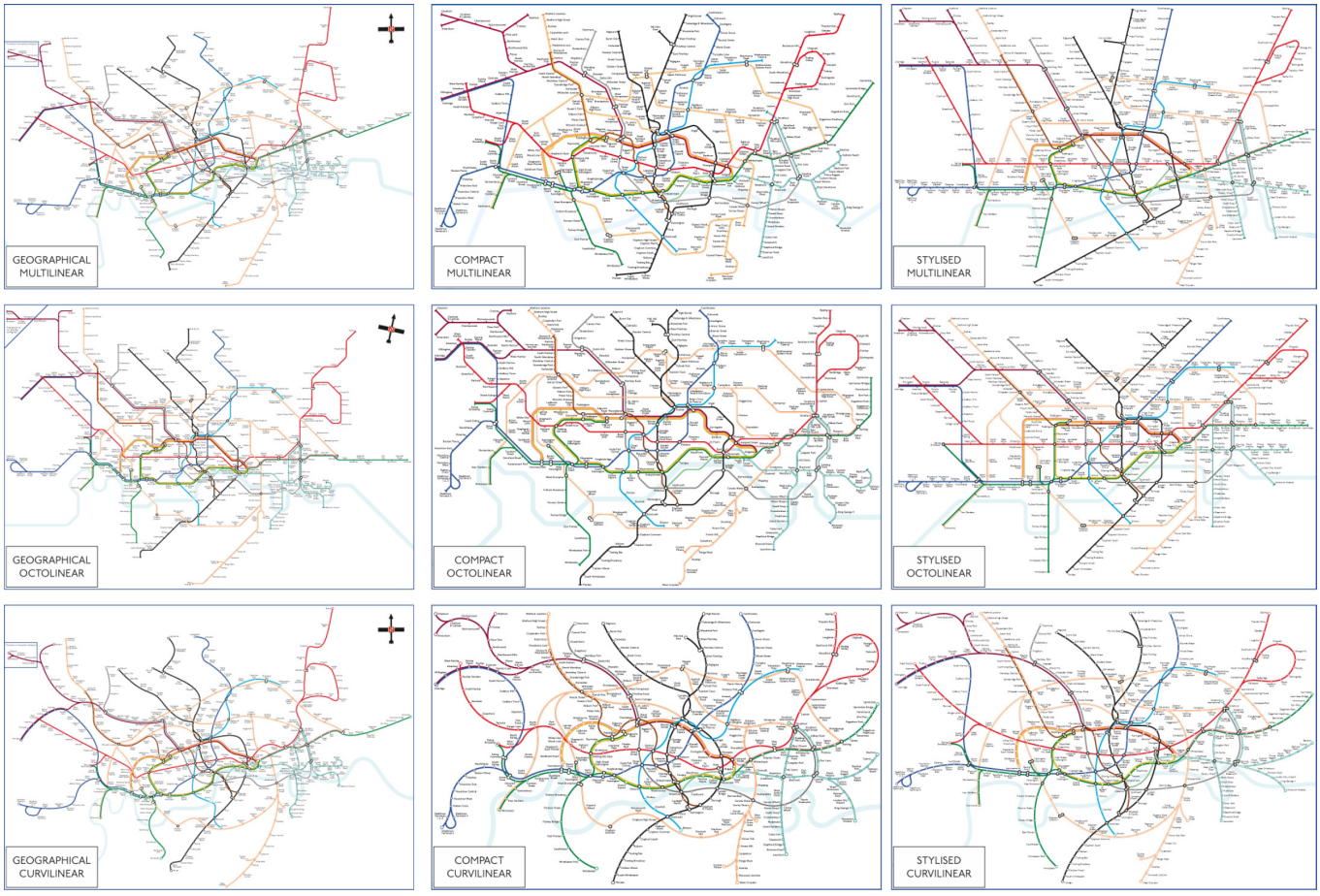


Figure 2.10: Nine maps used in the survey.

2.2 Travel time visualization

Embedding Cues about Travel Time in Schematic Maps. Haverkort [14] proposed eight possible solutions to give a user visual cues about which connections in a map are fast and which connections are slow in terms of distance on the map per minute of travel time. These solutions are described by Haverkort as follows:

1. *Width adjustment.* The width of each line segment is a sublinear function of its speed: the faster the wider.
2. *Shape adjustment.* The wiggliness of each line section is inversely proportional to its speed: slow lines get many serpentines, medium-slow lines a few slight swerves, medium-fast lines only an occasional bend, and fast lines follow a smooth course.
3. *Low-speed markers.* Serpentine-like markers are placed on slower lines, each marker signifying a five minutes' delay as compared to a typical medium-fast line.
4. *Heat map.* The brightness of the background is related to the speed of travel.
5. *Warped grid.* The map shows a regular grid of parallels and meridians at approximately equal intervals that is distorted in a way that is consistent with the mapping from geographical locations of stations to locations on the map.
6. *Checkerboard.* Same as above, clarified by shading the grid squares in a checkerboard pattern.
7. *Subdivision.* The map is divided into zones, such that, as long as one stays on the same train, a zone boundary is crossed once every 20 minutes. This causes zones to have small diameter on the map where speeds are low, and large diameter where speeds are high.
8. *Blob collection.* Same as above, but instead of drawing the boundaries between zones, a loosely packed collection of blob-shaped zones is drawn.

In his paper he visually worked out an example for every different solution (by hand), as well as combinations of different solutions. These visualizations are omitted here, but can be seen in [14]. This paper was the incentive for our own project, focusing on automatically generating maps that adhere to the *width-adjustment* proposal.

Your Favorite Traffic Map is Lying to You. An article by Stevens [15] discusses the fact that conventional traffic maps do not actually tell you how long it will take to get from point A to point B. Not only do traffic maps fail to communicate the information you are after, the information they do communicate can be very misleading. An example is shown in Figure 2.11. This problem is: Which speeds are within the yellow category? How about

the red category? Unless you have worked with the raw data, or dug deep into the chasm of documentation behind it, you will not be able to answer that question correctly.

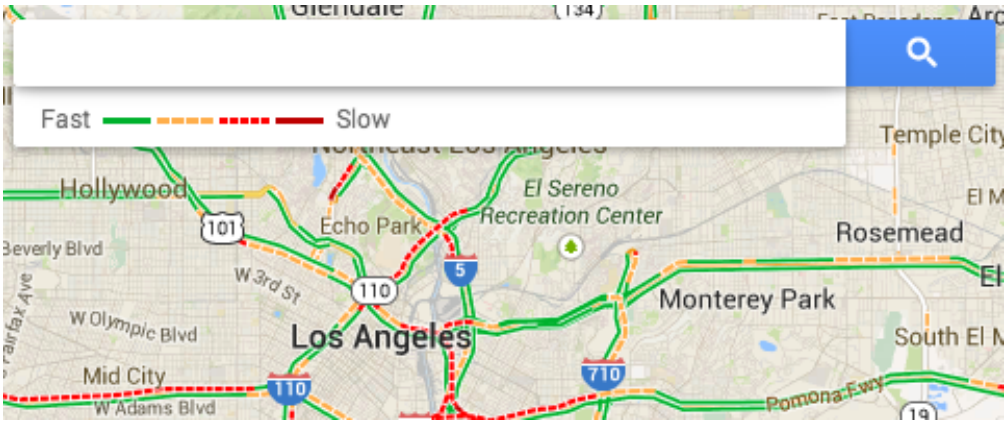


Figure 2.11: A map-based mystery: How fast is green?

He proposes two solutions for better traffic maps (See Figure 2.12):

1. Fix segment lengths to a standard unit of distance. When using the red-yellow-green color scheme, you forfeit accurate velocity estimation. The least you can do, then, is to make it easy for users to estimate distances. By fixing the colored segments to a fixed length (e.g., 5 miles, depending on map scale), you enable users to estimate distances very quickly and accurately. This is the *equal-interval approach*.

2. Show temporal, not geographic, segments. Instead of showing travel times indirectly, through velocity and color, travel times can be embedded directly into the road network. This is the *fixed-minute method*, because each cartographic symbol (or segment) represents some number of minutes of travel time. In the example, each purple 'chevron' symbol represents 1 minute of travel time. The darker, bolder symbols represent 5 minutes.

Figure 2.12: Two improvements (middle and right) to the conventional traffic map (left).

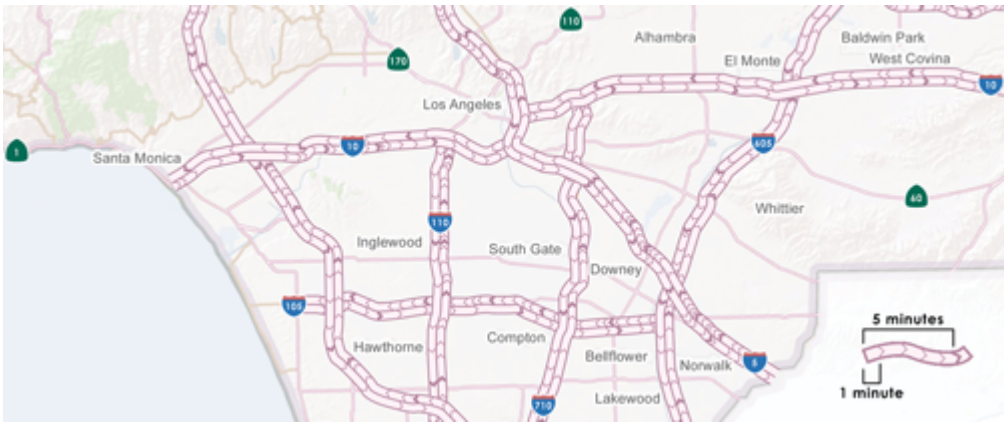


Figure 2.13: Fixed-minute method on a traffic map.

2.3 Usability testing

Automatic Metro Map Layout using Multicriteria Optimization. The empirical study of Stott et al. [6] shows that, for some of the maps tested, the layouts produced by their method can be considered better for route planning than both published and undistorted layouts. The users in their experiment were given a set of questions that takes 20 minutes to complete. These questions involved completing simple tasks on either of the following three types of maps:

- A normalized published (official) map — see Figure 2.1;
- An undistorted (geographically correct) map — see Figure 2.2;
- An automatically-drawn map generated with their layout method — see Figure 2.3.

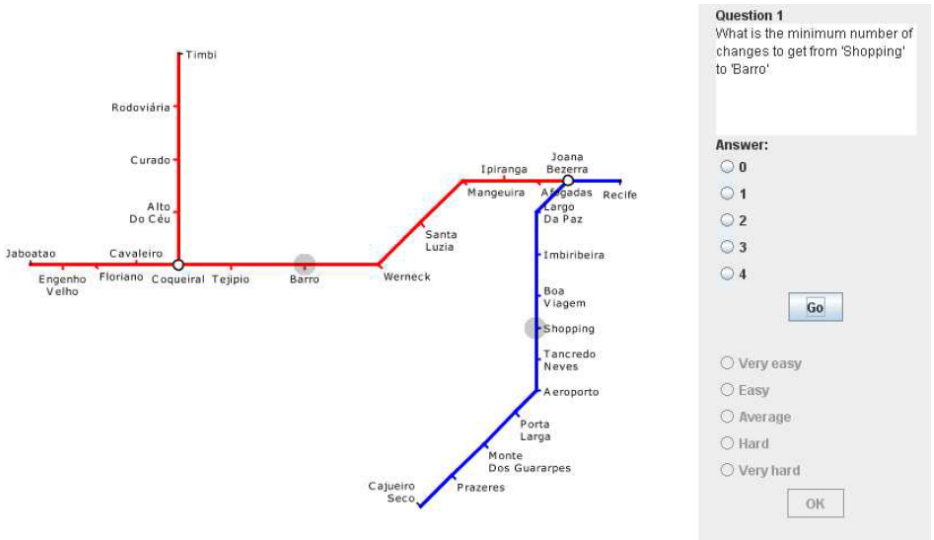


Figure 2.14: Screenshot of the software application used to conduct the empirical evaluation.

From the examples they published in their appendix we can see two types of tasks: one type involves counting the number of intermediate stations between station A and B, the other type of task involves counting the minimum number of changes to get from A to B.

A total of 43 subjects participated in their study, and they split the set of subjects into three balanced groups: I, II and III. Each group received the same questions in exactly the same order but they only saw one variant of each map per group for each question. The map variants were distributed evenly amongst the groups. A screenshot of the application that was used to conduct the empirical evaluation can be seen in Figure 2.14.

In their study they claim to have found some quantitative evidence to assert that the automatically drawn maps helps users to find routes more efficiently than the alternatives.

Metro Map Usability - an Experiment on the Usability of Automatically Generated Metro Maps. Ploum [16] conducted a user study on the usability of schematic metro maps. Using the algorithm by Nöllenburg and Wolff they generated a set of schematic maps. They distinguish the schematization types R4-optimized, R5-optimized, R6-optimized and Totally-optimized. The first three types correspond to the design rules R4, R5 and R6 by Nöllenburg and Wolff. The last type weighs all rules equally in the cost function and tries to optimize the map as much as possible.

Map types:

- R4-optimized: keep the number of bends along a given metro line small, and use obtuse rather than acute angles.
- R5-optimized: preserve the relative position between neighboring metro stations, that is if one station is located further north than another station, make sure that this is the case on the metro map as well.
- R6-optimized: keep the total edge length of the network small, this prevents the map from becoming too large.
- Totally-optimized: minimize the weighted sum of all rules: R4, R5 and R6.

Furthermore they also include a geographically correct map. Figures 2.15 through 2.19 show the different types of maps they used in their tests. Besides a map of Eindhoven they also tested with a fictional network called Random City. This network is in fact a map of Vienna in which the labels of the stations are replaced with made up names. The test involves performing three common tasks on metro maps:

T1 Finding a station.

T2 Counting the fewest number of stops it takes to get from one station to another.

T3 Counting the fewest number of line changes needed to get somewhere.

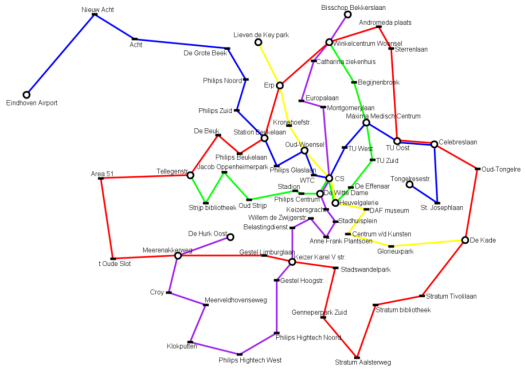


Figure 2.15: Geographic map of Eindhoven.

Figure 2.16: R4-optimized map of Eindhoven.

Figure 2.17: R5-optimized map of Eindhoven.

Figure 2.18: R6-optimized map of Eindhoven.

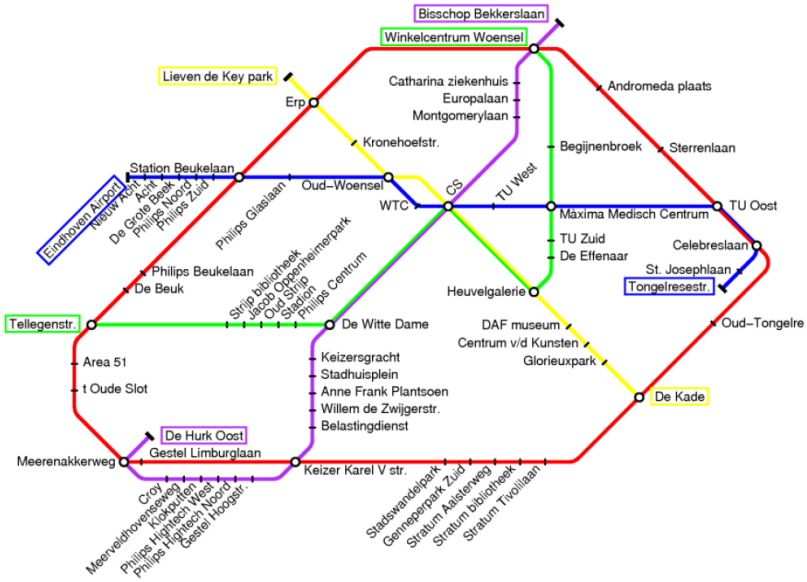
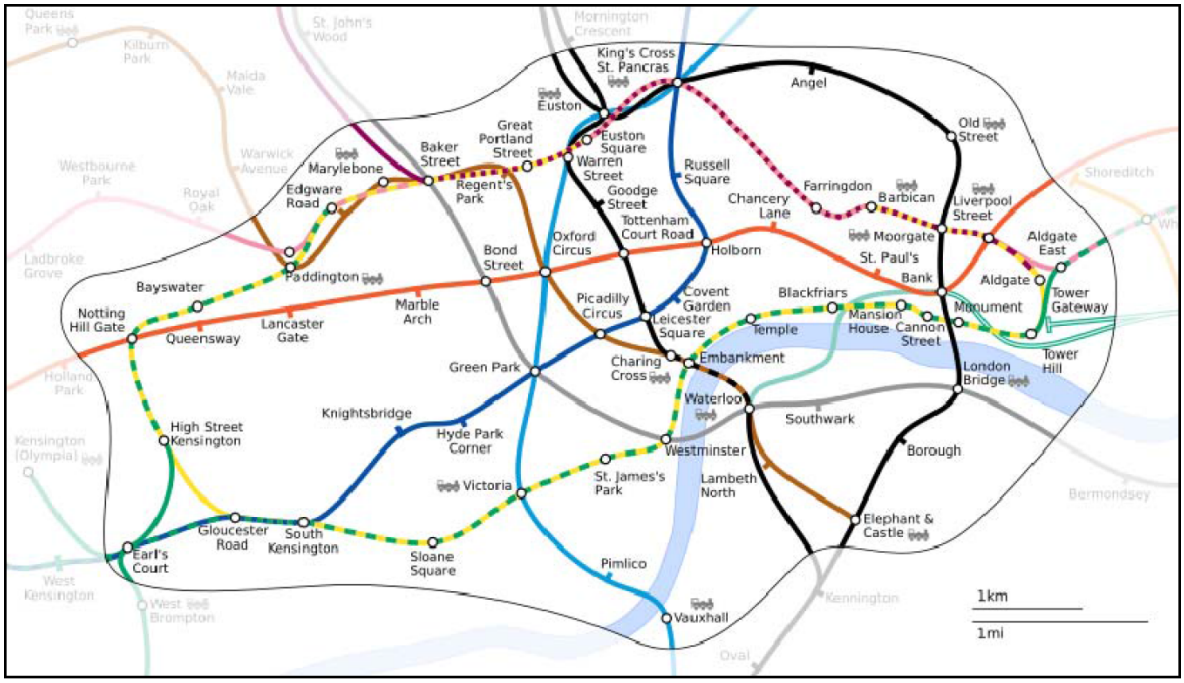


Figure 2.19: Totally optimized map of Eindhoven.

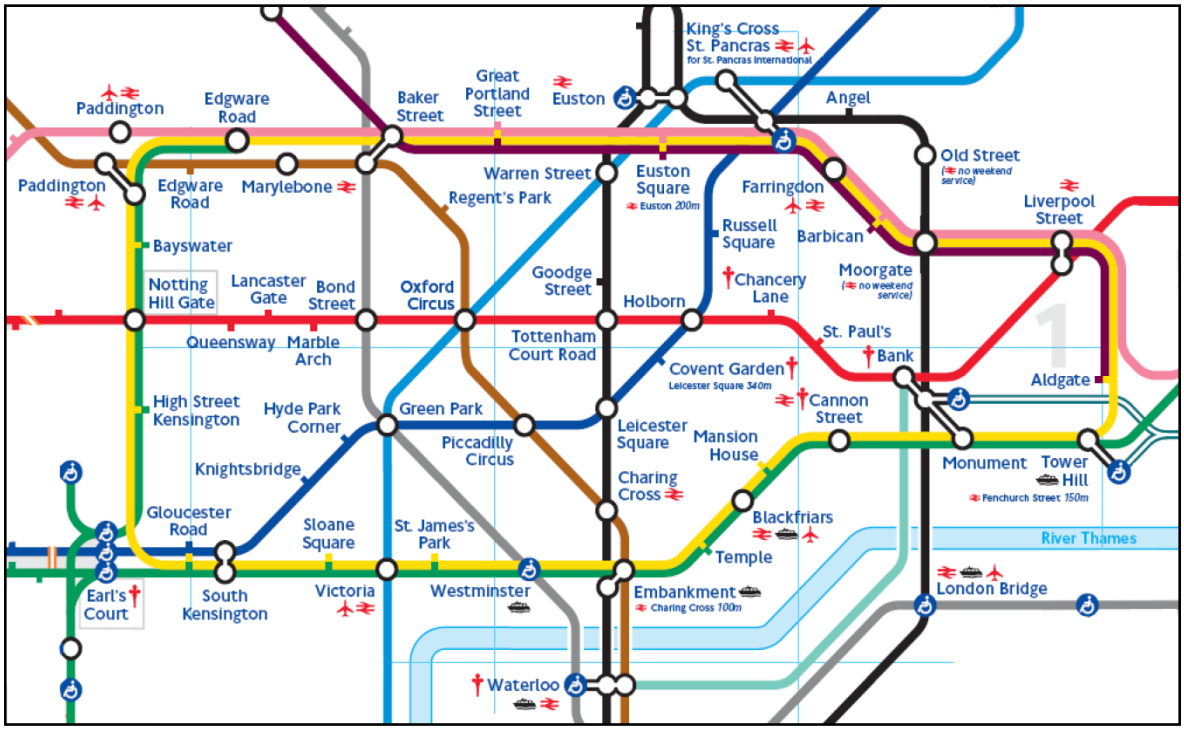
Most of the results of the experiment are not significant enough for any evident conclusions, but there are at least indications. Apparently the response times on questions about geographic maps is lower than on questions about schematized maps. There is also an indication that the response times of questions about maps that the user is familiar with are much lower than questions about maps that are unfamiliar to the user. However, it should be noted that they only tested with two types of maps, and these types may have been of different complexity. What their experiment did show is that subjects clearly have a higher preference for schematized metro maps over geographic ones (except if the schematization is extremely poor). Furthermore, the subjects rated the quality higher and the difficulty of the questions lower for schematized maps. It can thus be concluded with quite some confidence that schematized maps are appreciated by the subjects.

Mind the Map! The Impact of Transit Maps on Travel Decisions in Public Transit
Guo [17] investigates the effect of schematic transit maps on travel decisions in public transit systems. He used an actual map and an official tube map of London in his case study, as can be seen in Figure 2.20. The case study of London indicates that passengers often (mis)trust a transit map more than their actual experience; they often take a path that looks shorter on the system map but is longer in reality compared with alternative paths. They also try to avoid transfer stations when the connection on a map looks less convenient than it actually is.

The paper shows that the *codification* of transfer connections is important. Different codification can make a transfer look more or less convenient on a transit map than the reality, which will either decrease or increase the perceived transfer inconvenience for the corresponding stations. This paper identifies both situations in the London case study and quantifies this codification effect (in terms of the number of attracted or precluded transfers) for four major transfer stations: Baker St., Bank/Monument, Victoria, and Oxford Circus.



(a) Actual map.



(b) Tube map.

Figure 2.20: London Underground in Central London.

Objective versus subjective measures of Paris Metro map usability: Investigating traditional octolinear versus all-curves schematics. Roberts et al. [18] conducted an experiment on route planning on three different metro maps of Paris: (1) the Régie Autonome des Transports Parisiens (RATP) map, (2) an all-curves map designed by Roberts — displayed in Figure 2.21, and (3) the commercial map. Participants were asked to plan various complicated cross-Paris journeys. They used three different methods to conduct the experiments:

1. Method 1: The maps were printed on A3 paper and participants were given a starting station and a destination station. Their task was to draw the route that they would follow in order to travel from the given starting station to the given destination station.
2. Method 2: The same approach as the first method, but now the participants were required to first plan the route in their heads. When they indicated that they have a route in mind they were given a pencil to draw the route on the paper. The difference with the first method was the availability of separate timing measures for the two phases: planning time and drawing time.
3. Method 3: The same approach as the first method, but now there was a strict deadline (40 seconds) in which the participants had to complete the task.

They kept track of the following performance measures:

- Planning time;
- Drawing time;
- Estimated journey duration.

Across the three experiments, the all-curves map outperformed the RATP map in terms of mean planning time. For two of the three experiments, the all-curves map was superior taking into account all three measures of performance.

It is interesting to note that for Experiment 3, mean planning times are considerably shorter in their duration than for Experiment 1. Other than the deadline methodology, both involved a similar planning procedure. This faster planning for Experiment 3 is not associated with any overall increase in invalid routes, and suggests that for tasks without a deadline procedure, times may be raised unnecessarily, perhaps due to participants double-checking their plans or considering more alternatives. Even so, this did not affect the relative usability of the maps, and for the common journeys between Experiments 1 and 3, there is little to suggest that the additional planning time for Experiment 1 resulted in better journeys. However, the time deadline was exceeded for considerably more journeys for the RATP map than the all-curves map in Experiment 3.

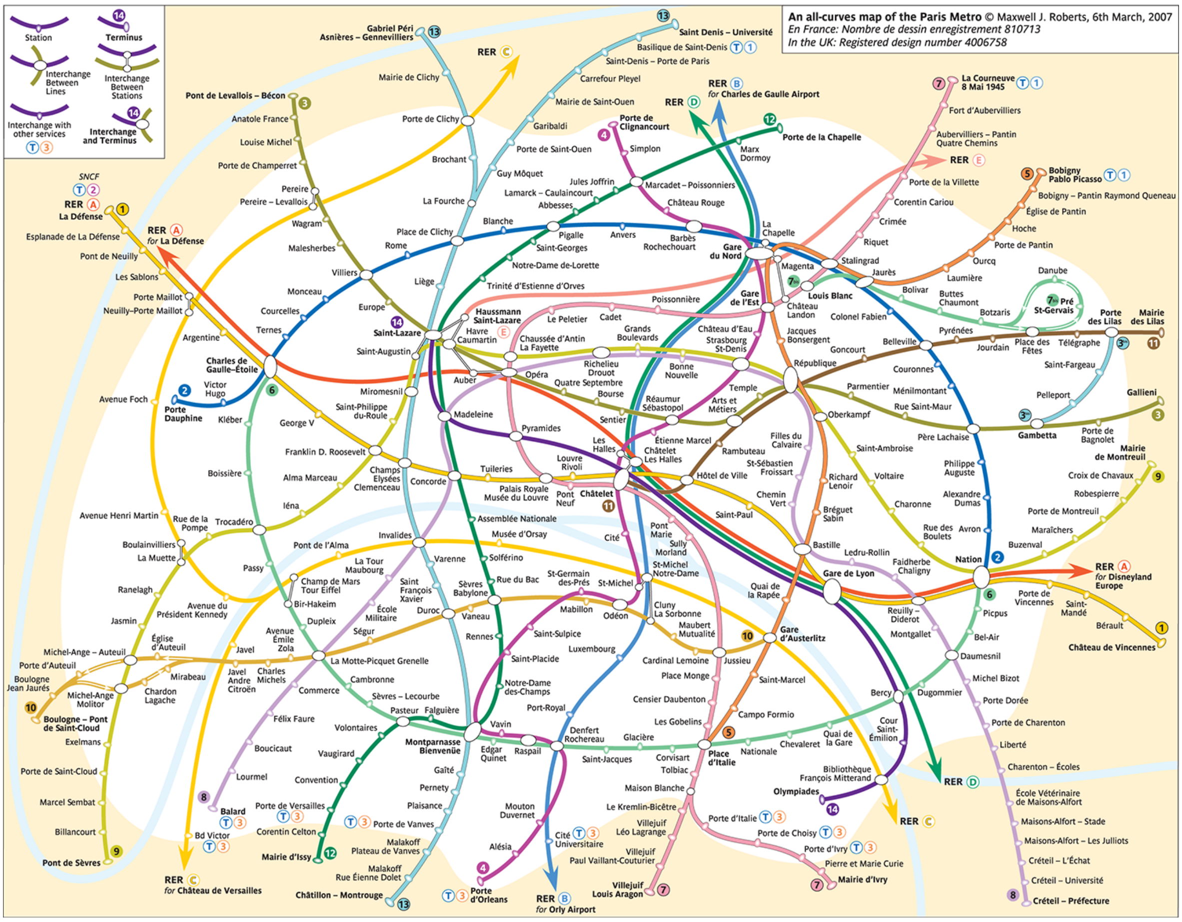


Figure 2.21: An all-curves map of the Paris Metro by Roberts.

Chapter 3

Data

This section describes our methods for retrieving and processing datasets for the cities London, Mexico, Stockholm, Sydney, Vienna and Washington. After processing, merging and post-processing, we possess the following data:

- Geographical locations (latitude and longitude) of the stations;
- A set of lines Ł, where $L \in$ Ł corresponds to a metro line of the underlying transport network;
- Traversal time of every edge $e \in E$ in seconds.
- The shortest path from starting station A to destination station B for every pair of stations (A, B).

3.1 Geographical data and timetables

3.1.1 London

The website *Transport For London* [19] provides live data feeds (e.g., live arrival times, live disruptions) as well as static data sets that are directly available for download. For extracting geographical data and timetables we are interested in the Journey Planner timetable feed. This feed contains up to date standard timetables for London Underground, bus, Docklands Light Railway (DLR) and river services. These different services are nicely split up in separate XML files, allowing us to filter out the XML files we are not interested in, i.e., the bus and river services.

Typically, XML files are structured consistently, allowing us to write a simple program which can extract all the desired information. A Journey Planner XML file contains a `<StopPoints>`

element which holds a list of stations in London. Each station is represented by a `<StopPoint>` element, in which we are interested in the following data:

- `AtcoCode` — a unique identifier used to refer to the station;
- `Descriptor/CommonName` — the name of the station;
- `Place/Location/Easting` — the geographic Cartesian x-coordinate of the station;
- `Place/Location/Northing` — the geographic Cartesian y-coordinate of the station.

Analyzing all `StopPoints` in our XML files results in a dataset which holds all the required information for the stations in London. The code below shows an example of the `StopPoint` Upton Park.

```
<StopPoint CreationDateTime="2013-08-29T00:00:00">
      <AtcoCode>9400ZZLUUPK2</AtcoCode>
      <Descriptor>
            <CommonName>Upton Park</CommonName>
      </Descriptor>
      <Place>
            <NptgLocalityRef>E0034715</NptgLocalityRef>
            <Location Precision="1m">
                  <Easting>541174</Easting>
                  <Northing>183765</Northing>
            </Location>
      </Place>
      <StopClassification>
            <StopType>RPL</StopType>
            <OffStreet>
                  <Rail>
                        <Platform />
                  </Rail>
            </OffStreet>
      </StopClassification>
      <AdministrativeAreaRef>000</AdministrativeAreaRef>
      <Notes>Upton Park</Notes>
</StopPoint>
```

What remains to be extracted is the set of edges, lines, and line colors. Each XML file represents a single line (e.g., Bakerloo Line, Central Line, Piccadilly Line, etc.), and the colors that correspond to these lines are consistent throughout all published metro maps. For extracting the set of edges we iterate through the `<JourneyPatternSections>` element and for every `<JourneyPatternTimingLink>` we encounter we extract the following information:

- `From/StopPointRef` — refers to the `AtcoCode` of one endpoint of the edge;
- `To/StopPointRef` — refers to the `AtcoCode` of one endpoint of the edge;
- `RunTime` — the time it takes to traverse the edge, formatted in ISO 8601's *duration*

format. The datasets of *Transport For London* round their timings in minutes, which results in a consistent format of the timings. For instance, `PT1M` denotes a period of 1 minute, and `PT4M` denotes a period of 4 minutes.

Analyzing all `JourneyPatternTimingLinks` in our XML files results in a dataset which holds all the required information about edges and lines. The code below shows an example of the `JourneyPatternTimingLink` (edge) that connects Hammersmith and Goldhawk Road.

```
<JourneyPatternTimingLink id="JPL_1-HAM-_-y05-320101-1-O-1-2">
      <From SequenceNumber="1">
            <Activity>pickUpAndSetDown</Activity>
            <StopPointRef>9400ZZLUHSC1</StopPointRef>
            <TimingStatus>PTP</TimingStatus>
      </From>
      <To SequenceNumber="2">
            <Activity>pickUpAndSetDown</Activity>
            <StopPointRef>9400ZZLUGHK1</StopPointRef>
            <TimingStatus>PTP</TimingStatus>
      </To>
      <RouteLinkRef>RL_1-HAM-_-y05-320101-O-1-1</RouteLinkRef>
      <RunTime>PT1M</RunTime>
</JourneyPatternTimingLink>
```

3.1.2 Mexico, Stockholm, Sydney, Vienna and Washington

Chivers and Rodgers recently published a paper on force-directed octilinear layouts [4]. An online resource [20] supporting this study is available at the University of Kent website, and provides raw metro map data for the cities Mexico, Stockholm, Sydney, Vienna and Washington in a convenient format. The provided data holds the following information:

- Station x- and y-coordinates;
- Station labels;
- Edge adjacency lists that represent lines between stations;
- Line colors.

The data files are stored in plain text format in which every line contains data for either a station or an edge. These two types can be distinguished by looking at the first four characters of the line string — if a line starts with `NODE` we know that we are dealing with a station, and if a line starts with `EDGE` we are dealing with an edge. Furthermore, the string after the fifth character of every line is a set of key-value pairs which are delimited by a colon.

In the snippet below we see that nodes (stations) contain the keys `label`, `x` and `y`. We parse these key-value pairs and store the values in our graph structure.

```
NODE:label=Leopoldau:x=494:y=173.8901
NODE:label=Grossfeldsiedlung:x=481.333:y=191.8901
NODE:label=Aderklaaer Strasse:x=489.333:y=219.2236
```

The following snippet shows that edges have a `label`, `color` and `adjlist` (adjacency list). The adjacency list is a sequence of node labels which form a path in our graph, where ∼ is the delimiter character. Furthermore, the `color` value is denoted in `r,g,b` format (red, green, blue), ranging from 0 to 255.

```
EDGE:label=U1:color=255,0,0:adjlist=Leopoldau∼Grossfeldsiedlung∼Aderklaaer Strasse∼
      Rennbahnweg∼Kagraner Platz∼Kagran∼Alte Donau∼Kaisermuehlen∼Donauinsel∼
      Vorgartenstrasse∼Praterstern∼Nestroyplatz∼Schwedenplatz∼Stephansplatz∼Karlsplatz∼
      Taubstummengasse∼Suedtiroler Platz∼Keplerplatz∼Reumannplatz
```

In some data sets the adjacency lists of edges only consist of a single edge, but there are multiple `EDGE` entries that belong to the same path. In this case we group the edges based on the edge label. For instance, from the snippet below we can extract the paths:

- Portales → Ermita → General Anaya → Tasquena, and
- Universidad → Copilco → Miguel A de Quevedo.

```
EDGE:label=2:color=27,1,229:adjlist=Portales∼Ermita
EDGE:label=2:color=27,1,229:adjlist=Ermita∼General Anaya
EDGE:label=2:color=27,1,229:adjlist=General Anaya∼Tasquena
EDGE:label=3:color=131,158,6:adjlist=Universidad∼Copilco
EDGE:label=3:color=131,158,6:adjlist=Copilco∼Miguel A de Quevedo
```

Sydney and Vienna timetables

The timetables for the Sydney metro network are taken from the official Sydney Trains website [21]. We looked up the timetables for each line and based on the time difference between two stations we computed the traversal time for a single edge. For instance, if a train arrives in Burwood at 17:21 and arrives at the next station (Strathfield) at 17:24, we know that it takes 180 seconds to traverse the edge (Burwood,Strathfield).

The timetables for the Vienna metro network are taken from an unofficial page [22] that lists the complete schedule of Vienna's network in a single page. For extracting the traversal times of every edge we used the same approach as for Sydney.

3.2 Graph format

Our graph structure adheres to the widely used Graph Markup Language [23]. GraphML is the XML-standard for storing graph-like diagrams. Nodes (stations) are represented by `<node>` elements, which hold data about the x-coordinate, y-coordinate and label. The code example below shows a node which represents Upton Park.

```
<node id="Upton Park">
      <data key="x">451.74</data>
      <data key="y">157.65</data>
      <data key="label">Upton Park</data>
</node>
```

Edges are represented by `<edge>` elements, which hold data about the metro lines that the edge is part of, and the time it takes to traverse the edge. The code example below represents the edge from Southwark to London Bridge Station, and is part of line `l9` (Jubilee Line). It takes 120 seconds to travel from one endpoint of the edge to the other.

```
<edge id="e96" source="Southwark" target="London Bridge Station">
      <data key="l9">true</data>
      <data key="time" line="l9">120</data>
</edge>
```

In many situations an edge is part of more than one metro line (see Figure 3.2). The code example below shows an edge that is part of three lines: `l1`, `l3` and `l8`.

```
<edge id="e271" source="Moorgate" target="Liverpool Street">
      <data key="l1">true</data>
      <data key="l3">true</data>
      <data key="l8">true</data>
      <data key="time" line="l8">60</data>
      <data key="time" line="l3">60</data>
      <data key="time" line="l1">60</data>
</edge>
```

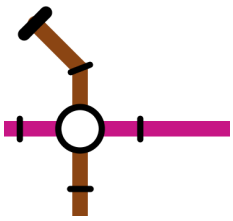


Figure 3.1: No parallel lines.

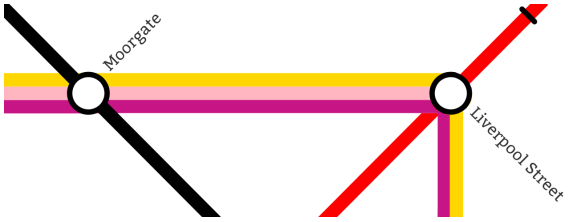


Figure 3.2: Three parallel lines between Moorgate and Liverpool Street.

Note that the keys and attributes that we are using (`x`, `y`, `label`, `l0...ln`, `time`) are not defined by default in GraphML. We use the following header in our XML file to represent the meaning of the structure in the context of metro maps:

```
<key id="x" for="node" attr.name="x_coordinate" attr.type="int"/>
<key id="y" for="node" attr.name="y_coordinate" attr.type="int"/>
<key id="label" for="node" attr.name="station_name" attr.type="string"/>
<key id="time" for="edge" attr.name="edge_time" attr.type="integer"/>
<key id="l0" for="edge" attr.type="boolean" color.r="34" color.g="139" color.b="34">
      <default>FALSE</default>
</key>
<key id="l1" for="edge" attr.type="boolean" color.r="165" color.g="42" color.b="42">
      <default>FALSE</default>
</key>
...
<key id="ln" for="edge" attr.type="boolean" color.r="208" color.g="32" color.b="144">
      <default>FALSE</default>
</key>
```

The attributes `color.r`, `color.g` and `color.b` represent red, green and blue values between 0 and 255 that define the color of the metro line. Furthermore, the `line`-attribute in a `<data key="time">` element is not part of our GraphML structure and is parsed manually by our Java GraphML reader.

3.3 Computing shortest paths

In order to distinguish important lines (lines that are used most often in fastest routes) from unimportant lines (lines that are rarely used in fastest routes) we compute all fastest routes from A to B for all pairs of stations (A, B). We realize that switching from one train to another takes time. This extra time is taken into account when computing the fastest route from one station to another. We incur a penalty of 300 seconds for transfers from one train to another. As a result, taking a (small) detour may sometimes be faster than taking a shorter route with more transfers.

We will compute the shortest path from A to B using Dijkstra's algorithm [24], but since this algorithm can not handle parallel lines we will first transform our graph G to a so-called *transfer graph* T_G. This is done by transforming every station s to a complete subgraph K_s. The number of vertices in K_s is equal to the number of different lines that run through s. This transformation is best described by giving an example. Consider the situation as seen in Figure 3.3. Every station with two or more adjacent metro lines is a potential transfer

point, hence we transform each of these stations to a complete subgraph K_s where every edge in K_s has weight W. Here, the weight of an edge denotes the time it takes to travel from one endpoint of the edge to the other endpoint. In our implementation we set the transfer time W to a fixed value of 300 seconds, which means that we do not take actual timetables or the size of the station into account. Note that these fixed weights are not mandatory for our approach — a nice improvement to our maps would be to let the weights depend on the time it would actually take to transfer from one line to another. As a result, our maps would reflect that it is sometimes better to avoid changing trains on large stations because on these stations it generally takes longer to transfer from one train to another due to longer walks.

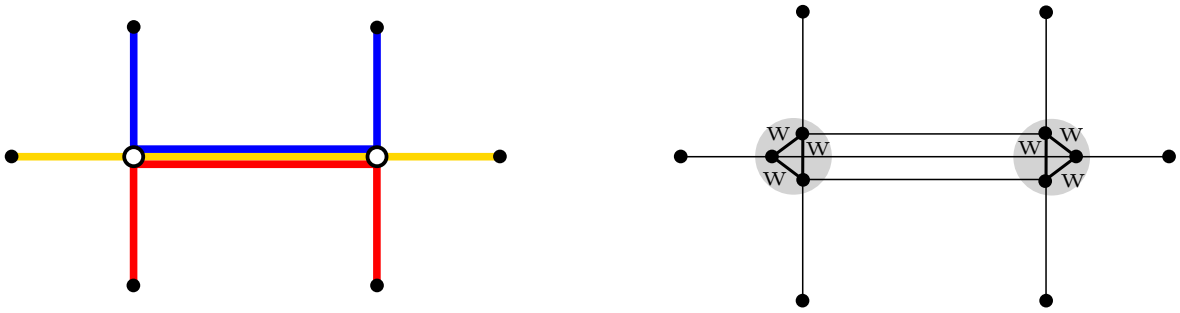


Figure 3.3: Transfer graph transformation.

The resulting *transfer graph* is a regular undirected graph on which we can run Dijkstra's algorithm for computing shortest paths. One problem that occurs when computing the shortest path from station A to station B is that we do not know which vertex $v_A \in K_A$ should be the starting vertex, and which vertex $v_B \in K_B$ should be the destination vertex. To solve this problem we compute the set S shortest paths for every pair (v_A, v_B) of stations. The path with the shortest travel time in S is the actual shortest path from A to B. The size of S is equal to $n_A * n_B$, where n_i denotes the number of vertices in K_i.

Chapter 4

Schematization

This chapter describes the schematization algorithms that we used to transform our geographically correct maps to schematic maps. Section 4.1 describes the original algorithm by Nöllenburg and Wolff, followed by our extensions to this algorithm in Section 4.2. Appendix A shows all 6 schematizations that were used in our project.

4.1 Raw (original) algorithm

As explained in Section 1.2.1 we use the mixed-integer program (MIP) by Nöllenburg and Wolff [5] as the basis for our schematization program. Their MIP is able to generate (labeled) metro maps for small and medium-size metro networks that are of high visual quality and that can compete with official maps. For these networks it can take several hours before the algorithm is finished, however, the first intermediate (suboptimal) results are often quickly generated and the optimal layouts differ only slightly from some of the earlier layouts, making it a useful tool to assist graphic designers. In their paper they identified the following design rules for metro maps:

R1. Restrict all line segments to the four *octilinear* orientations horizontal, vertical, and $\pm 45°$-diagonal.

R2. Do not change the geographical network topology. This is crucial to support the mental map of the passengers.

R3. Avoid bends along individual metro lines, especially in interchange stations, to keep them easy to follow for map readers. If bends cannot be avoided, obtuse angles are preferred over acute angles.

R4. Preserve the relative position between stations to avoid confusion with the mental map. For example, a station being north of some other station in reality should not

be placed south of it in the metro map.

R5. Keep edge lengths between adjacent stations as uniform as possible with a strict minimum length. This usually implies enlarging the city center at the expense of the periphery.

R6. Stations must be labeled and station names should not obscure other labels or parts of the network. Horizontal labels are preferred and labels along the track between two interchanges should use the same side of the corresponding path if possible.

R7. Use distinctive colors to denote the different metro lines. This means that edges used by multiple lines are drawn thicker and use colored copies for each line.

Let $G = (V, E)$ be a *plane* input graph, that is, a graph together with an embedding. We further assume that we know the geographic location $\Pi(v)$ of each vertex $v \in V$ in the plane. Let Ł be a *line cover* of G, that is, a set of paths of G such that each edge of G belongs to at least one element of Ł. An element $L \in$ Ł is called a *line* and corresponds to a metro line of the underlying transport network. We denote the pair $(G, Ł)$ by the *metro graph*. The task is now to find a drawing Γ of $(G, Ł)$ according to the rules (R1)-(R7). The design rules (R1)-(R5) translate to *hard constraints* (H1)-(H4) and *soft constraints* (S1)-(S3), and are defined by Nöllenburg and Wolff as follows:

H1. For each edge e, the line segment $\Gamma(e)$ must be octilinear.

H2. For each vertex v, the circular order of its neighbors must agree in Γ and the input embedding.

H3. For each edge e, the line segment $\Gamma(e)$ must have length at least ℓ_e.

H4. Each edge e must have distance at least $d_{\min} > 0$ from each non-incident edge in Γ.

S1. The lines in Ł should have few bends in Γ, and the bend angles ($< 180°$) should be as large as possible.

S2. For each pair of adjacent vertices (u, v), their relative position should be preserved, that is, the angle $\angle(\Gamma(u), \Gamma(v))$ should be similar to the angle $\angle(\Pi(u), \Pi(v))$, where $\angle(a, b)$ is the angle between the x-axis and the line through a and b.

S3. The total edge length of Γ should be small.

Hard constraints describe conditions that must always be met. If a combination of two or more hard constraints turns out to be unsolvable the MIP solver will quickly conclude infeasibility. Soft constraints are not crucial for feasibility — they determine the value of the target function. Many violations of soft constraints will result in a high target value. The goal of the MIP solver is to find a solution in which all hard constraints are met, in such a way that the violations of soft constraints are minimized.

4.2 Extensions

We extended the schematization algorithm of Nöllenburg and Wolff such that important and unimportant lines are taken into account in the constraints of the program. The extensions we applied are described in the next subsections.

4.2.1 Minimizing line bends on important lines

The soft constraint S1 states that lines in Ł should have few bends in Γ, and the bend angles ($< 180°$) should be as large as possible. Let $1 \leq \lambda_{(Si)} \leq 10$ be the factor that determines the importance of soft constraint Si. Different maps can be generated by playing around with the soft constraint factors $\lambda_{S1}, \lambda_{S2}, \lambda_{S3}$. For instance, a setting of $\lambda_{S1} = 10, \lambda_{S2} = 1, \lambda_{S3} = 1$ results in a map where line bends are penalized 10 times harder than the other constraints. As a result, a map is generated with almost all straight lines.

The idea behind minimizing line bends is that individual metro lines are easier to follow (see design rule R3). In our model we can use the notion of important and unimportant lines to determine which lines should be easy to follow and which lines should be less constrained in terms of bend minimization. In the original model the factor λ_{S1} is given as input and consistent for every line. We modified the logic in the algorithm such that λ_{S2} and λ_{S3} are static values, and λ_{S1} is a dynamic value that is computed separately for every edge. In Section 3.3 we explained how we compute shortest paths from A to B for all pairs of stations (A, B). From the set of shortest paths we can derive the amount of times A_e a specific edge e is used in any shortest path. We use this amount to determine the importance of an edge in the network. After experimental fine-tuning we determined the following factors result in an aesthetically pleasing schematic map in which important lines contain less line bends than unimportant lines:

$$\lambda_{S1} = \max\left(5 + \frac{A_e - \text{avg}(A)}{\max(A) - \text{avg}(A)} \cdot 10,\ \ 1\right),$$
$$\lambda_{S2} = 5,$$
$$\lambda_{S3} = 5$$

"Aesthetically pleasing" maps are the result of a healthy mix between factors S1, S2 and S3. The S1 factor should not be too overwhelming for the S2 and S3 factors, because if S1 is too overwhelming it would result in a map in which lines are straightened out too much. The impact of soft constraint S2 would be negligible and as a result would violate design rules R2 and R4 too heavily. Therefore we chose the weight of λ_{S1} in such a way that lines that are below average importance are assigned a weight between 1 and 5, and the lines that

are above average importance are assigned a weight between 5 and 15. We found an upper bound of 15 to be exactly on point, such that important lines are straightened out, but not too overwhelming.

Figure 4.1 shows a schematic map of Sydney generated by the raw (original) algorithm of Nöllenburg and Wolff. Note that thick lines denote important lines and thin lines denote unimportant lines. Here, it can be seen that the yellow line that runs through the center of the map is considered to be important, but since importance of lines is not taken into account the bend penalty is equal for all edges. In contrast, Figure 4.2 shows a schematic map of Sydney generated by our modified algorithm. Here, it can be seen that the thick yellow line contains less bends along the line and is therefore easier to follow according to design rule R3. We can also observe that thin lines are not necessarily straightened out as much as possible, this can be seen when we compare the bottom left parts of the maps.

A drawback of this approach can be seen on the very top-left of Figure 4.2. Here, two line bends are introduced at the end of the yellow line, however it would be better for the readability to straighten out this line (similar to Figure 4.1). This phenomenon is caused by the fact that edges close to a degree-1 station are considered unimportant, as they are rarely part of any fastest route. As a result the S1 factor is very low on these edges, which means that the impact on the objective value of the mixed-integer program is negligible.

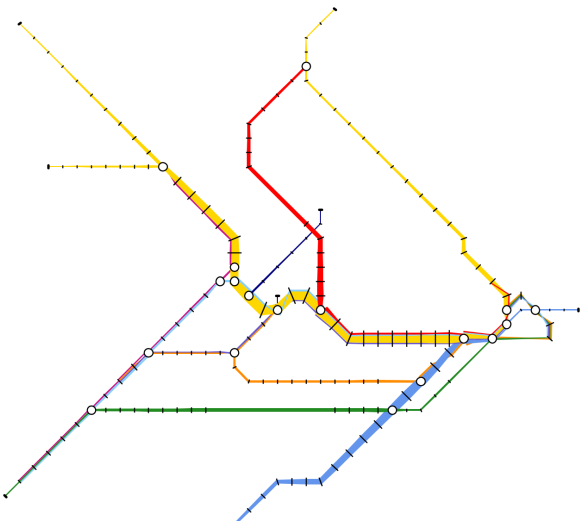


Figure 4.1: Schematic map of Sydney generated by the raw (original) algorithm of Nöllenburg and Wolff.

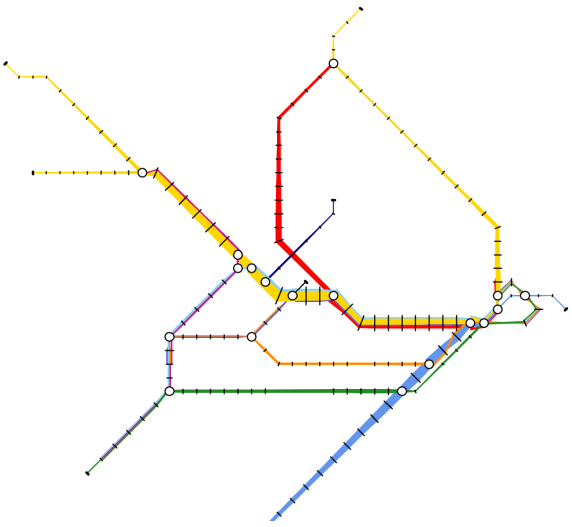


Figure 4.2: Schematic map of Sydney generated by the algorithm of Nöllenburg and Wolff, extended with line bend minimization on important lines.

4.2.2 Allocating more space in cluttered areas

In their paper, Nöllenburg and Wolff propose the following possible extension to their model:

> "It remains an open problem how to treat edges that are used by many parallel lines. Such 'thick' edges should be modeled as rectangles that actually consume space in the drawing. Analogously, stations on such thick edges must be modeled as disks or polygons rather than points."

Modeling large stations as disks would be easy to do, since a disk d can essentially be modeled as a single point p with the addition that each edge must have distance at least d_{radius} from p. This could have been implemented with not too many modifications to the algorithm. It would be interesting and more challenging to modify the algorithm such that rectangles are supported. This could have been done by taking the edges $e_1^r, \ldots, e_4^r$ of the rectangle r apart. For every edge $e_1^r, \ldots, e_4^r$ we can define a *half-plane* that does not intersect the interior of r. We would need to extend the model such that all other nodes and edges in the graph intersect at least one of these half planes. We do not want to modify the algorithm to this extent, because these modifications only provide a slightly more theoretically correct solution, but are of very little profit in practice because the influence of hard constraint H3 already makes sure that enough space is reserved for even the thickest edges in our visualizations. Furthermore, the introduction of even more constraints to the mixed-integer program only brings more complexity in the code and results in a larger linear program which takes longer to solve. Although space reservation is not a problem and parallel edges do not overlap in our visualizations, we do realize that cluttered areas can be detrimental for the readability of the map. Therefore, we introduced a relatively simple method for the allocation of extra space in cluttered areas. For every edge e that is adjacent to an intersection vertex v we compute the maximum line thickness $t_{\max}$ of the neighborhood of v. Now, we change the minimum length ℓ_e of edge e by

$$\ell_e = \ell_e + t_{\max} \cdot c_1 + e_{\text{mult}} \cdot c_2,$$

where c_1 is a constant that influences $t_{\max}$, and c_2 is a constant that influences the *multiplicity* e_{mult} of e. Multiplicity means the amount of edges in a parallel line — lines with high multiplicity generally require more space. In general, the values $c_1 = 0.1$ and $c_2 = 0.2$ result in aesthetically pleasing maps where areas that were cluttered before (mostly around intersections) are now more sparse. Figure 4.4 shows a part of a schematic map of Sydney generated by the raw (original) algorithm of Nöllenburg and Wolff. Here it can be seen that the stations and lines are cluttered around intersections. It may be hard to follow individual lines through these cluttered areas. Figure 4.5 shows a part of a schematic map of Sydney generated by our modified algorithm. Here, it can be seen that more space is allocated around the intersection stations, making it easier to follow individual lines (according to design rule R3).

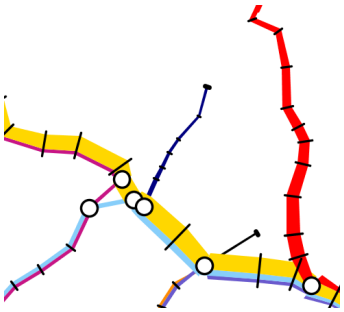


Figure 4.3: Original embedding of Sydney.

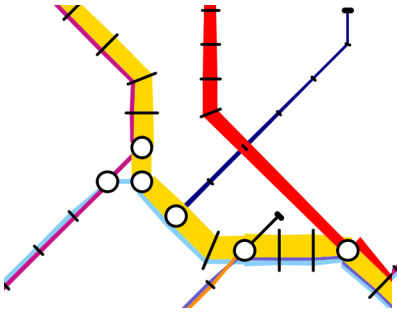


Figure 4.4: Schematic map of Sydney generated by the raw (original) algorithm of Nöllenburg and Wolff.

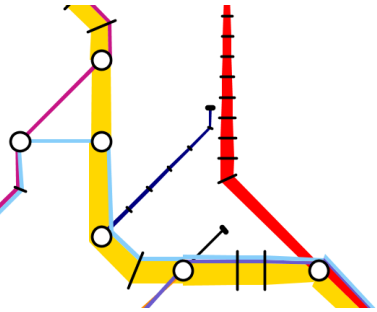


Figure 4.5: Schematic map of Sydney generated by the algorithm, extended with more space allocation in cluttered areas.

We realize that the introduction of the hard constraint that forces more space allocation in certain areas influences the soft constraints S1, S2 and S3. An example can be seen in Figures 4.4 and 4.5 — in the first figure we observe that the dark blue line crosses the red line, but in the second figure the dark blue line and red line do not cross. If we look at the original (geographically correct) embedding of Sydney's metro network we observe that these lines do not cross either. This indicates that the allocation of extra space could be beneficial for soft constraint S2, however in some cases it can also be detrimental for soft constraint S2. If we consider Figures 4.7 and 4.8 we observe that the raw algorithm does a good job in maintaining the relative positions of the stations, but our modified version introduces a line bend of 90 degrees (between the two intersections) that is completely unnecessary. In this case it would be better to draw these line segments vertically.

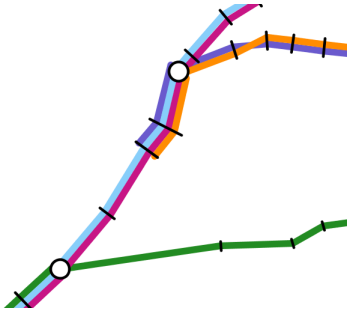


Figure 4.6: Original embedding of Sydney.

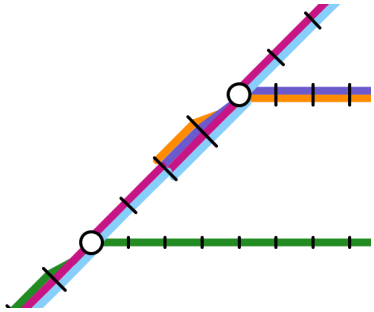


Figure 4.7: Schematic map of Sydney generated by the raw (original) algorithm of Nöllenburg and Wolff.

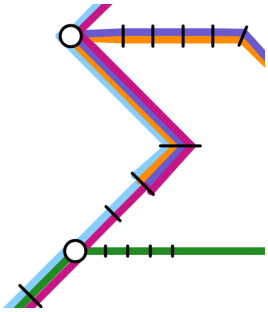


Figure 4.8: Schematic map of Sydney generated by the algorithm, extended with more space allocation in cluttered areas.

Chapter 5

Visualization

This chapter describes the different visualizations for our schematic maps. In Section 5.1 we describe characteristics that apply to all map types, followed by 6 different visualization types described in Section 5.2.

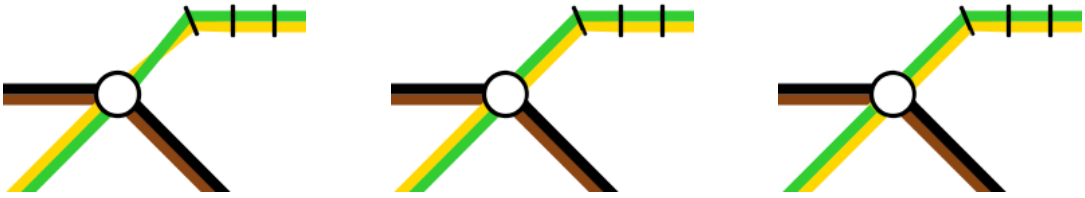
5.1 Characteristics

Our visualizations have the following characteristics:

- A regular (intermediate) station is drawn as a small black line perpendicular to the two adjacent edges. This is done because when we draw a regular station as a single dot it is not clear that all parallel lines of an edge run through that station. Drawing the station as a small black line covers the full width of an edge, such that it is more clear that all parallel lines of the edge actually run through this station. If a station happens to be exactly on a line bend the two adjacent edges have different angles; in that case the angle of the small black line is computed by taking the average angle of the two edges plus 90 degrees.
- An intersection station is drawn as a white circle with a black border. This design choice is adopted from official maps, and we chose to maintain this design choice to reduce confusion among users that are already familiar with official maps.
- A station of degree 1 is called a *terminus* and is drawn slightly thicker. This helps the user identify that a line stops at that point.
- Over the years people such as Roberts [3] have been experimenting with different types of metro maps, such as octilinear maps, force-directed curvy maps, maps with lines that follow a hexagonal grid, and maps with curved lines based on concentric circles. We chose to use the schematization algorithm of Nöllenburg and Wolff that adheres to

the octilinear layout scheme, because octilinearity is also used in the official maps of the cities we are experimenting with. Our automatically generated maps with fixed line thicknesses and no extra visual cues generally do not differ much from official layouts (in terms of design choices and schematization), and therefore our different visualizations and the results of our user experiments can be applied to official maps too, which indicates the potential practical use of this project.

- Line colors are retrieved from the GraphML data file, and correspond to the colors that are being used in official metro maps of their corresponding cities. We chose to maintain the same color scheme as the official maps to reduce confusion among users that are already familiar with the official maps.
- Parallel line segments contain two or more individual line segments that are drawn adjacent to each other. When drawing parallel lines the ordering of the individual lines is a complex problem. The ordering must be consistent on a path of degree-2 vertices and when the paths split up the number of overlaps of the individual lines must be minimized. The algorithm of Nöllenburg and Wolff provides an algorithm for consistency in paths of degree-2 vertices, but does not guarantee that the number of overlaps of individual lines is minimized. Implementing an automated method for this problem is very complex. Swapping two lines in one place may introduce more overlaps elsewhere in the map, and there exist situations in which overlaps can not be avoided. Because of the complexity we decided to implement a quick and easy solution in our program, namely an option that allows us to interactively flip the ordering of parallel edges so that the visualization can be manually polished after it is generated.



(a) Undesired situation where the green line overlaps the yellow line.

(b) Solution with disconnected line segments.

(c) Solution with possible introduction of more problems elsewhere.

Figure 5.1: Overlapping problem visualized.

See Figure 5.1. Case (a) shows an undesired situation in which the green line overlaps the yellow line. In case (b) the overlapping problem is solved locally, but the yellow and green line segments are no longer *connected*. We decided to not implement this approach, because it may be too hard to follow an individual line when it runs through an intersection. Case (c) shows a solution that solves this problem, but could introduce even more overlapping problems further along the line segment on the bottom-left,

because the yellow line and green line swapped places.

5.2 Map types

5.2.1 Fixed thickness

Figure 5.2 shows a schematic map of Vienna with fixed line thicknesses and no extra visual cues. This map type has most similarities with official maps, since these maps also have fixed line thicknesses without any visual cues about travel times.

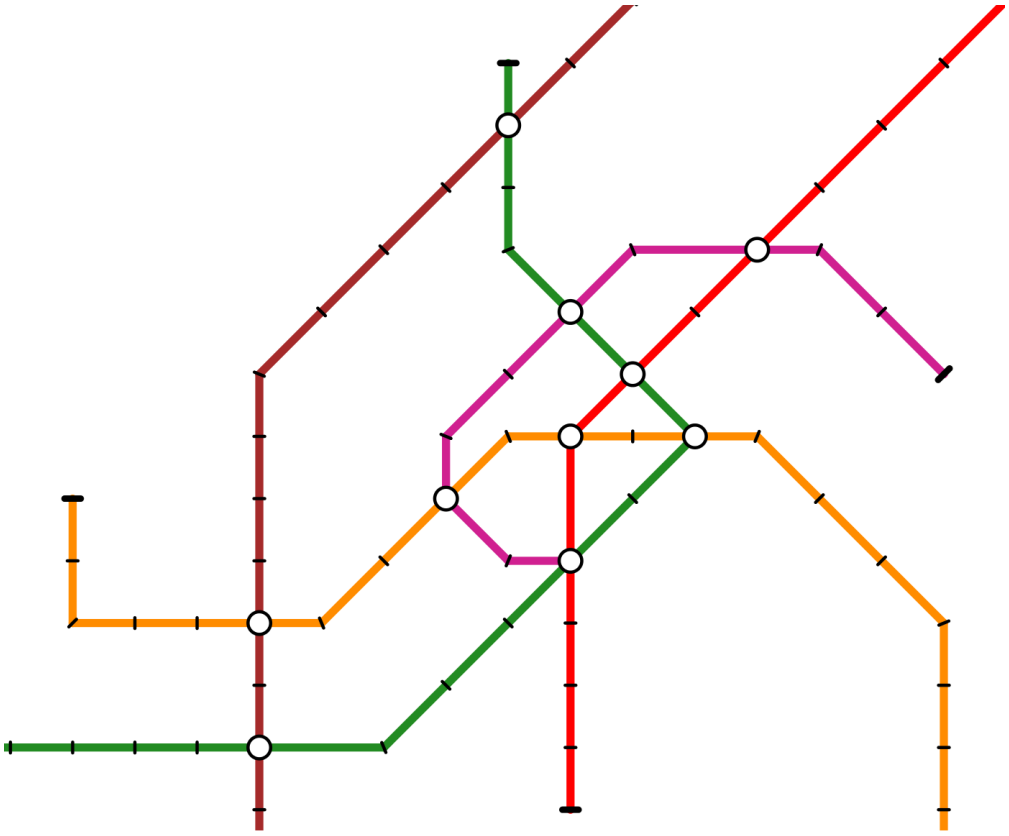


Figure 5.2: Fixed thickness

5.2.2 Fixed thickness + dotted lines

Figure 5.3 shows a schematic map of Vienna with fixed line thicknesses, where some lines are dotted instead of solid. In Section 3.3 we described how we compute the set S of shortest paths from A to B for every pair of stations (A, B). In this map type we require a given starting station A, marked with a red circle in Figure 5.3. Now, we can iterate over S and

extract the set S' of shortest paths from A to all possible destination stations. Next, we can iterate over all edges in S' in order to compute how often each edge is part of a shortest path. With this knowledge we identify which edges are not part of any shortest path, and we replace these edges by dotted lines. Travelers that go from starting station A to any arbitrary destination station must avoid the dotted lines, as traveling over these lines would mean that they did not take the fastest route to reach their destination. We chose to use dotted lines because this depicts *optionality* and the recognizable dotted pattern makes users aware of the fact that something is going on regarding these edges. An alternative is to omit unused lines completely, but we believe that this has too much impact on the mental map of the users since the maps would differ too much visually if we would compare maps with different starting stations.

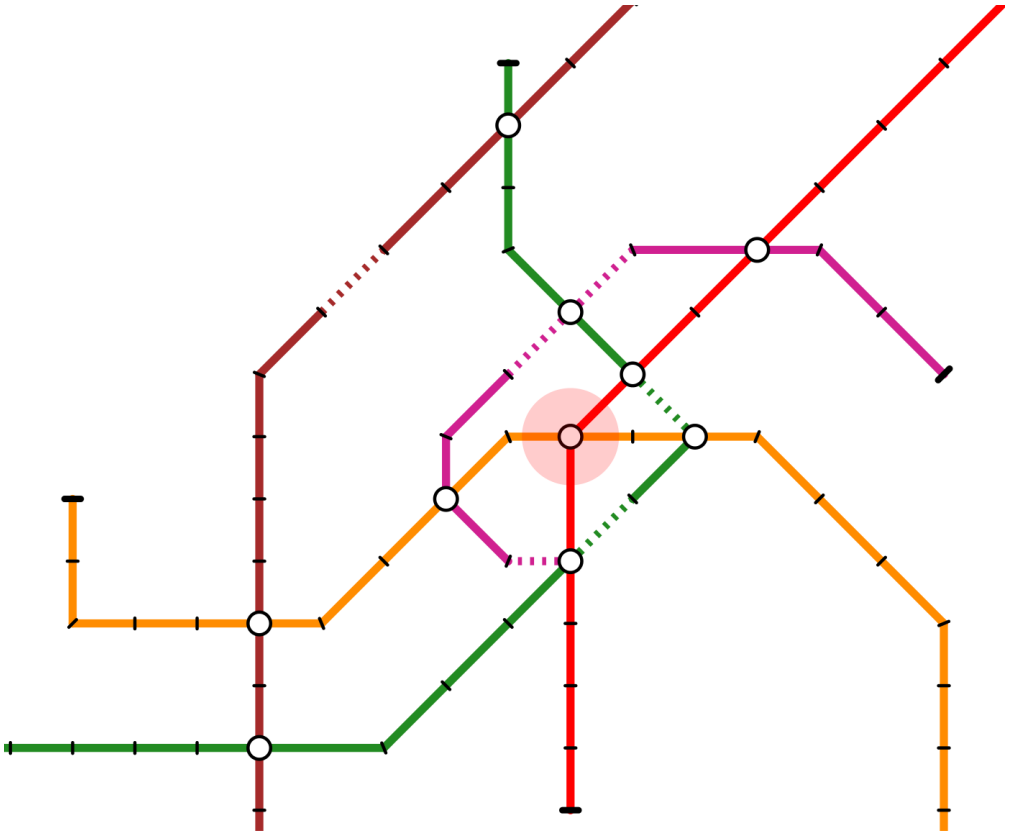


Figure 5.3: Fixed thickness + dotted lines

5.2.3 Fixed thickness + arrow hints

Figure 5.4 shows a schematic map of Vienna with fixed line thicknesses, where most lines are annotated with arrows. In Section 5.2.2 we described how we compute a set of shortest paths S' where every path starts at a given starting station A. In this type of visualization

we traverse all paths in S' and for every edge we pass we store the direction in which we are traversing. The result is a directed graph which is a tree rooted at A. Travelers that go from starting station A to any arbitrary destination station must only travel over line segments on which the arrow is pointing to the direction they are traveling. Lines that are not part of any shortest path are not annotated with an arrow and must be avoided. We chose to display these arrows with a white fill and a black border, such that they are clearly visible on edges of various colors. Also, arrows must point the user in the right direction without occluding the underlying map, hence the width of the arrows never exceeds the width of an edge, and the length of an arrow is equal to approximately half the length of the edge.

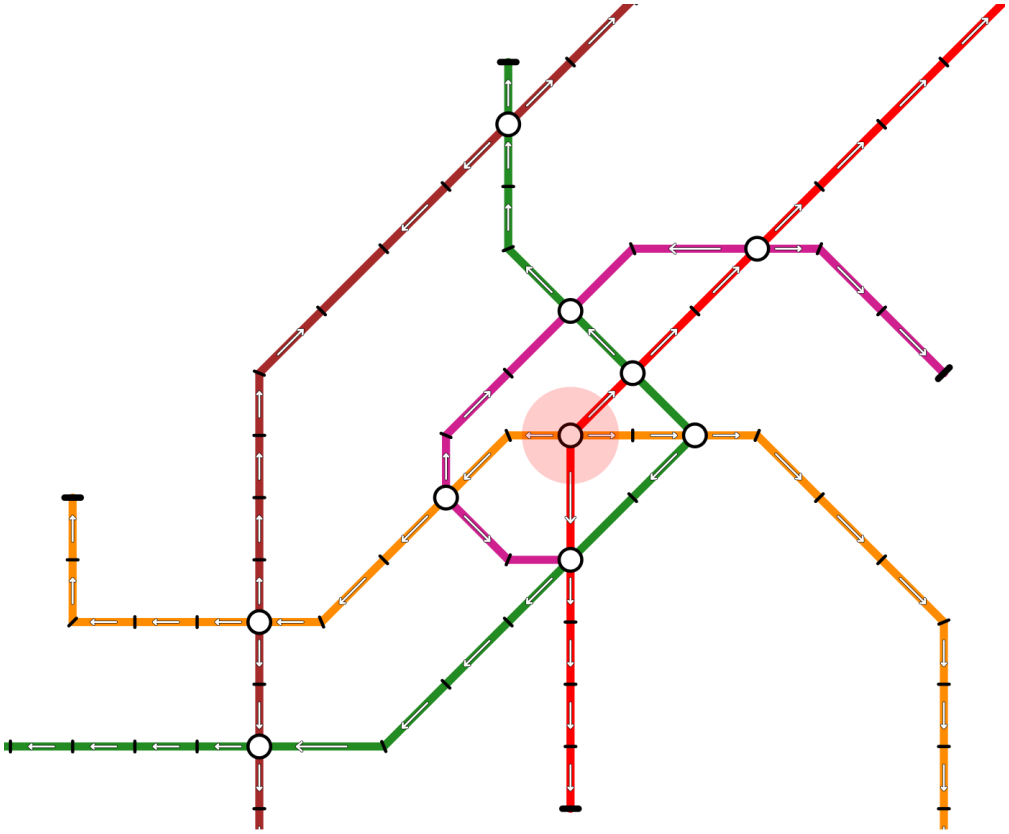


Figure 5.4: Fixed thickness + arrow hints

5.2.4 Varying thickness (all-pairs based)

Figure 5.9 shows a schematic map of Vienna with varying line thicknesses. This type of visualization is all-pairs based, which means that we do not have a predetermined starting station. From our set S of shortest paths we can compute the number of times each edge is part of a shortest path. We call this number the *line count* of an edge e, denoted by c_e. Now, we want to find a function for the line thickness t that flattens out as c_e increases such that

t stays within certain bounds. Let Δ be the difference between the lowest c_e and the highest c_e in the network, we define the thickness t_e of edge e by the formula

$$t_e = 1 + \arctan(\frac{c_e^2}{\Delta^2}) \cdot 10.$$

If we compute the line thicknesses for every edge separately we end up with a map where edges are drawn as rectangles and are not nicely connected when neighboring edges have different thicknesses. In order to tackle this problem we draw edges as trapezoids, so that we can make neighboring edges fit together. The thickness on one endpoint of the edge can be different than the thickness on the other endpoint. We compute the thickness on an endpoint by interpolating both neighboring edges. Figures 5.5 and 5.6 illustrate the difference. In Figure 5.5 we can see that the widths of edges are equal on both endpoints. In Figure 5.6 we applied our interpolation method, so that lines are no longer jagged, but are visualized as a smooth line.

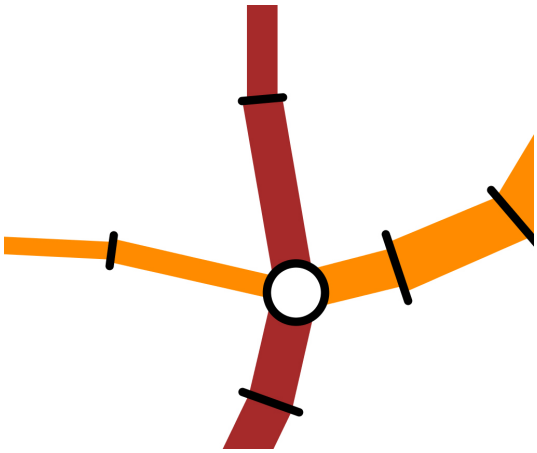


Figure 5.5: Edge thickness computed separately for every edge.

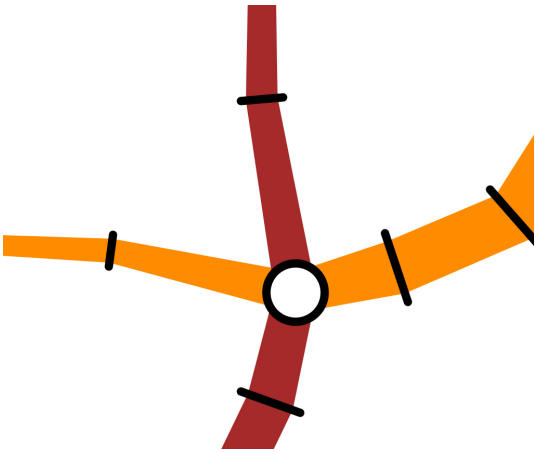


Figure 5.6: Edge thickness interpolated with neighboring edges.

The result of our all-pairs based visualization algorithm is a map in which important lines are displayed thicker than unimportant lines, where the line thicknesses are not relative to a given starting station. An example of a situation in which this visualization can be helpful is seen in Figures 5.7 and 5.8. Imagine that you want to travel from A to B — there are two possible routes, $A \to C \to B$ and $A \to D \to B$. We do not know in which way this schematic map is distorted, so the question arises: which of these routes is the fastest? In the first figure it is not clear which of these routes is faster, however, in the second figure we can infer that $A \to C \to B$ is most likely the fastest route, since the line segments between D and B are thin, and thus rarely used in any fastest route in the network. Note that you should always follow your intuition, for instance if we want to travel from D to B it is faster to go directly to B instead of taking a long detour via A and C.

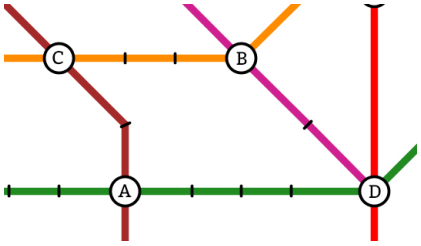


Figure 5.7: When traveling from A to B, it is not clear which route is faster: $A \to C \to B$ or $A \to D \to B$

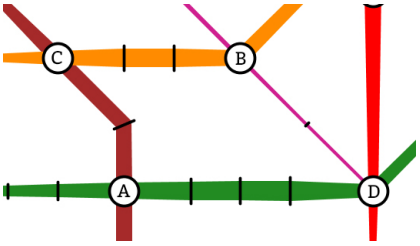


Figure 5.8: All-pairs based visualization. When traveling from A to B, $A \to C \to B$ is most likely the fastest route.

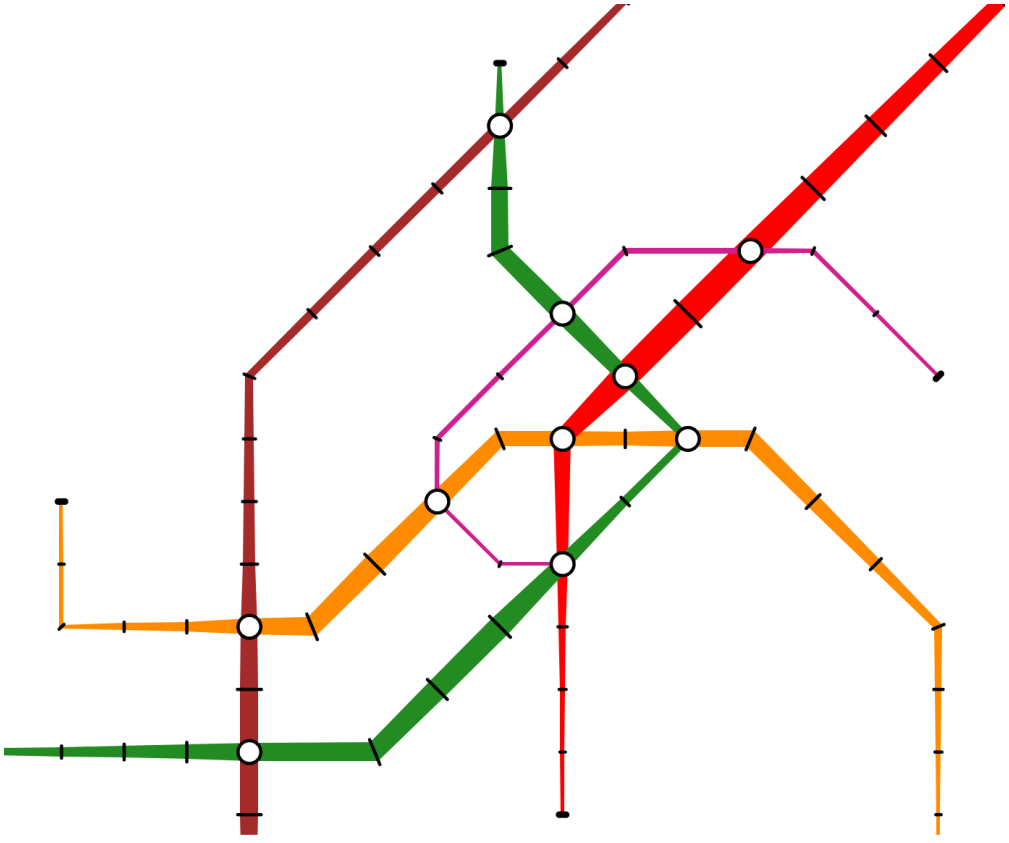


Figure 5.9: Varying thickness (all-pairs based)

5.2.5 Varying thickness (source based) + dotted lines

Figure 5.10 shows a schematic map of Vienna with varying line thicknesses, where some lines are dotted instead of solid. This type of visualization is source based, which means that we require a predetermined starting station A. We stick to the same line counts as computed in Section 5.2.2, where only the shortest paths are considered in which A is the starting station. Similarly, if an edge has a line count of zero we replace the solid line by a dotted line. Similar to the all-pairs based approach we compute the line thickness t_e of edge e a formula that flattens out as c_e increases. For our source based approach we modify the formula slightly — we add an additional constant in the numerator of the fraction to make up for the lower line counts:

$$t_e = 1 + \arctan(\frac{c_1 \cdot c_e^2}{\Delta^2}) \cdot 10.$$

The constant c_1 is experimentally fine-tuned and it turns out that a value of $c_1 = 3$ results in aesthetically pleasing maps where the varying thicknesses are justifiable. This constant is a compromise that works well on the maps we have tested: London, Sydney and Vienna.

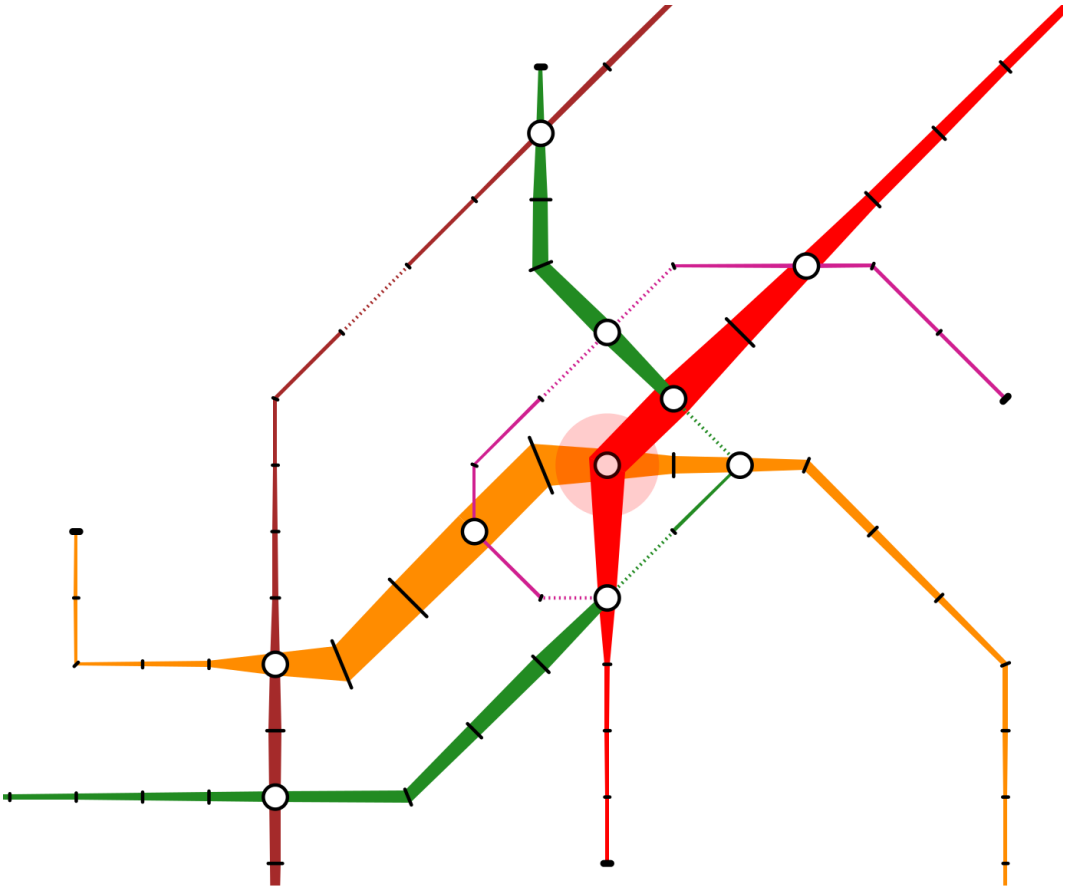


Figure 5.10: Varying thickness (source based) + dotted lines

As described earlier we draw edges as trapezoids and we use an interpolation method to create smooth lines. This introduces a peculiar phenomenon which is clearly visible in the "Varying thickness (source based)" types, and can be seen in Figure 5.11. Mostly around intersections the thickness difference between the endpoints can be large, and as a result some edges appear very unnatural and skewed (see the edge annotated by the dashed red circle). Although peculiar, we believe that this effect is not harmful for route planning, because such a skewed looking edge is always followed by thin edges which do reflect the importance properly. If travelers plan their route from A to B they do not base their decisions on the first edge leaving an intersection, but rather on the thickness of the whole route.

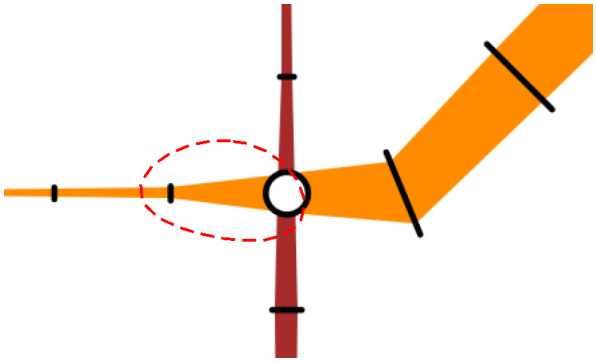


Figure 5.11: Peculiar interpolation effect.

5.2.6 Varying thickness (source based) + arrow hints

Figure 5.12 shows a schematic map of Vienna with varying line thicknesses, where most lines are annotated with arrows. This type of visualization is source based, which means that we require a predetermined starting station A. Similar to the approach in Section 5.2.3 we compute a directed graph which is a tree rooted at A. Here, we use the same formula for the line thickness as described in Section 5.2.5.

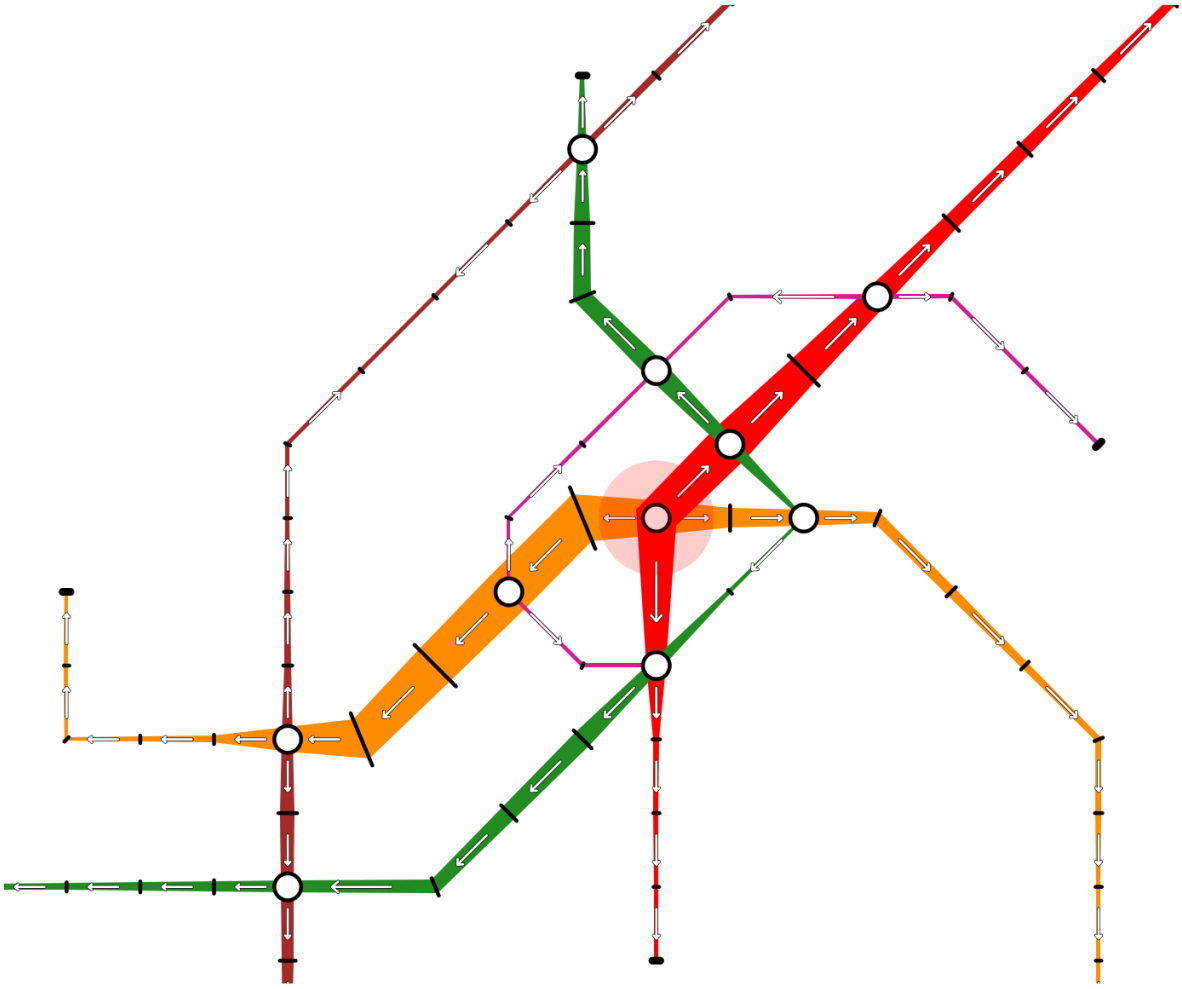


Figure 5.12: Varying thickness (source based) + arrow hints

Chapter 6

User study

In order to investigate the usability of our different visualizations we developed an interactive online questionnaire in which users can perform tasks. A task entails finding the fastest route from A to B on a set of 20 different maps. The efficiency (time in seconds) and correctness of these 20 tasks was measured, and the collected data is analyzed in Section 7.

6.1 Task

Users were given a set of 20 different maps with different visualizations. In these maps, the user's task was to find the fastest route from a given starting station A to a given destination station B. The answer could be given by clicking on all intersections that lie on this route, and finally the destination station had to be clicked to submit the answer. Before each question we let the user know which type of visualization he or she was going to encounter. After 20 questions the questionnaire was complete and the user was asked to fill in additional data, namely: age, level of education, familiarity with metro maps, and general remarks.

6.2 Metrics

ISO 9241-11 [25] describes the following metrics for testing the usability of software in general:

- **Effectiveness.** "Accuracy and completeness with which users achieve specified goals".
- **Efficiency.** "Resources expended in relation to the accuracy and completeness with which users achieve goals".
- **Satisfaction.** "Freedom from discomfort, and positive attitudes towards the use of the product."

According to Ploum [16], these metrics are also useful for assessing the usability of metro maps, because for metro map reading as well as software usage the main usability issue is navigation. Roberts' paper [13] on effective schematic map design shows that there is no direct correlation between attractiveness and usability of metro maps. Since we are solely investigating the usability (planning a route from A to B), we decided to drop the satisfaction metric. In our online questionnaire we measured the following metrics:

- **Efficiency:** the time it takes in seconds for a user to complete the task;
- **Correctness:** a boolean value (0 or 1) indicating whether the user gave the correct answer to the question. Here, the "correct answer" means the actual shortest route as computed by Dijkstra's shortest path algorithm.

6.3 Experimental design

In this experiment we are working with different schematizations and visualizations. These different types are described earlier in Chapters 4 and 5. We have 2 schematization types and 6 visualization types, which combines to 12 different map types.

- **Schematizations:**
  - **Raw:** The original unmodified schematization algorithm by Nöllenburg and Wolff;
  - **Modified:** The algorithm of Nöllenburg and Wolff extended with our own modifications.
- **Visualizations:**
  - **Fixed thickness:** Lines have a fixed width. No extra cues are implemented.
  - **Fixed thickness + dotted lines:** Lines have a fixed width. Edges that are not part of any shortest route are represented with dotted lines. For this type of visualization a given starting station is required.
  - **Fixed thickness + arrow hints:** Lines have a fixed width. Edges are annotated with arrows pointing to the direction that the user must travel. For this type of visualization a given starting station is required.
  - **Varying thickness (all-pairs based):** The thickness of a line depends on how important the line is in the network. The importance is computed by taking all shortest paths between all pairs of stations, while keeping track of how often edges are being used.
  - **Varying thickness (source based) + dotted lines:** The thickness of a line depends on how important the line is in the network. Edges that are not part of

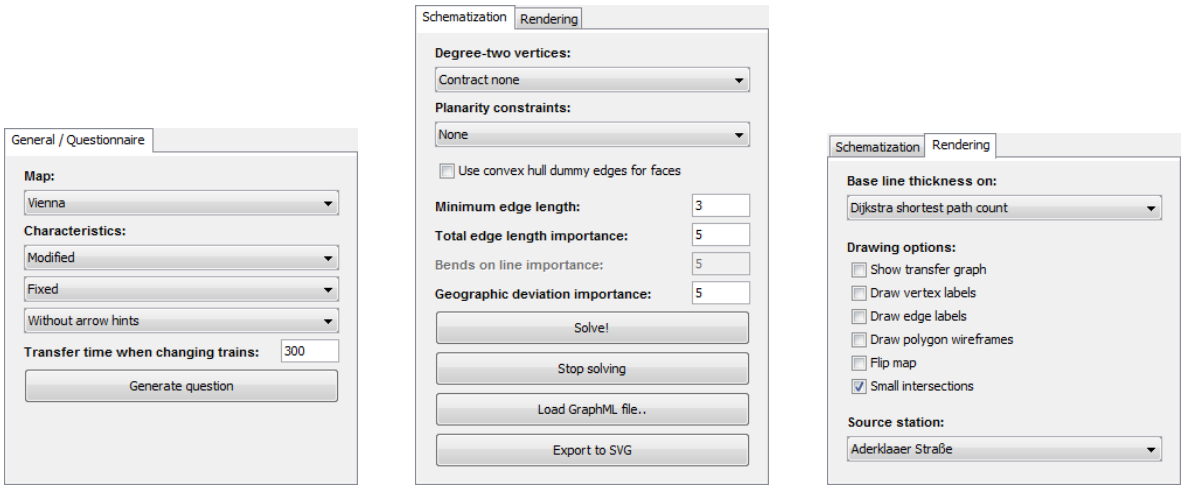
any shortest route are represented with dotted lines. For this type of visualization a given starting station is required.

– **Varying thickness (source based) + arrow hints:** The thickness of a line depends on how important the line is in the network. Edges are annotated with arrows pointing to the direction that the user must travel. For this type of visualization a given starting station is required.

We will now briefly discuss the functionality of our Java application which is used to generate maps and questions. Participants of our questionnaire do not use this application — for our questionnaire we developed a separate web-application as can be read in Section 6.4.

Figure 6.1 (a) shows a panel in which we can select the map (London, Sydney or Vienna), the schematization type, the visualization type and the transfer time when changing trains. The panel in Figure 6.1 (b) allows us to modify some important parameters that are required as input for the schematization algorithm. Figure 6.1 (c) shows a list of visualization options that are primarily used to debug our maps during the development of the schematization and visualization algorithms.

The resulting map is drawn on a `Canvas` object and displayed in our application. Figure 6.2 shows a full screenshot of our application. On the drawing canvas we can select a starting station (marked by the red circle) and a destination station (marked by the green circle) by right-clicking or shift + right-clicking on a station, respectively. When holding down the shift-button, the fastest route from the starting station to the destination station is automatically highlighted.



(a) (b) (c)

Figure 6.1: Panels in our Java application.

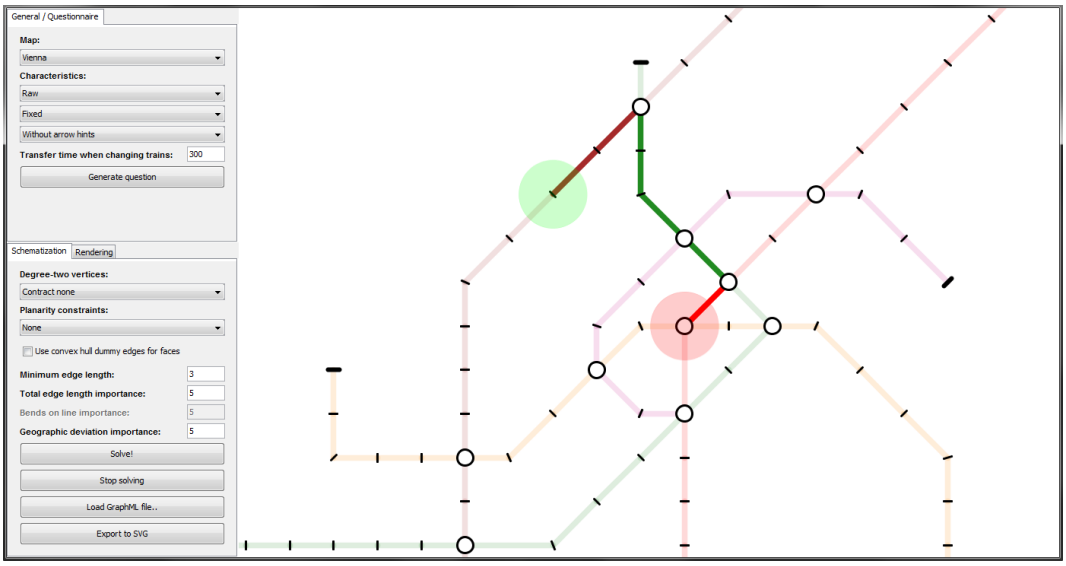


Figure 6.2: Screenshot of our Java application.

Pressing the 'Generate question' button exports a PNG image of the current canvas, combined with an `info.php` file that contains an array with meta data. An example of such a file can be seen below.

```
<?php return array(
      'map' => 'Vienna-raw.graphml',
      'start' => new Point('start', 382, 490),
      'end' => new Point('end', 751, 305),
      'intersections' => array(
            new Point('Praterstern', 566, 182),
            new Point('Spittelau', 320, 59),
            ...
            new Point('Westbahnhof', 74, 551),
      ),
      'answer' => array(
            'Stephansplatz',
            'Schwedenplatz',
            'Praterstern',
      ),
      'schematization' => 'raw',
      'visualization' => 'fixed',
      'arrowhints' => false,
      'averageEdgeTime' => 87,
);
```

This file provides all the map data that our online questionnaire requires. The `Point` objects take a station name, x-coordinate, and y-coordinate as arguments and are used to create clickable areas in our online questionnaire interface.

We will now discuss how we generated the different questions that we presented to the users. We carefully selected 15 pairs of stations (A, B) where A is the starting station and B is the destination station. These pairs are equally distributed among the three maps: London, Sydney and Vienna; this comes down to 5 pairs per map. The criteria for selecting these pairs were as follows:

- The length of the fastest route must be sufficient. Routes that cross at least two intersections are categorized as sufficiently long, given that there are at least 5 to 10 regular stations somewhere on the route too.
- When trying to find the fastest route from A to B there must be at least one crucial decision point where the introduced visual cues can be decisive.

Now, for every pair of stations (15) and every type of schematization (2) and every type of visualization (6) we generate a PNG image and a corresponding `info.php` file. This results in a total of $15 \times 2 \times 6 = 180$ different questions. Appendix B shows a set of 18 examples — 6 different types of visualizations for the maps London, Sydney and Vienna.

6.4 Questionnaire interface

Users that visit the webpage of our questionnaire are first welcomed and introduced to the different visualizations of our maps. They are told that important lines are thick and that they are often part of a fastest route. Furthermore they are told that dotted lines must be avoided and arrows should be followed. These different visualizations are explained using example images. Finally, we say that we incur a penalty of 5 minutes for every train switch. When the users are done reading the material they can click a 'Continue' button to move on to the next page.

On the next page we do a compatibility test where we ask the user to put their browser window in full screen. We provide a dummy image that should fit entirely on the screen. Finally the users must do a compatibility test by selecting all intersection stations on a given map. Recall that an answer to a question must be given by clicking on all intersections that lie on the fastest route, and finally the destination station must be clicked to submit the answer.

Appendix C shows four different figures that describe the process of answering questions.

After 20 questions have been answered the users are redirected to a final page where they can fill in some additional information, namely: age, level of education, familiarity with metro maps, and general remarks.

Chapter 7

Results and discussion

In this project we generated many schematized maps with different parameters. Based on the visual representations of these maps and our intuition we formulated a set of hypotheses (see Section 7.2). To which extent these hypotheses turn out to be correct is discussed in Section 7.4.

7.1 Quality of a map

In this chapter we introduce the *quality of a map*. The quality $Q \in \mathbb{R}$ $(0 \leq Q \leq 1)$ of a map correlates to the average time it takes for a user to find the fastest route from an arbitrary starting station A to an arbitrary destination station B. Maps in which a user finds the fastest route quickly will receive a higher quality value than maps in which it takes a long time before the user finds the fastest route. If a user fails to find the fastest route (i.e., he or she answers the question incorrectly) we assign a quality value of 0. Our goal is to find a function for Q that starts at $y = 1$ and decreases to $y = 0$ as the time t increases. This can be accomplished by the linear function $Q = \max(0, 1 - t/c)$, such that Q intersects the x-axis at value c. If we jump ahead and consider the results in Table 7.1 and 7.2 we notice that the maximum time it took to complete a task in our questionnaire is 71.75 seconds. A good estimate for the constant c would be just above the maximum, e.g., $c = 100$. Now our map is assigned a higher quality when t is small, however, for larger t values we do not want the same quality penalty as for small t values. Our goal is therefore to find a function that decreases fast in the beginning, but flattens out as t approaches $c = 100$. This can be achieved by using a logarithmic function (see Figure 7.1):

$$Q = \max\left(0,\ \ 1 - \log_c(t)\right).$$

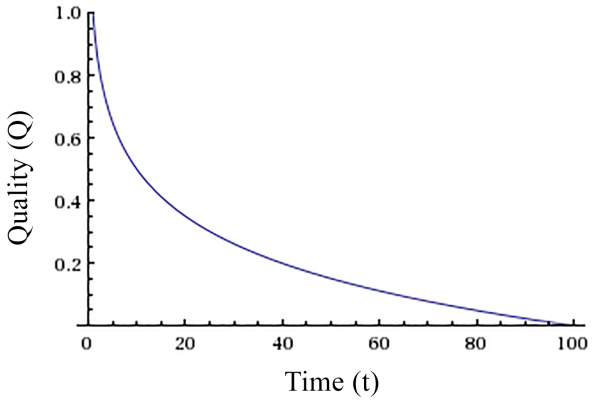


Figure 7.1: Quality function Q for $c = 100$.

Now we must ask ourselves: is the curve in function Q of a correct shape? Perhaps the slope of the function is too steep in the beginning, and as a result the quality drops towards zero too quickly. Or maybe a linear function instead of a downward logarithmic function would be more suitable? We justify the shape of the function Q by looking at the distribution in Figure 7.2. Here we can see that quite a large portion of the questions is answered in under 10 seconds, which is in our opinion a good threshold for classifying an answer as "quick". We want to reflect this in our quality function by assigning substantially higher values for $t \leq 10$. For instance, the difference between $t = 6$ and $t = 7$ should be much more reflected in the function than the difference between $t = 49$ and $t = 50$. We believe that the choice of our curve shape is hereby sufficiently justified.

7.2 Hypotheses

Based on the visual representations of our generated maps and our intuition we formulated the following hypotheses:

H1. Maps that were generated with the modified schematization algorithm are of higher quality than maps generated with the original schematization algorithm of Nöllenburg and Wolff.

H2. Maps that adhere to an all-pairs based visualization are of higher quality than maps with fixed line thicknesses.

H3. Maps with varying thicknesses that adhere to a source based visualization are of higher quality than maps with an all-pairs based visualization.

H4. The introduction of dotted lines increases the quality of a map.

H5. The introduction of arrow hints increases the quality of a map.

In the next sections we will present and analyze our results based on the collected data from our online questionnaire.

7.3 Results

After our online questionnaire had been up and running for two full weeks we pulled the data from the database for analysis. A total of 814 tasks were completed by 46 different people. Unfortunately, some people did not complete all 20 questions — 9 test subjects dropped out before the questionnaire was completed. In order to keep the experiment as unbiased as possible we decided to throw out the partially completed sessions, leaving us with a total of 740 answers provided by 37 different people.

We analyzed our dataset using IBM SPSS Statistics 20 [26]. First, we split our dataset such that the output is organized by map type (we have 12 different map types, as can be seen in Section 6.3). A frequency analysis on the measured times and corresponding qualities results in a set of 12 histograms. From these histograms we can spot several outliers (e.g., a time measurement of 204 seconds when the average lies around 21 seconds). We want to throw out these outliers from the dataset. A general rule of thumb is to throw out all measurements that differ more than 3σ from the mean, where σ denotes the standard deviation. Our dataset of 740 answers is hereby reduced to 725 answers.

Table 7.1 shows statistics for maps that were generated with the original algorithm of Nöllenburg and Wolff. Table 7.2 shows statistics for maps that were generated with our modified schematization algorithm. These tables contain the following information:

- **Time in seconds:**
  - **Min.**: the minimum time it took to complete a task in our questionnaire;
  - **Max.**: the maximum time it took to complete a task in our questionnaire;
  - **Std.dev.**: the standard deviation of the distribution;
  - **Skew**: the skewness of the distribution. A positive skew indicates that the tail on the right side is longer than the left side;
  - **Mean**: the average time it took to complete a task in our questionnaire;
- **Correctness mean:** the percentage of tasks that were correctly answered in our questionnaire, represented as a number between 0 and 1;
- **Quality mean:** the average quality of the map type;

Schematization: Raw

	Time in seconds					Correctness	Quality
Visualization	**Min.**	**Max.**	**Std.dev.**	**Skew**	**Mean**	**Mean**	**Mean**
Fixed thickness	5.11	52.85	11.24	1.22	18.10	0.41	0.172
Fixed thickness + dotted lines	5.18	71.75	13.39	1.99	17.45	**0.57**	**0.243**
Fixed thickness + arrow hints	5.11	59.99	13.42	1.49	19.51	0.41	0.167
Varying thickness (all-pairs based)	**4.06**	66.50	14.76	**2.26**	17.72	0.53	0.238
Varying thickness (source based) + dotted lines	4.41	46.34	**10.09**	1.42	**16.19**	0.56	0.230
Varying thickness (source based) + arrow hints	5.39	**40.60**	10.11	1.17	16.33	0.48	0.216

Table 7.1: Results for 6 different types of visualizations, using Nöllenburg and Wolff's original schematization algorithm.

Schematization: Modified

	Time in seconds					Correctness	Quality
Visualization	**Min.**	**Max.**	**Std.dev.**	**Skew**	**Mean**	**Mean**	**Mean**
Fixed thickness	4.27	67.76	15.38	1.09	21.23	0.44	0.172
Fixed thickness + dotted lines	5.81	48.16	9.48	1.46	16.47	**0.71**	0.305
Fixed thickness + arrow hints	**2.87**	44.45	9.01	1.38	13.80	0.62	0.290
Varying thickness (all-pairs based)	4.48	46.27	9.68	1.17	16.11	0.48	0.209
Varying thickness (source based) + dotted lines	3.36	50.47	10.45	**1.70**	15.33	0.61	0.279
Varying thickness (source based) + arrow hints	3.78	**40.32**	**8.20**	1.18	**13.64**	0.67	**0.324**

Table 7.2: Results for 6 different types of visualizations, using our modified schematization algorithm.

A histogram of the complete dataset shows that we are dealing with a *right-tailed* distribution (positive skewness), as can be seen in Figure 7.2. When we split this histogram into 12 smaller histograms — one for every map type — we notice the same (right-tailed) distributions.

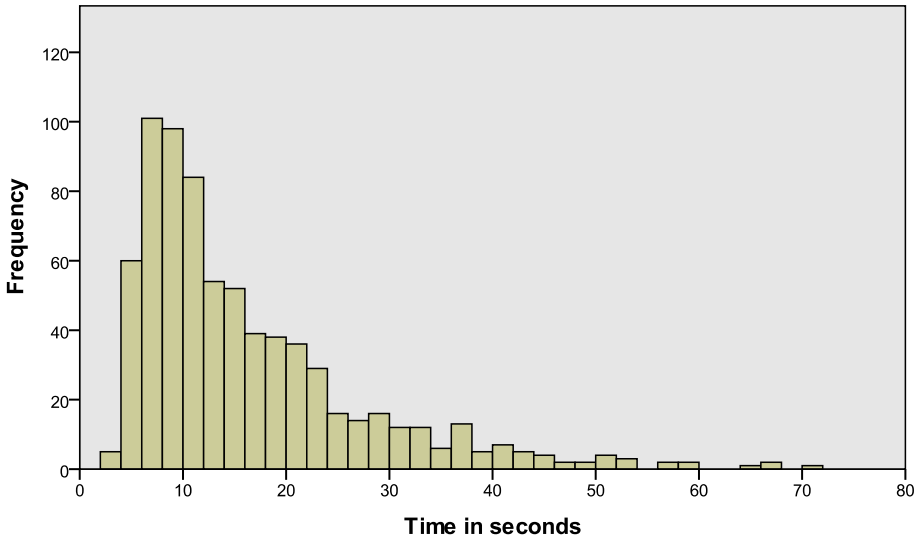


Figure 7.2: Right-tailed distribution of measured times.

Table 7.3 shows mean correctnesses and mean timings for every map type. For every column the highest mean correctness and the lowest mean timings are displayed in boldface. These results are discussed in Section 7.4 and onwards.

	Raw schematization			**Modified schematization**		
	London	**Sydney**	**Vienna**	**London**	**Sydney**	**Vienna**
Fixed thickness	MC: 0.44 MT: 13.81	MC: 0.62 MT: 21.90	MC: 0.27 MT: 20.43	MC: 0.43 MT: 18.62	MC: 0.42 MT: 27.32	MC: 0.44 MT: 19.35
Fixed thickness + dotted lines	MC: 0.73 MT: 23.66	MC: **0.84** MT: 20.34	MC: 0.33 MT: 17.54	MC: 0.71 MT: 21.87	MC: 0.65 MT: **13.55**	MC: **0.73** MT: 16.70
Fixed thickness + arrow hints	MC: 0.67 MT: **11.67**	MC: 0.67 MT: 22.21	MC: 0.33 MT: 21.42	MC: **0.72** MT: **14.32**	MC: 0.22 MT: 14.26	MC: 0.60 MT: 15.12
Varying thickness (all-pairs based)	MC: 0.45 MT: 31.28	MC: 0.78 MT: **13.22**	MC: 0.43 MT: 20.50	MC: 0.50 MT: 16.27	MC: 0.47 MT: 16.30	MC: 0.50 MT: 17.90
Varying thickness (source based) + dotted lines	MC: 0.63 MT: 17.78	MC: 0.43 MT: 18.51	MC: **0.58** MT: **15.07**	MC: 0.63 MT: 18.52	MC: 0.61 MT: 15.33	MC: 0.53 MT: **14.11**
Varying thickness (source based) + arrow hints	MC: **0.75** MT: 14.74	MC: 0.50 MT: 15.12	MC: 0.54 MT: 18.47	MC: 0.69 MT: 14.46	MC: **0.70** MT: 13.82	MC: 0.65 MT: 14.12

Table 7.3: Mean correctnesses (MC) and mean timings (MT) in seconds for each map.

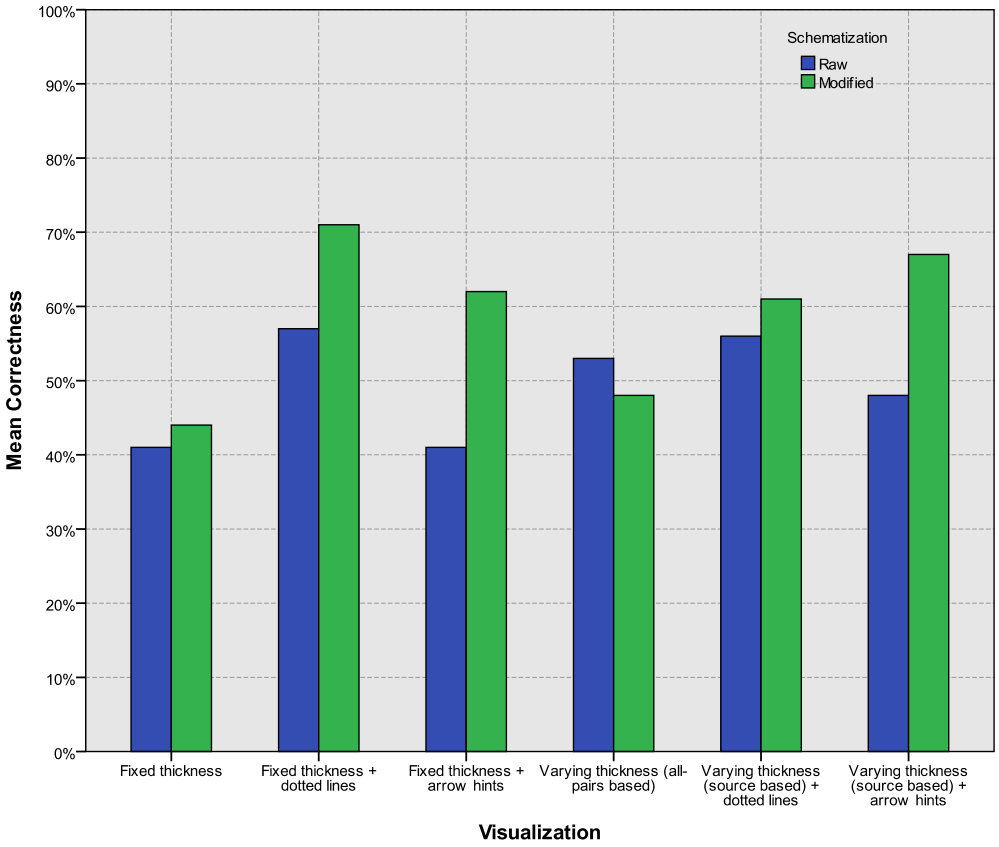


Figure 7.3: Bar chart clustered by schematization type. The horizontal axis denotes the type of visualization and the vertical axis denotes the average correctness in percentages.

7.4 Discussion

In this section we make several observations and derive conclusions based on the collected data. The bar charts and box plots we present here give an overview of the correctness, time measurements and a combination of these two factors: the quality.

7.4.1 Correctness

Figure 7.3 shows a bar chart where the horizontal axis denotes the type of visualization and the vertical axis denotes the average correctness in percentages. For instance, a correctness of 71% in the category "Fixed thickness + dotted lines (Modified)" means that out of all people that answered a question in this category 71% gave the correct answer by selecting the actual fastest route on the map and 29% of the people selected a different (longer) route.

Prior to drawing any conclusions based on the data we must first ask ourselves: are the differences among the groups statistically significant? We answer this question by computing

the probability of error (p-value) using an independent-samples t-test in SPSS. The p-value represents the probability that the observation can be attributed to chance. A smaller p-value thus indicates that the observation is less likely a coincidence. Significance levels can be found in Table 7.4. In this table we compare every map type to the "Fixed thickness" type of their corresponding schematization, because we are interested in measuring whether our visualizations are a significant improvement over the original (Fixed thickness) visualizations. The difference between two groups is judged to be statistically significant when $p = 0.05$ or less — we marked these values by making them bold in the table.

Schematization	**Visualization**	**p-value compared to "Fixed thickness"**
Raw	Fixed thickness + dotted lines	0.081
Raw	Fixed thickness + arrow hints	0.996
Raw	Varying thickness (all-pairs based)	0.203
Raw	Varying thickness (source based) + dotted lines	0.131
Raw	Varying thickness (source based) + arrow hints	0.504
Modified	Fixed thickness + dotted lines	**0.005**
Modified	Fixed thickness + arrow hints	**0.024**
Modified	Varying thickness (all-pairs based)	0.632
Modified	Varying thickness (source based) + dotted lines	0.073
Modified	Varying thickness (source based) + arrow hints	**0.003**

Table 7.4: Significance levels of correctness differences between the "Fixed thickness" group and other map types. A smaller p-value indicates that the observation is less likely a coincidence.

If we ignore the time measurements and focus only on correctness (i.e., we only consider whether the answer is correct or not) we observe that users perform the worst (42% when we combine the schematizations) at maps without any visual aids (Fixed thickness). An all-pairs based visualization boosts the performance slightly to 51%, but from Table 7.4 we can derive that this is statistically insignificant and thus likely to be a coincidence. We can conclude that the map type "Fixed thickness + dotted lines" wins with high statistical significance with an average correctness of 62%.

Furthermore, we observe that in all map types (except one) the modified schematization algorithm beats the raw schematization algorithm. Table 7.4 shows that the p-values are substantially lower in maps with modified schematization, compared to their raw schematization counterparts. This means that it is very unlikely that these differences are a coincidence. We believe that this is caused by the fact that our modified schematization algorithm provides a better overview of the network by allocating more space around intersections and important lines. Additionally, important lines are drawn with a minimal amount of line bends at the cost of heavy geographical distortions. These heavy distortions could be confusing to users in practice, as human beings tend to compare schematic maps to their own mental map of a

city. However, for the sole purpose of route planning these heavy modifications turn out to be beneficial for the correctness results.

One might think that the amount of incorrect answers is quite high — roughly half of all the given answers was incorrect. We are not aware of any studies that lets users select a fastest route in an interactive setting like ours, so we do not have any references to compare our results to. Most metro map usability studies involve searching or counting tasks, that may not properly reflect the true purpose (according to Beck [1]) of metro maps: finding the fastest route from A to B. A comparable study done by Roberts et al. [18] measures the estimated journey duration, but they do not judge an answer as strictly as we do when we say that an answer is completely wrong even if the alternative route was only slightly slower than the fastest route. Given the fact that our questions are generally more difficult than the tasks that are conducted in earlier research it is very realistic that roughly half of the questions are answered incorrectly.

7.4.2 Time measurements

Figure 7.5 shows a box plot where the horizontal axis denotes the type of visualization and the vertical axis denotes the time in seconds it took to complete the task. Figure 7.4 explains how a box plot should be interpreted. A box and its whiskers can be divided into four quartiles, each of which account for 25% of the data points. The bottom and top edges of the box indicate the *interquartile range* (IQR), and the whiskers that extend from this box indicate the range of values that are outside of the IQR. Values that are higher than Q3 + 1.5*IQR are considered to be *mild outliers*, denoted by a small circle. Values that are higher than Q3 + 3*IQR are considered to be *extreme outliers*, denoted by a small asterisk.

From Figure 7.5 we can observe that a large portion of the users completes a task somewhere between 10 and 20 seconds. What stands out is that the first green box (Fixed thickness, modified) is substantially larger than the rest, in other words, people tend to require more time for this combination of visualization and schematization. Furthermore, we observe that time measurements on maps with arrow hints and a modified schematization are relatively low. We believe people pick up routes easier and more quickly when they are led by the arrow hints. One participant also mentioned this in the questionnaire feedback. She stated that she did not need to scan through the network prior to planning a route, but instead started clicking intersections right away because the arrows are leading you in the right direction. From our own observations we saw that raw schematization maps with arrow hints are quite chaotic and are detrimental to the readability of the map. This could explain the high time measurements on the map type “Fixed thickness + arrow hints, raw”.

There is no schematization which results in consistently better times for all visualizations. In

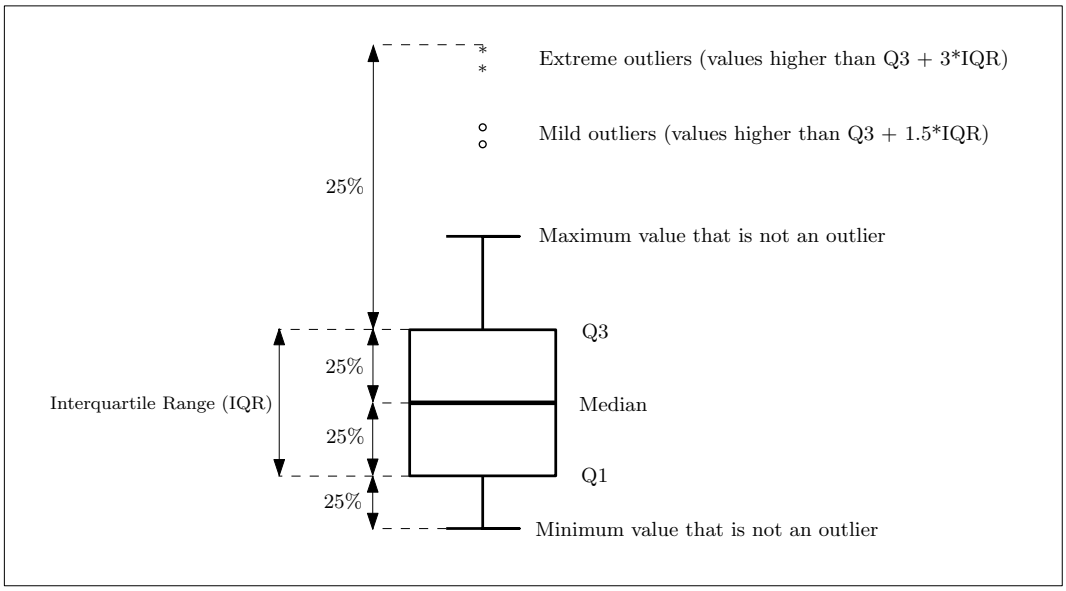


Figure 7.4: How to interpret a box plot.

3 out of 6 visualizations the raw schematization beats the modified schematization, and in the other 3 visualizations we see the exact opposite. Also, in most cases the difference is so small that we can not label either of these schematizations 'better' than the other.

7.4.3 Quality

Until now we have investigated correctness and time measurements separately, hence we do not know the correlation between these two variables. Our quality function

$$Q = \gamma \cdot \max\left(0,\ \ 1 - \log_{100}(t)\right),$$

$$\gamma = \begin{cases} 1 & \text{if the given answer is the correct answer} \\ 0 & \text{otherwise} \end{cases},$$

as described in the introduction of this section is a measure that combines the correctness and time into one variable. These quality values can conclusively be used to measure the overall usability of a map.

Figure 7.6 shows a bar chart where the horizontal axis denotes the type of visualization and the vertical axis denotes the average quality. We have seen that time measurements do not differ very much among different visualizations, hence the time factor impact in our quality computations is approximately the same for all visualizations. As a result, the bar chart in Figure 7.6 is almost identical to the correctness bar chart in Figure 7.3. We can make the same observations as earlier, and derive the same conclusions based on these observations.

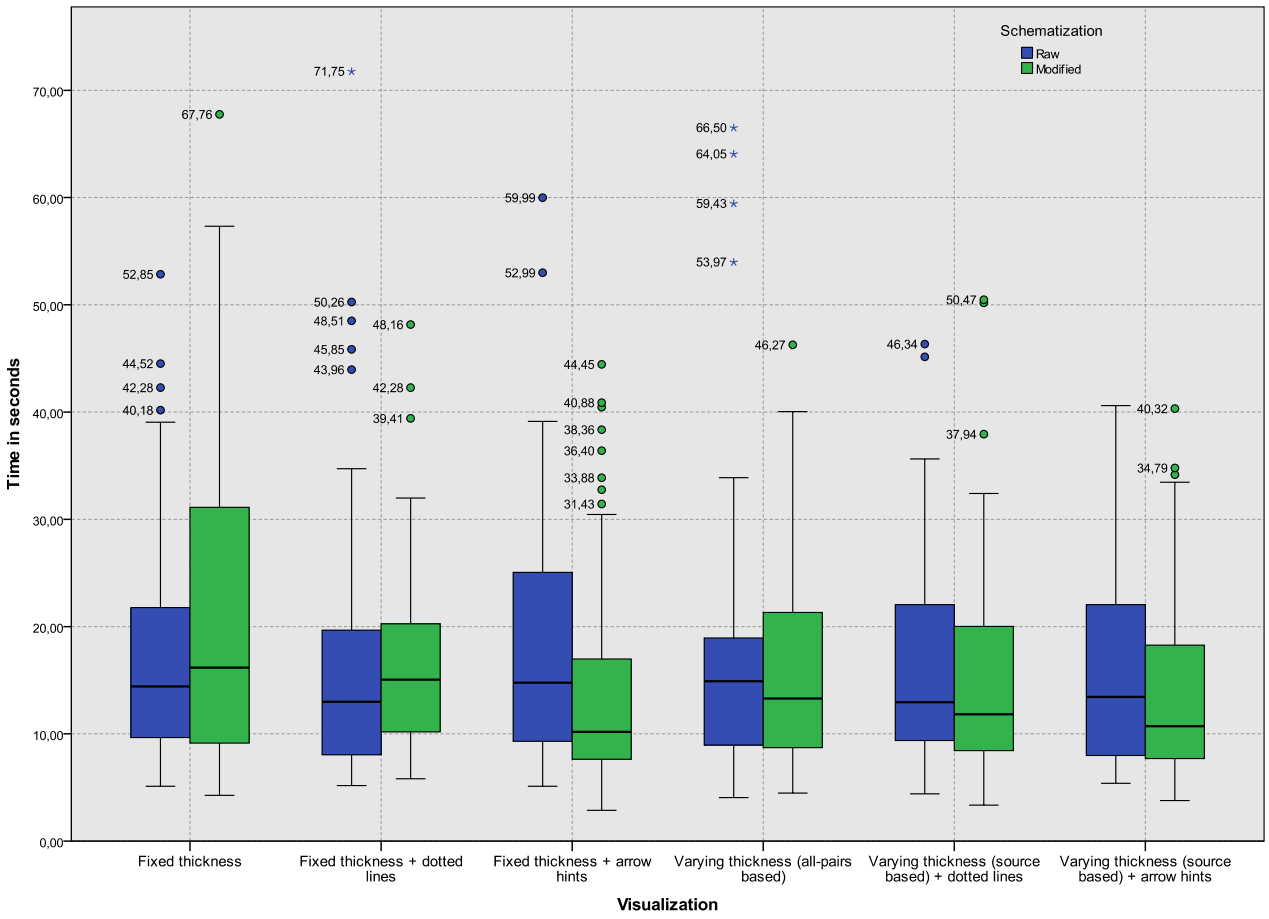


Figure 7.5: Box plot clustered by schematization type. The horizontal axis denotes the type of visualization and the vertical axis denotes the time in seconds it took to complete the task.

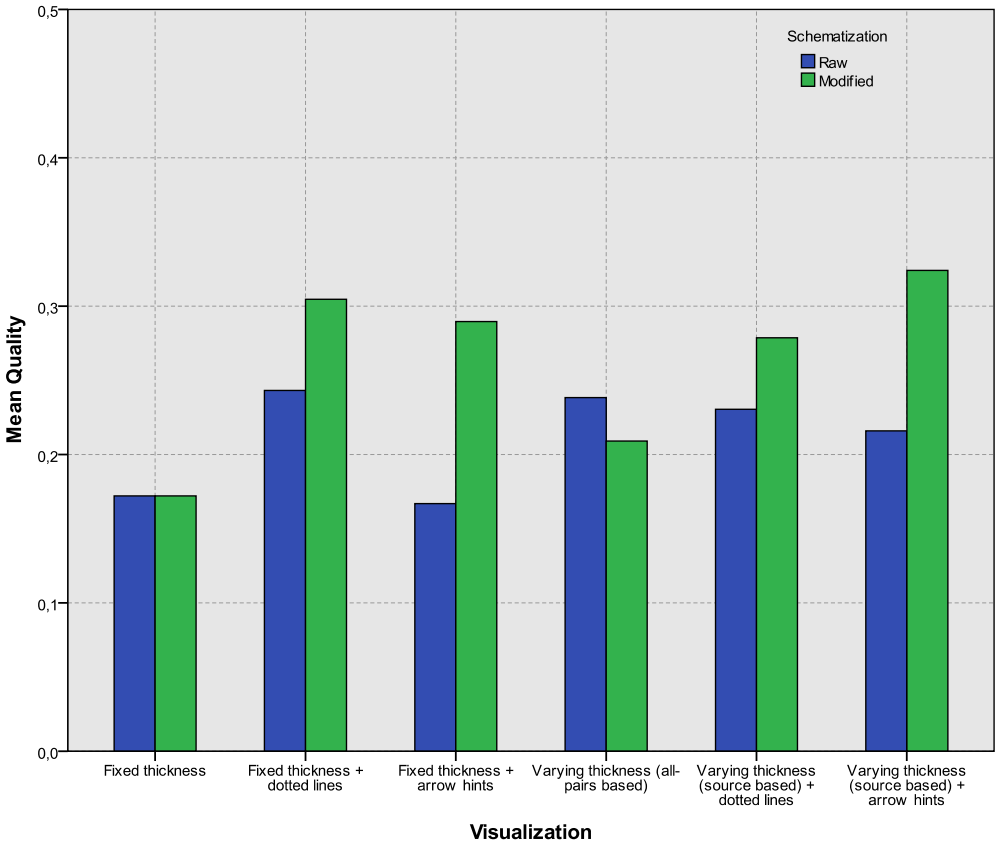


Figure 7.6: Bar chart clustered by schematization type. The horizontal axis denotes the type of visualization and the vertical axis denotes the average quality.

7.4.4 Hypotheses tests

In this section we discuss whether the hypotheses H1 through H5 can be accepted or rejected. An observation is said to be statistically significant if the p-value is lower than or equal to 0.05 — we marked these values by making them bold in the tables.

H1. Maps that were generated with the modified schematization algorithm are of higher quality than maps generated with the original schematization algorithm of Nöllenburg and Wolff.

If we want to answer this question we must first verify that the differences between the raw and modified schematization groups are statistically significant. Table 7.5 shows the mean differences and p-values for quality comparisons between raw and modified schematizations. A positive mean difference indicates that the modified schematization is of higher quality than the raw schematization counterpart.

In the two cases with arrow hints we observe a p-value ≤ 0.05, which indicates that modified schematization is of higher quality than raw schematization when we use arrow hints. Hypothesis H1 states that modified schematization is better than raw schematization in all situations, but unfortunately the numbers in Table 7.5 do not provide enough evidence to accept this hypothesis, because only 2 out of 6 p-values fall within the threshold of $p \leq 0.05$ and the other values are too far off. Hence, hypothesis H1 can not be validated nor rejected.

Visualization	**Mean difference**	**p-value**
Fixed thickness	0.000	1.000
Fixed thickness + dotted lines	0.061	0.170
Fixed thickness + arrow hints	0.123	**0.011**
Varying thickness (all-pairs based)	-0.029	0.541
Varying thickness (source based) + dotted lines	0.048	0.329
Varying thickness (source based) + arrow hints	0.108	**0.037**

Table 7.5: Quality differences between the raw and modified schematization groups. A positive mean difference indicates that the modified schematization is of higher quality than the raw schematization counterpart.

H2. Maps that adhere to an all-pairs based visualization are of higher quality than maps with fixed line thicknesses.

Table 7.6 shows the mean differences and p-values for quality comparisons between the "Fixed thickness" and "Varying thickness (all-pairs based)" groups. A positive mean difference in-

dicates that the all-pairs based visualization is of higher quality than the fixed thickness visualization. It turns out that in both schematizations the all-pairs based visualization is of higher quality, however, the p-values are much higher than 0.05, which means that the differences in quality are not statistically significant at all. Therefore, hypothesis H2 can not be validated nor rejected.

Schematization	**Mean difference**	p**-value**
Raw	0.066	0.153
Modified	0.037	0.393

Table 7.6: Quality differences between the "Fixed thickness" and "Varying thickness (all-pairs based)" groups. A positive mean difference indicates that the all-pairs based visualization is of higher quality than the fixed thickness visualization.

H3. Maps with varying thicknesses that adhere to a source based visualization are of higher quality than maps with an all-pairs based visualization.

Table 7.7 shows the mean differences and p-values for quality comparisons with the "Varying thickness (all-pairs based)" group. A positive mean difference indicates that the all-pairs based visualization is of higher quality than the fixed thickness visualization. Here, we observe two positive mean differences with extreme statistical insignificance and two negative mean differences which are significantly more accurate. Unfortunately these numbers do not provide enough evidence to accept the hypothesis, because only one of the p-values falls within the threshold of $p \leq 0.05$ and the other values are too far off. Hence, hypothesis H3 can not be validated nor rejected.

Schematization	**Visualization**	**Mean difference**	p**-value**
Raw	Varying thickness (source based) + dotted lines	0.008	0.877
Raw	Varying thickness (source based) + arrow hints	0.022	0.703
Modified	Varying thickness (source based) + dotted lines	-0.070	0.132
Modified	Varying thickness (source based) + arrow hints	-0.115	**0.004**

Table 7.7: Quality differences compared to the "Varying thickness (all-pairs based)" visualization. A positive mean difference indicates that the all-pairs based visualization of the corresponding schematization type is of higher quality.

H4. The introduction of dotted lines increases the quality of a map.

Table 7.8 shows the mean differences and p-values for quality comparisons with the "Fixed thickness" visualizations of their corresponding schematization types. A positive mean difference indicates that the fixed thickness visualization of the corresponding schematization type is of higher quality. Here it can be seen that both mean differences are negative: -0.071 and -0.132, with p-values 0.100 and 0.003 respectively. A p-value of 0.100 is not within the threshold of 0.05, which means that — although there is a weak indication that H4 holds — we can not accept nor reject this hypothesis.

Schematization	**Visualization**	**Mean difference**	**p-value**
Raw	Fixed thickness + dotted lines	-0.071	0.100
Modified	Fixed thickness + dotted lines	-0.132	**0.003**

Table 7.8: Quality differences compared to the "Fixed thickness" visualization. A positive mean difference indicates that the fixed thickness visualization of the corresponding schematization type is of higher quality.

H5. The introduction of arrow hints increases the quality of a map.

Table 7.9 shows the mean differences and p-values for quality comparisons with the "Fixed thickness" visualizations of their corresponding schematization types. A positive mean difference indicates that the fixed thickness visualization of the corresponding schematization type is of higher quality. Here it can be seen that we have one very small mean difference which is extremely inaccurate (high p-value). The negative mean difference (-0.117) indicates with a high level of significance that the introduction of arrow hints does indeed increase the quality of a map. Unfortunately these numbers do not provide enough evidence to accept the hypothesis, because only one of the p-values falls within the threshold of $p \leq 0.05$ and the other value is too far off. Hence, hypothesis H5 can not be validated nor rejected.

Schematization	**Visualization**	**Mean difference**	**p-value**
Raw	Fixed thickness + arrow hints	0.005	0.914
Modified	Fixed thickness + arrow hints	-0.117	**0.003**

Table 7.9: Quality differences compared to the "Fixed thickness" visualization. A positive mean difference indicates that the fixed thickness visualization of the corresponding schematization type is of higher quality.

Chapter 8

Conclusions

Although most of our results turned out to be not statistically significant enough, we do have indicators that some of our schematizations and visualizations allows travelers for better route planning.

Among the source based visualizations there are indications that on maps generated with our modified schematization algorithm travelers can plan their route better than on maps generated with the raw schematization algorithm. This is reflected in the analysis of hypothesis H1 in Section 7.4.4. Unfortunately the statistical significance of our results was inadequate. Perhaps a larger sample size can provide more statistically significant results on this matter.

All-pairs based visualizations in general failed to result in qualitatively better maps, at least as far as the statistics have taught us. All-pairs based visualizations provide less relevant information for the starting station than source based visualizations, but it may still be better than not adding any visual travel time cues at all. In the results there are some indications that all-pairs based visualizations are better than not adding any visual travel time cues at all, but these results all turned out to be insignificant. Therefore we can not draw any evident conclusions from our analyses.

Furthermore, we proved that the introduction of dotted lines allows travelers for better route planning compared to maps without any extra visual cues. This is also in line with our expectations; when a user must choose between two or more possible fastest routes, in a lot of cases the dotted lines cancel out some of these options (since traveling over a dotted line is never the fastest route). As a result, people find the actual fastest route more often. Some people that participated in the questionnaire suggested to omit the dotted lines completely, as they should never be used anyways. We believe that this heavy modification would be too confusing since the map would look incomplete and too different for every starting station.

There are also some indications that the introduction of arrow hints allows travelers for better

route planning compared to maps without any extra visual cues. From the users' feedback in the questionnaire we learned that some people find themselves guided by the arrows early on, while other people thought the arrows did not help at all. This may be related to the experience that people have with reading metro maps. A participant that stated that the arrows were no help at all also filled in that he is very familiar with metro maps. Two participants that were guided by the arrows said that they have no experience with route planning on metro maps.

8.1 Future work

Transfer times

In our approach we incur a penalty of 300 seconds when changing trains. This is an estimate based on the intervals with which trains arrive and leave on most metro lines. A nice improvement to our maps would be to let the transfer time depend on the time it would actually take to transfer from one line to another. As a result, our maps would reflect that it is sometimes better to avoid changing trains on large stations because on these stations it generally takes longer to transfer from one train to another due to longer walks. It is also realistic to believe that schedules of individual trains are geared to one another to make transfer times as short as possible. Note that this does not apply to metro networks of larger cities, because when trains arrive and leave at very short intervals (say 10 times per hour) it is nearly impossible to adjust them to one another. If all this data were available it would be little effort to implement this in our model. This would only require some modifications in the part where we transform the metro map to a transfer graph (see Section 3.3). Instead of taking a fixed weight W for every edge in the complete subgraphs K_s we would then compute the weights based on the actual data.

Overlaps in parallel lines

Parallel line segments contain two or more individual line segments that are drawn adjacent to each other. When drawing parallel lines the ordering of the individual lines is a complex problem. The ordering must be consistent on a path of degree-2 vertices and when the paths split up the number of overlaps of the individual lines must be minimized. The algorithm of Nöllenburg and Wolff provides an algorithm for consistency in paths of degree-2 vertices, but does not guarantee that the number of overlaps of individual lines is minimized. Figure 5.1 shows an example where it would give a nicer result if the green and yellow lines were swapped in the intersection. This problem can be quite complex and it may not always be possible to

reduce the amount of overlaps to zero. It would be interesting to extend our visualizations with an automated method for minimizing the number of line overlaps.

Sample size

Most of our results turned out to be statistically insignificant. Hence, our sample size of 37 may not be sufficient for this type of experiment. It would be interesting to see whether a larger sample size would increase the significance of our results and possibly have an effect on our drawn conclusions.

Difficulty measure of individual questions

It could happen that test subjects must answer a question on, for instance, a "Fixed thickness + dotted lines" map where the dotted lines only eliminate the obviously wrong answers, or they eliminate nothing at all because the dotted lines are placed on parts of the map that are irrelevant for the completion of the task. In further research it would be interesting to have some kind of measurement that indicates how much the modified schematization and introduced visual cues have an effect on the difficulty of the individual questions. More factors that could determine the complexity of a question could be: the length of the fastest route, the number of stations on the fastest route, the number of intersections on the fastest route, the number of train switches, and possibly the length of a faster detour in relation to the path with the shortest distance. Involving more meta data may give more insight in the difference between 'easy questions' and 'hard questions', and how this affects the users' performance.

Bibliography

[1] Ken Garland. *Mr Beck's Underground Map.* Capital Transport, 1994.

[2] Maxwell J. Roberts. *Underground maps after Beck.* Capital Transport, 2005.

[3] Maxwell J. Roberts. *Underground maps unravelled: Explorations in information design.* Self-published, 2012.

[4] Daniel Chivers and Peter Rodgers. Octilinear force-directed layout with mental map preservation for schematic diagrams. In *Diagrammatic Representation and Inference*, pages 1–8. Springer, 2014.

[5] Martin Nöllenburg and Alexander Wolff. Drawing and labeling high-quality metro maps by mixed-integer programming. *Visualization and Computer Graphics, IEEE Trans.*, 17(5):626–641, 2011.

[6] Jonathan Stott, Peter Rodgers, Juan Carlos Martinez-Ovando, and Stephen G. Walker. Automatic metro map layout using multicriteria optimization. *Visualization and Computer Graphics, IEEE Trans.*, 17(1):101–114, 2011.

[7] IBM ILOG CPLEX, http://www.ilog.com/products/cplex, 2015.

[8] Seok-Hee Hong, Damian Merrick, and Hugo A.D. do Nascimento. The metro map layout problem. In *Graph Drawing*, pages 482–491. Springer, 2005.

[9] Arne Frick, Andreas Ludwig, and Heiko Mehldau. A fast adaptive layout algorithm for undirected graphs. In *Graph Drawing*, pages 388–403. Springer, 1995.

[10] Franois Bertault. A force-directed algorithm that preserves edge crossing properties. In *Graph Drawing*, pages 351–358. Springer, 1999.

[11] Kozo Sugiyama and Kazuo Misue. Graph drawing by the magnetic spring model. *Journal of Visual Languages and Computing*, 6(3):217–231, 1995.

[12] Peter Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[13] Maxwell J. Roberts. *What's your theory of effective schematic map design?* University of Essex, 2014.

[14] Herman Haverkort. Embedding cues about travel time in schematic maps. In *Schematic Mapping Workshop*, 2014.

[15] Joshua Stevens. Your favorite traffic map is lying to you. blog post, 2013.

[16] Eva Ploum. *Metro Map Usability: an Experiment on the Usability of Automatically Generated Metro Maps.* Eindhoven University of Technology, 2009.

[17] Zhan Guo. Mind the map! the impact of transit maps on path choice in public transit. *Transportation Research Part A: Policy and Practice*, 45(7):625–639, 2011.

[18] Maxwell J. Roberts, Elizabeth J. Newton, Fabio D. Lagattolla, Simon Hughes, and Megan C. Hasler. Objective versus subjective measures of paris metro map usability: Investigating traditional octolinear versus all-curves schematics. *International Journal of Human-Computer Studies*, 71(3):363–386, 2013.

[19] Transport for London API, https://api-portal.tfl.gov.uk/docs, 2015.

[20] Daniel Chivers. Graphs - Force-Directed Octilinear Layout, http://www.cs.kent.ac.uk/projects/fdol, 2015.

[21] Timetables - Sydney Trains, http://www.sydneytrains.info/timetables/, 2015.

[22] Horst Prillinger. Vienna Metro: Timetables, http://homepage.univie.ac.at/horst.prillinger/ubahn/english/timetables.html, 2015.

[23] Ulrik Brandes, Markus Eiglsperger, Jürgen Lerner, and Christian Pich. *Graph markup language (GraphML).* Bibliothek der Universität Konstanz, 2010.

[24] Data Structures and Algorithms: Dijkstra's Algorithm, https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html, 2015.

[25] WD ISO. 9241-11. ergonomic requirements for office work with visual display terminals (vdts). *The international organization for standardization*, 1998.

[26] IBM SPSS Statistics, http://www-03.ibm.com/software/products/en/spss-stats-standard, 2015.

Appendix A

Schematizations

In this chapter we present the different schematizations for London, Sydney and Vienna.

List of Figures

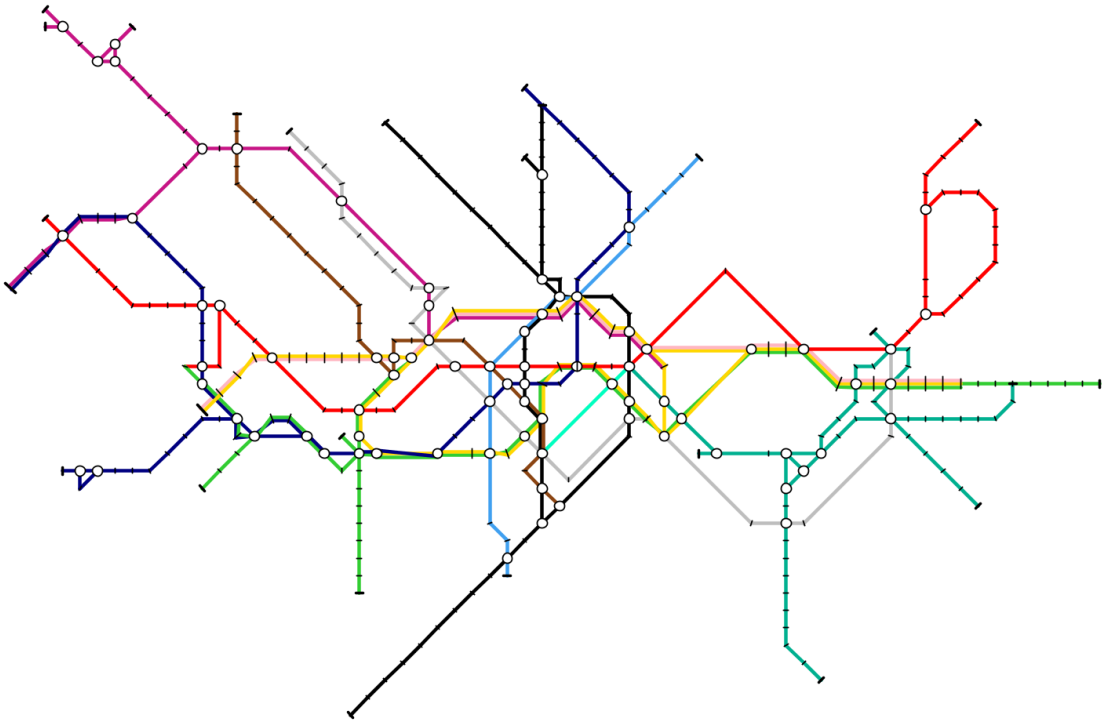


Figure A.1: London, raw schematization.

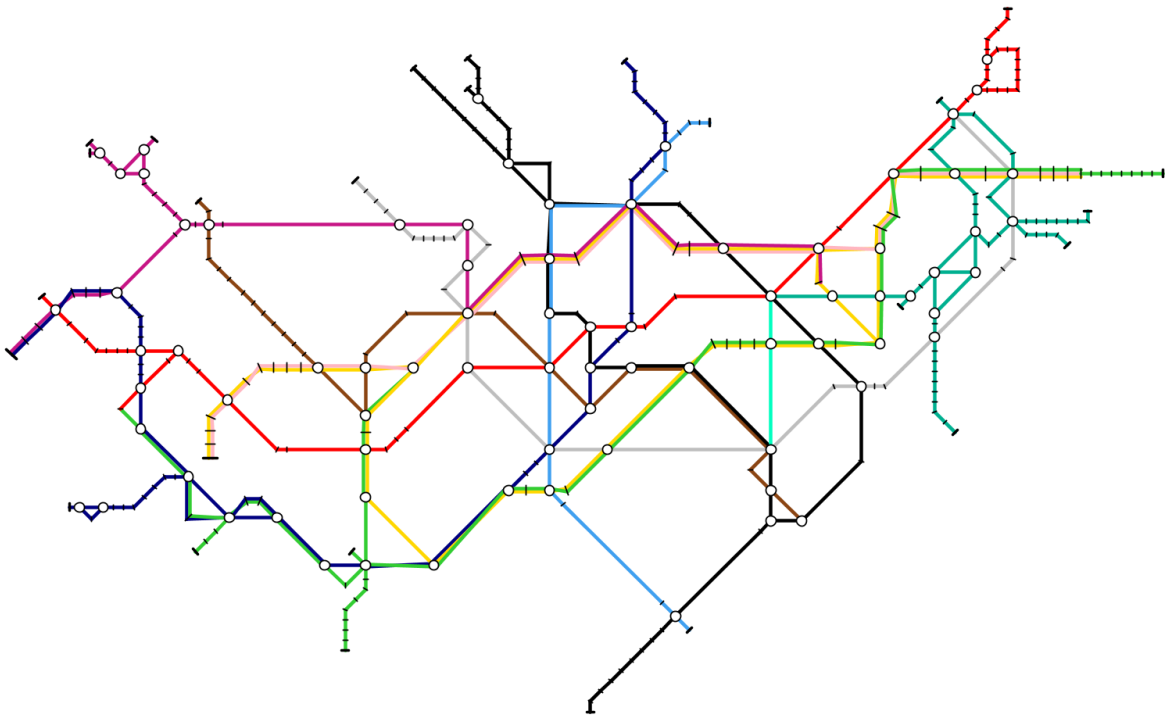


Figure A.2: London, modified schematization.

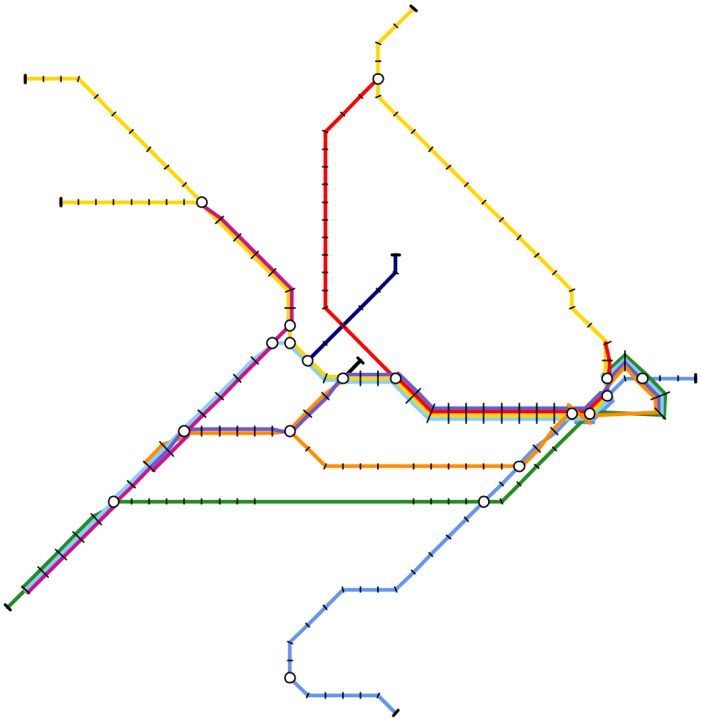


Figure A.3: Sydney, raw schematization.

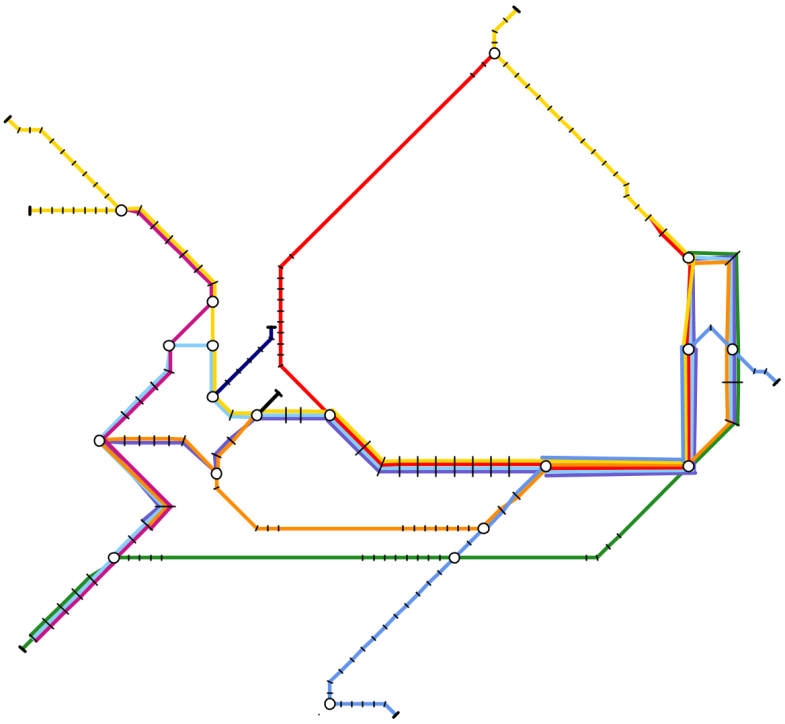


Figure A.4: Sydney, modified schematization.

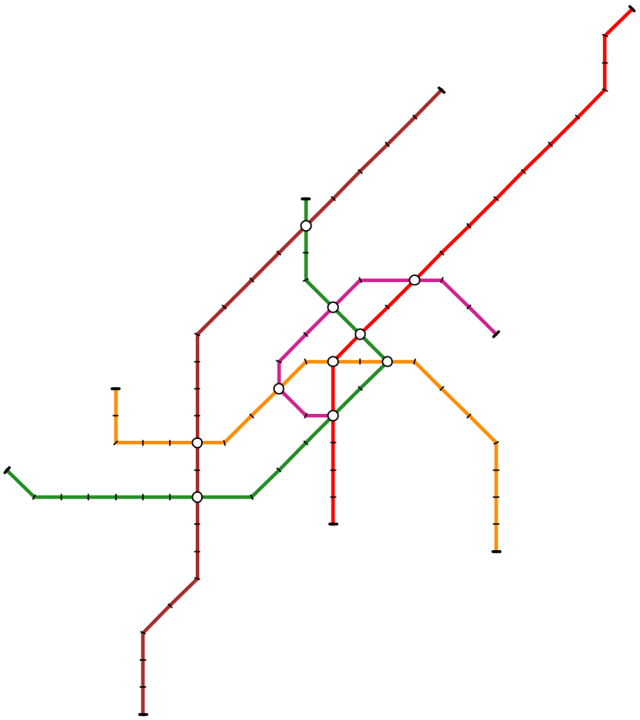


Figure A.5: Vienna, raw schematization.

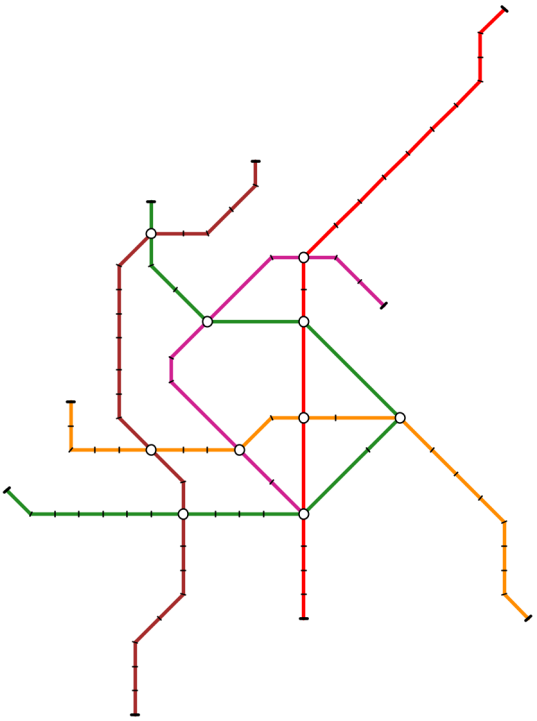


Figure A.6: Vienna, modified schematization.

Appendix B

Questionnaire questions

Our questionnaire counts a total of 180 questions. In this chapter we present a set of 18 examples — 6 different types of visualizations for the maps London, Sydney and Vienna.

List of Figures

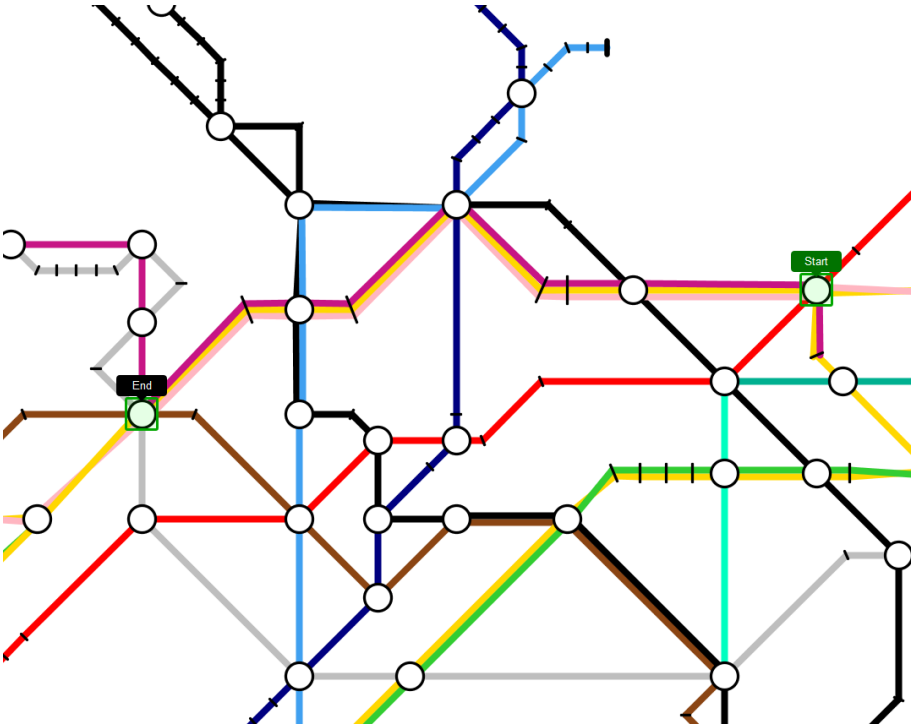


Figure B.1: London. Fixed thickness, modified schematization.

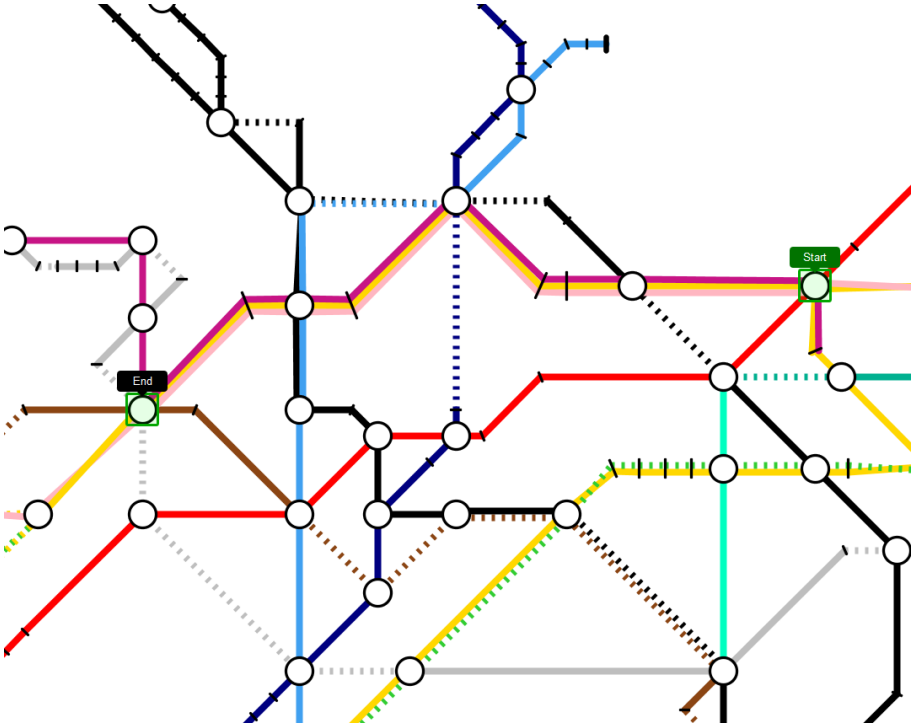


Figure B.2: London. Fixed thickness + dotted lines, modified schematization.

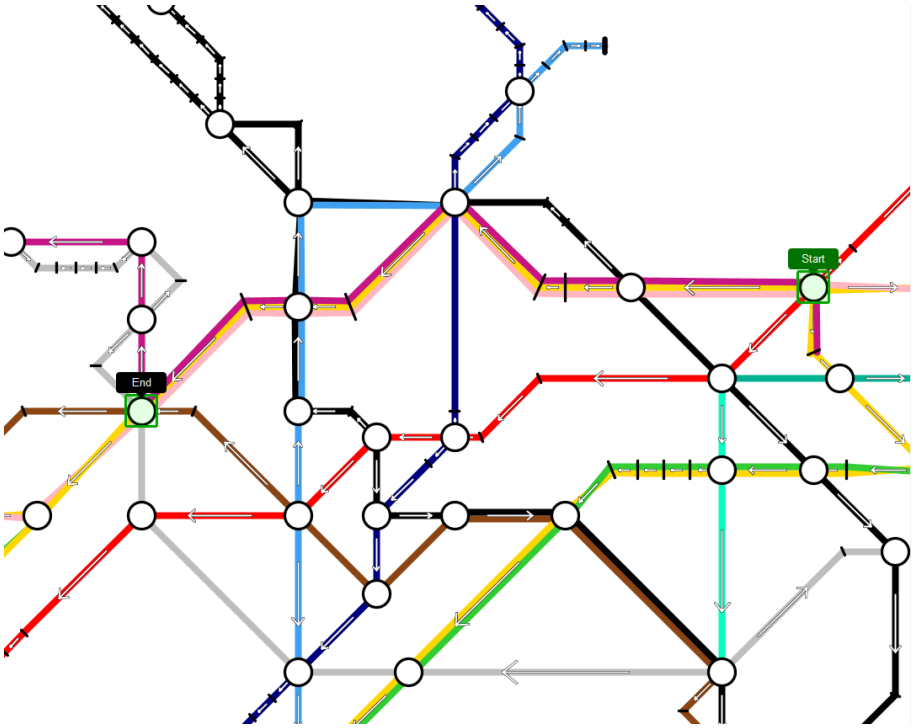


Figure B.3: London. Fixed thickness + arrow hints, modified schematization.

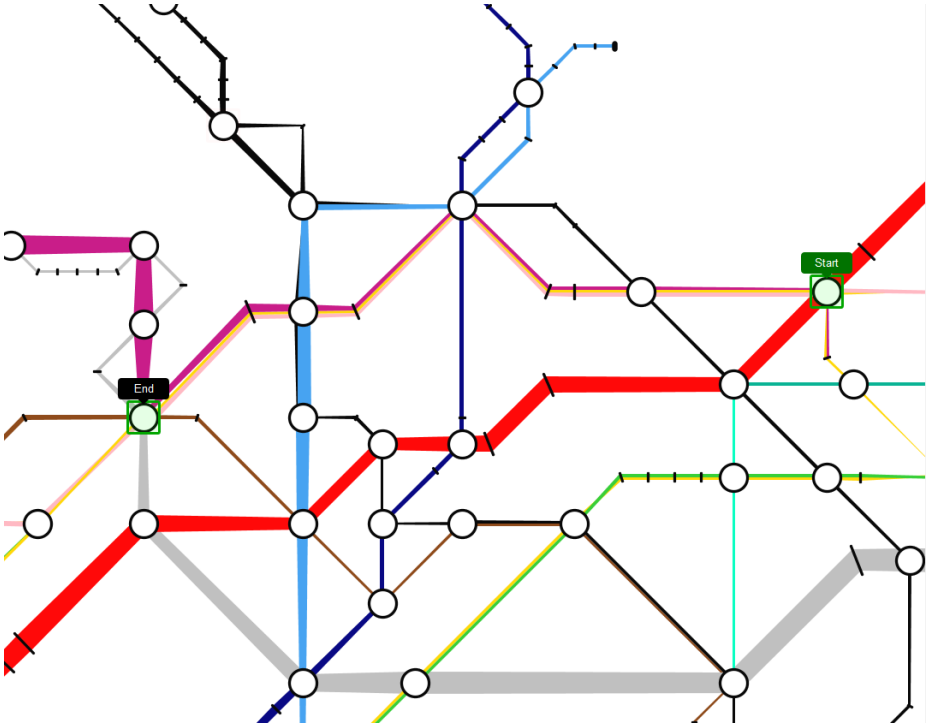


Figure B.4: London. Varying thickness (all-pairs based), modified schematization.

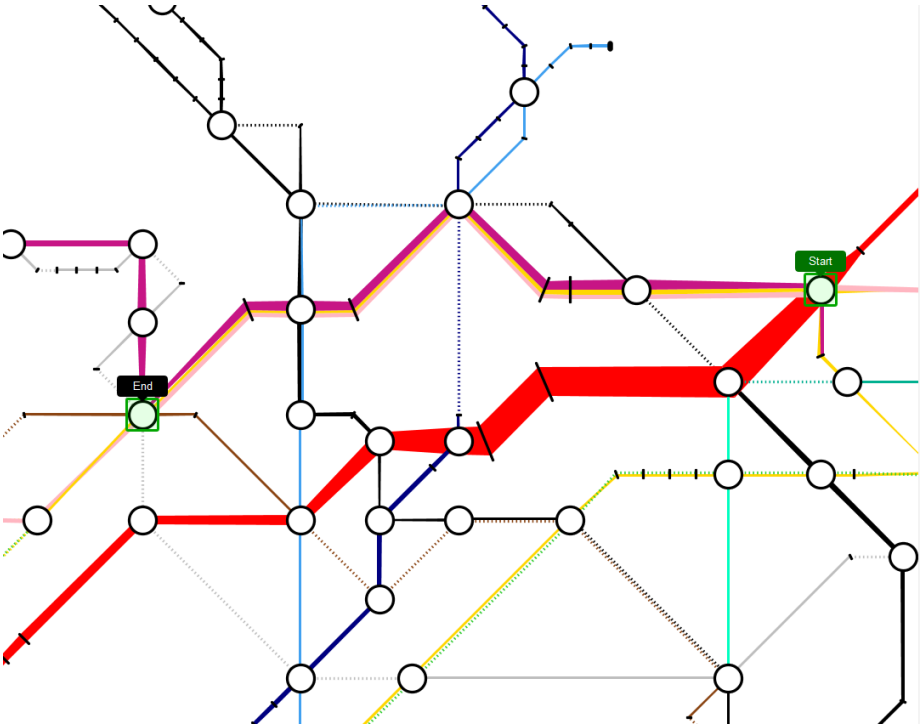


Figure B.5: London. Varying thickness (source based) + dotted lines, modified schematization.

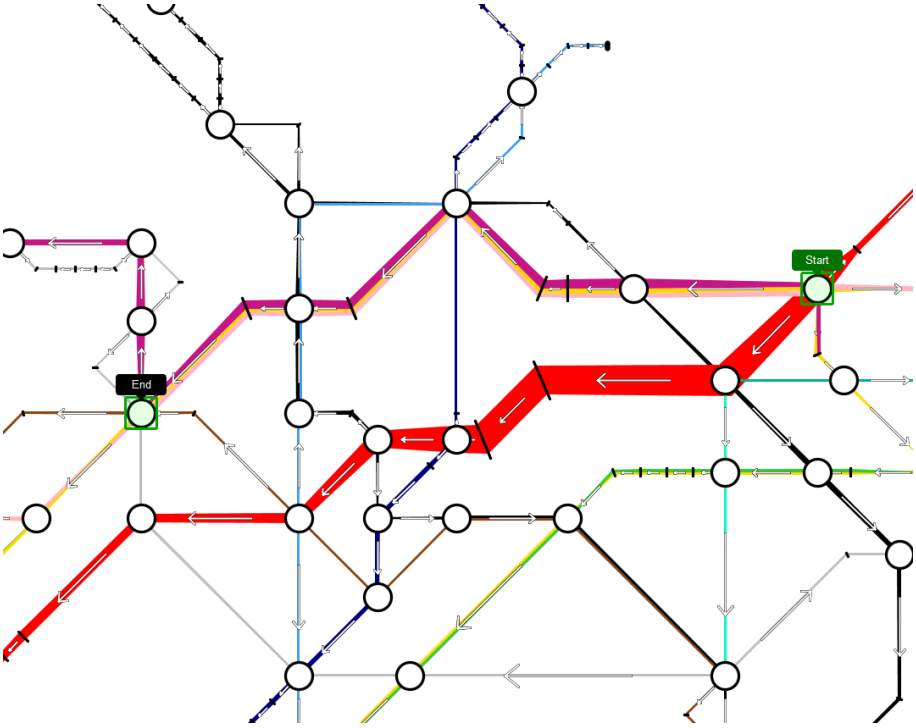


Figure B.6: London. Varying thickness (source based) + arrow hints, modified schematization.

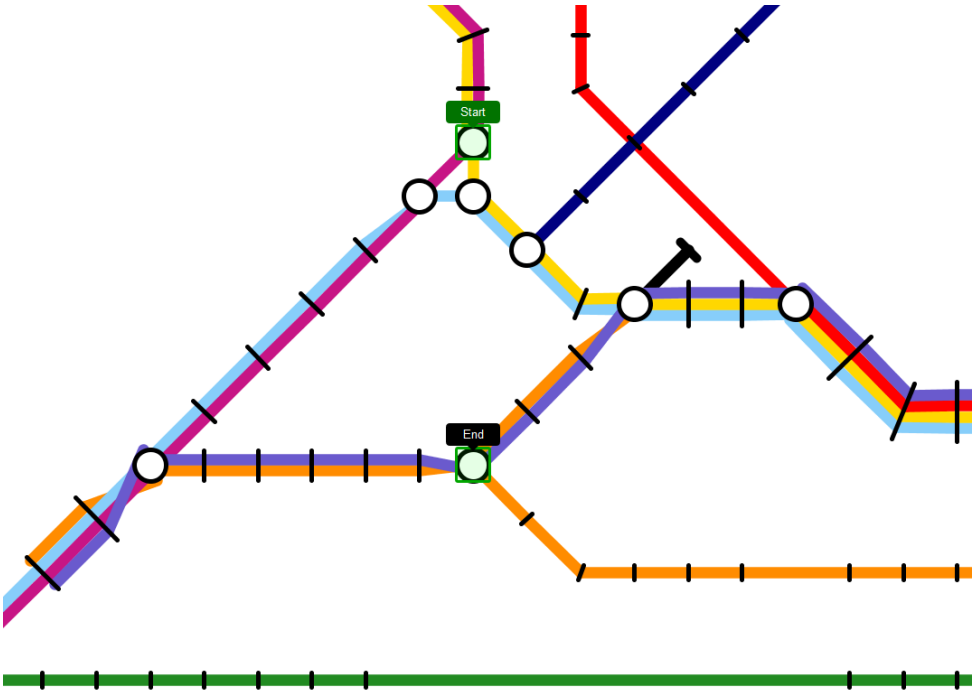


Figure B.7: Sydney. Fixed thickness, raw schematization.

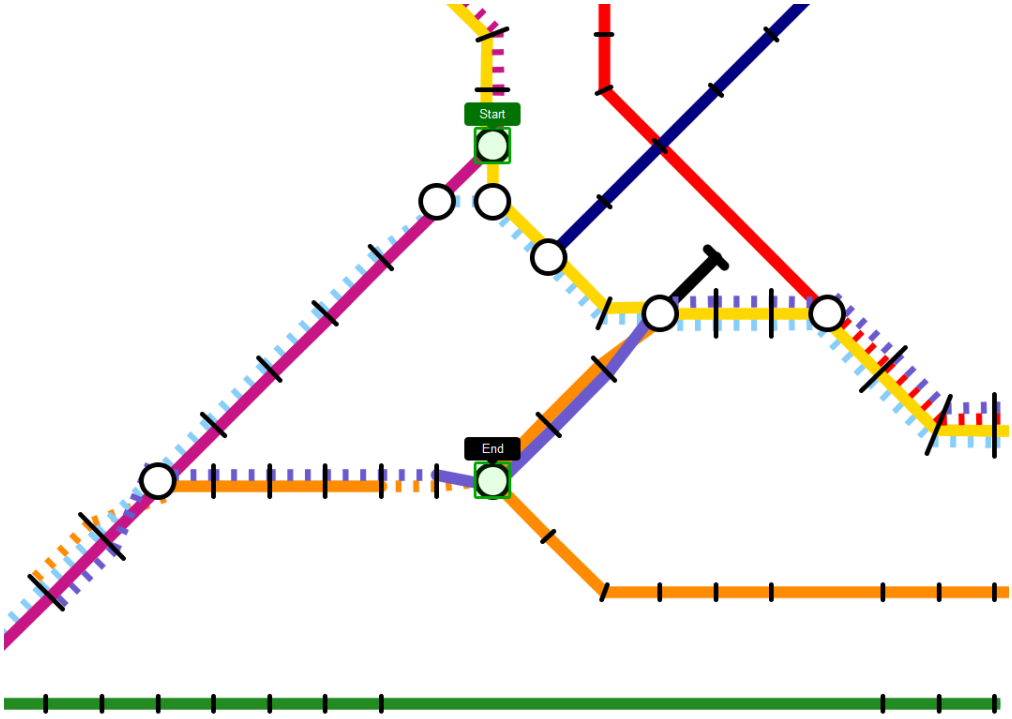


Figure B.8: Sydney. Fixed thickness + dotted lines, raw schematization.

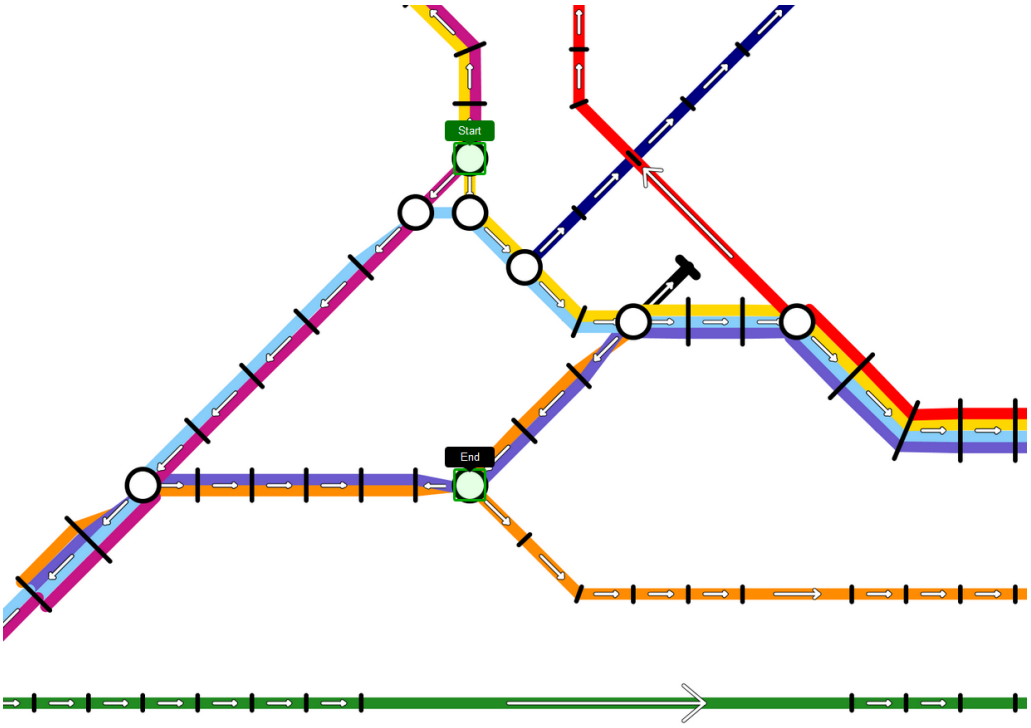


Figure B.9: Sydney. Fixed thickness + arrow hints, raw schematization.

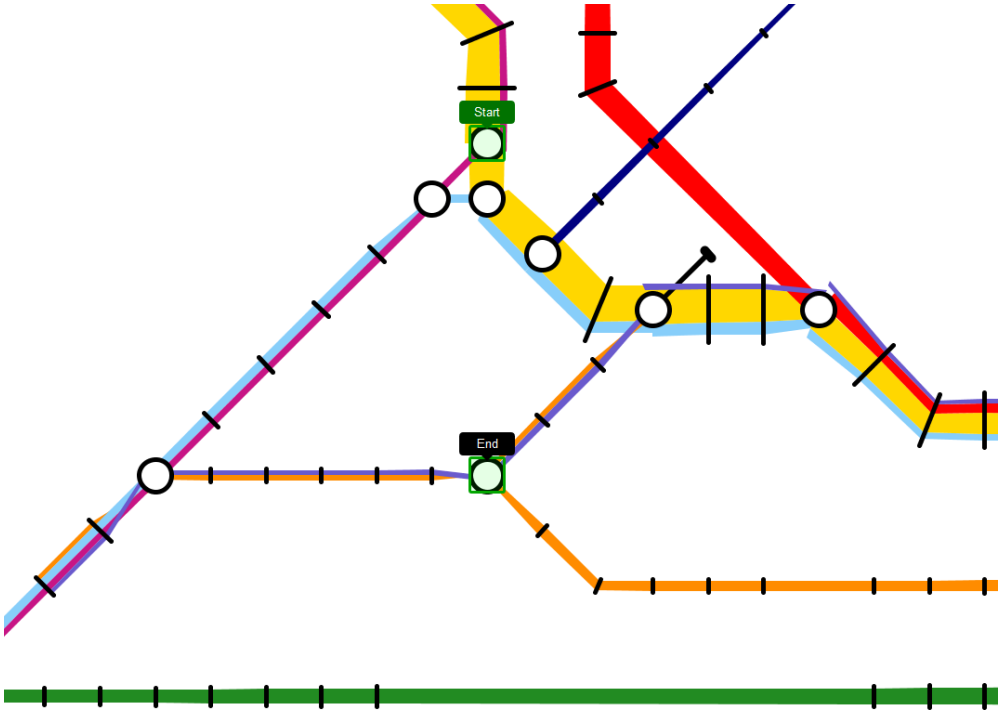


Figure B.10: Sydney. Varying thickness (all-pairs based), raw schematization.

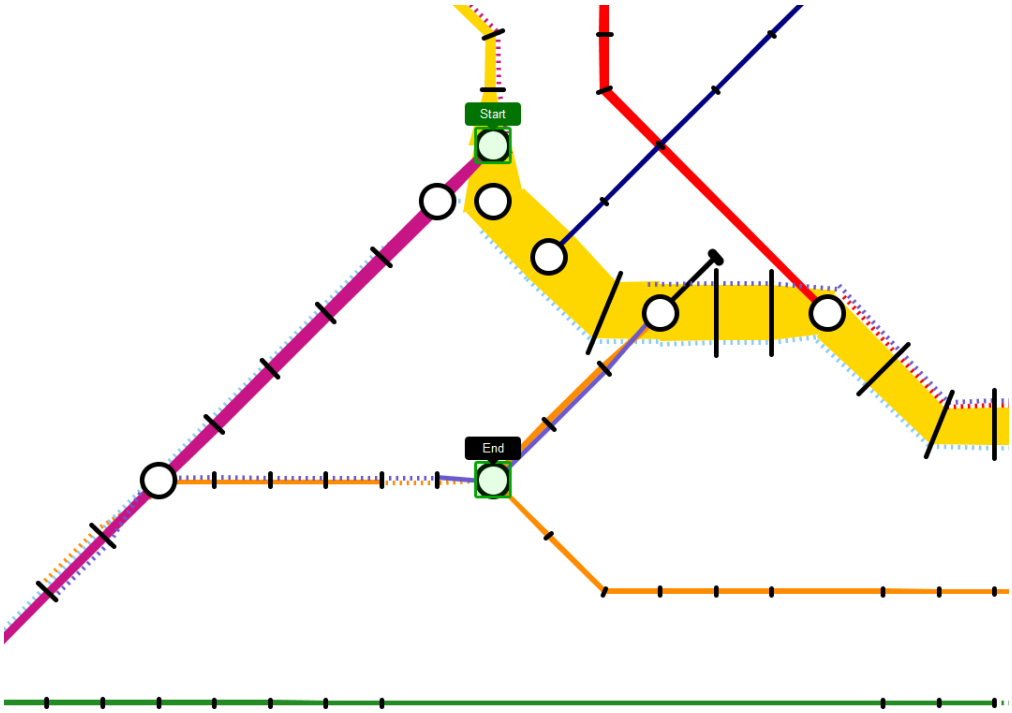


Figure B.11: Sydney. Varying thickness (source based) + dotted lines, raw schematization.

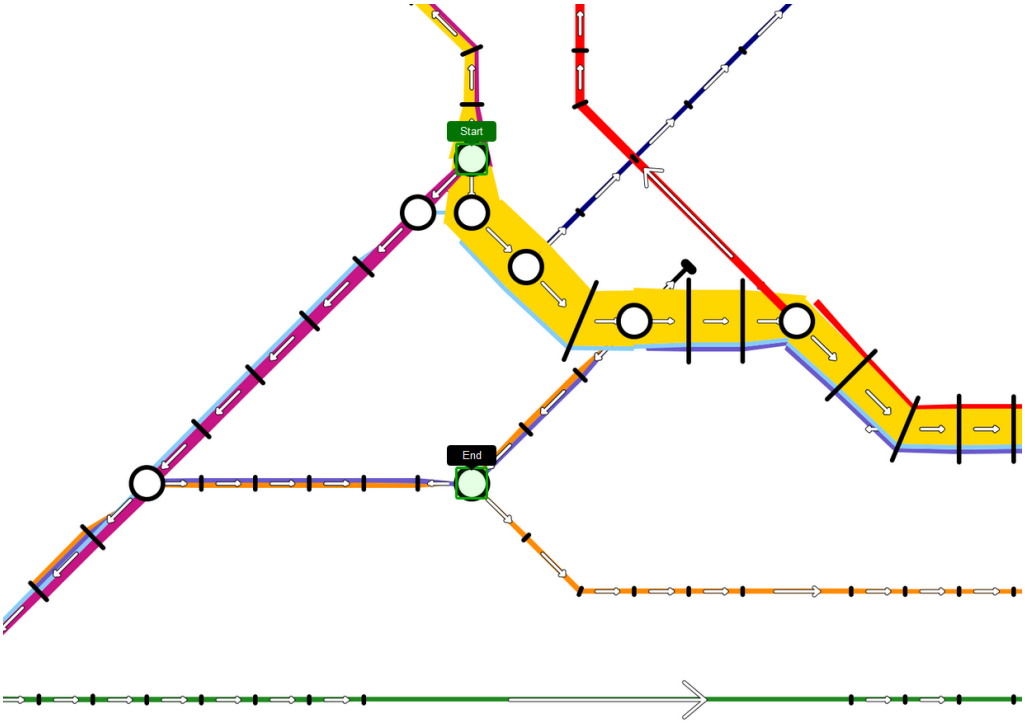


Figure B.12: Sydney. Varying thickness (source based) + arrow hints, raw schematization.

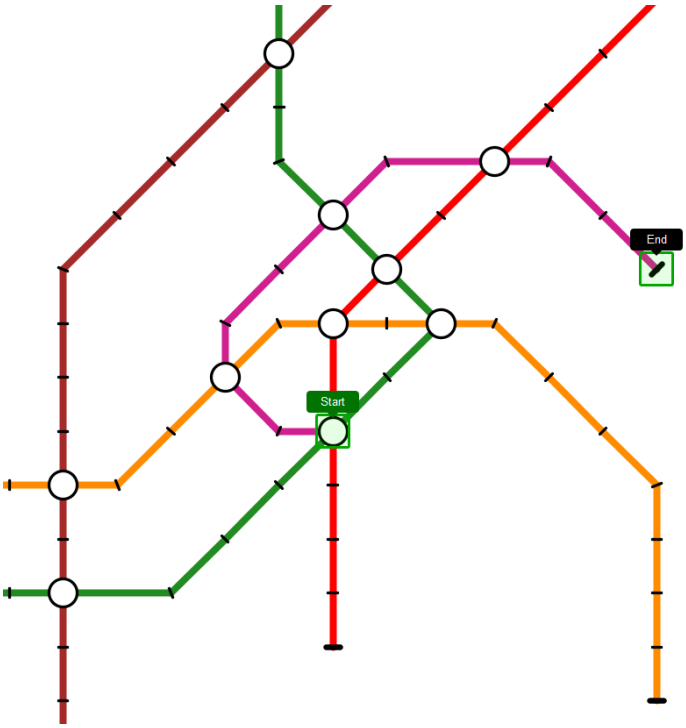


Figure B.13: Vienna. Fixed thickness, raw schematization.

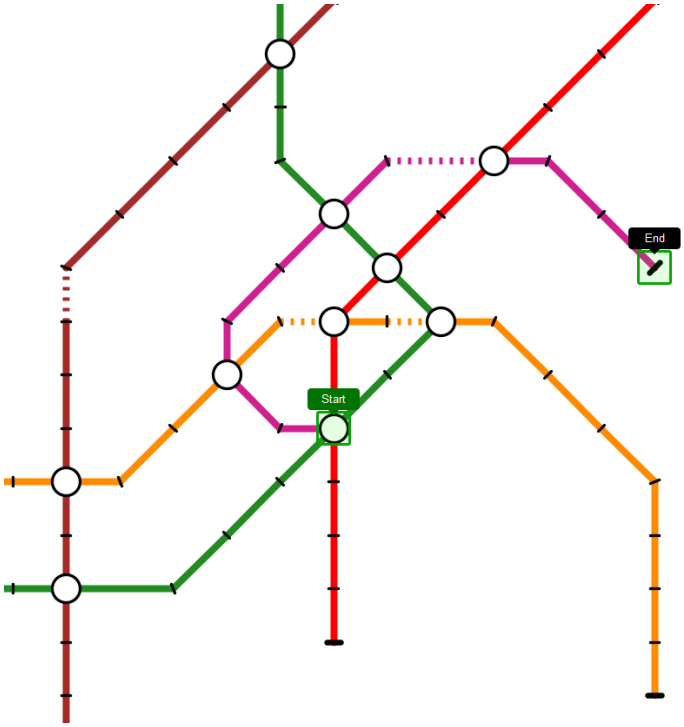


Figure B.14: Vienna. Fixed thickness + dotted lines, raw schematization.

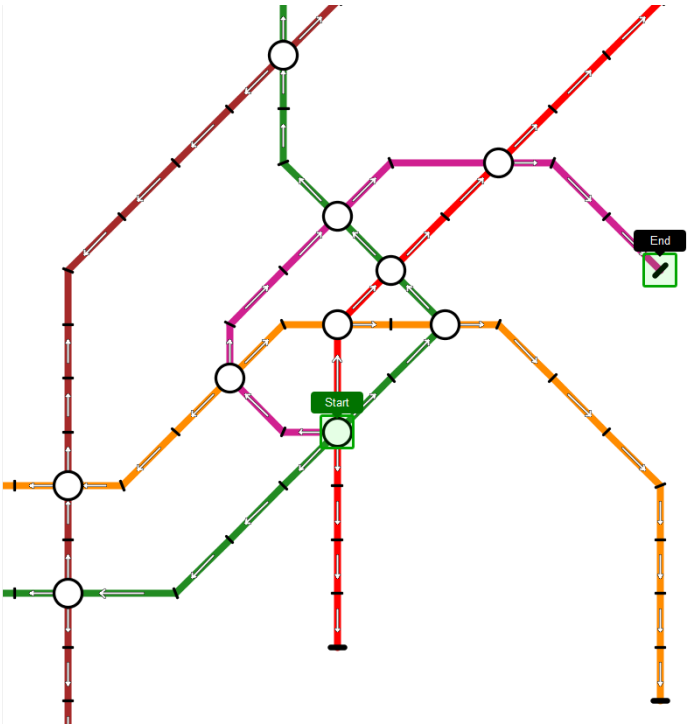


Figure B.15: Vienna. Fixed thickness + arrow hints, raw schematization.

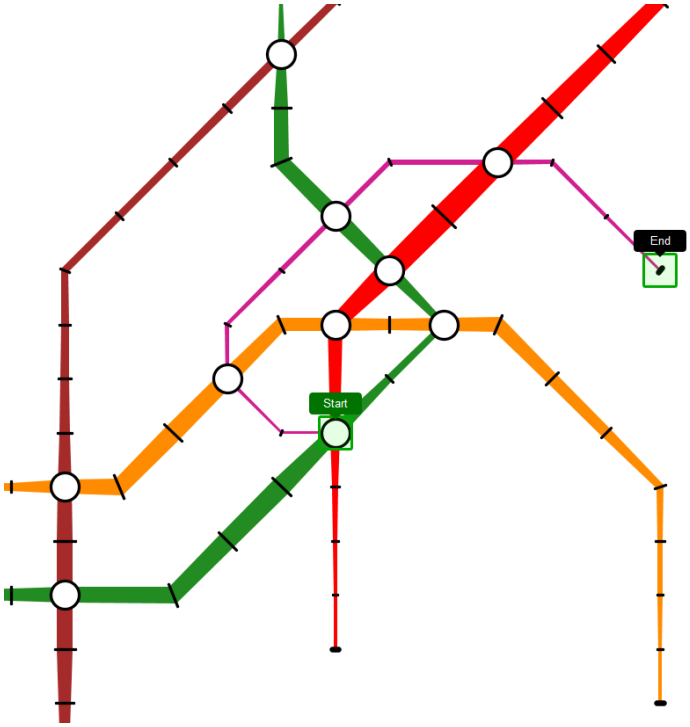


Figure B.16: Vienna. Varying thickness (all-pairs based), raw schematization.

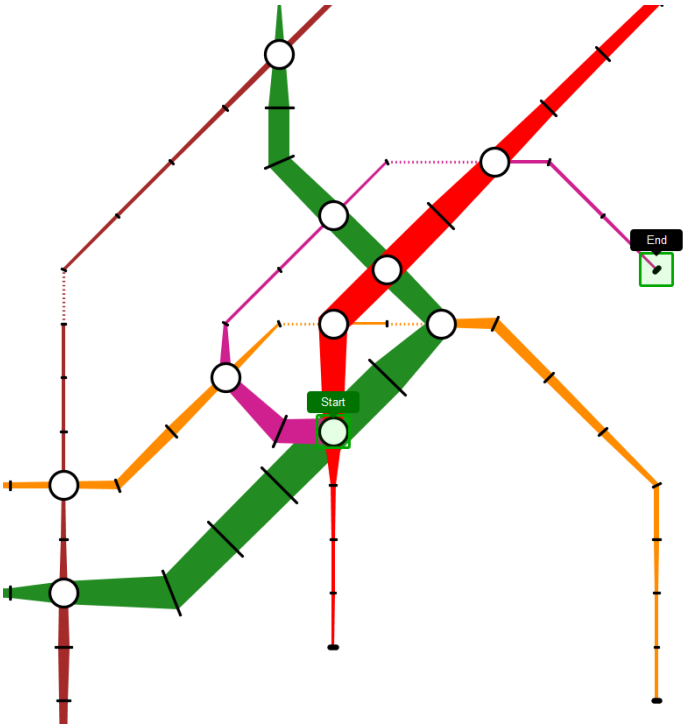


Figure B.17: Vienna. Varying thickness (source based) + dotted lines, raw schematization.

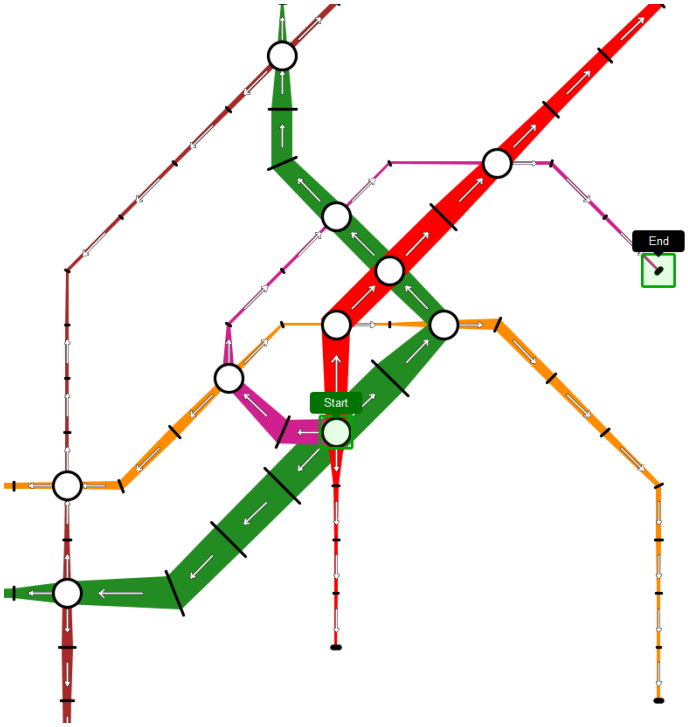


Figure B.18: Vienna. Varying thickness (source based) + arrow hints, raw schematization.

Appendix C

Questionnaire user interface

In this chapter we display screenshots of our online questionnaire.

List of Figures

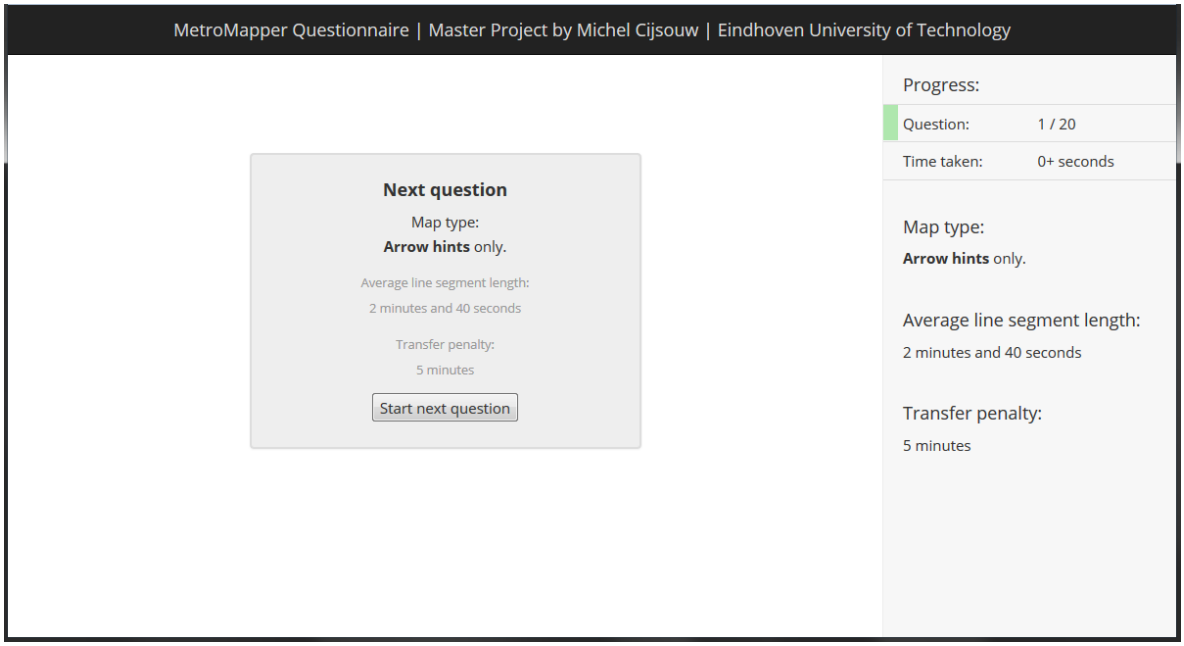


Figure C.1: Prior to each question we let the user know the type of visualization, the average line segment length and the transfer penalty.

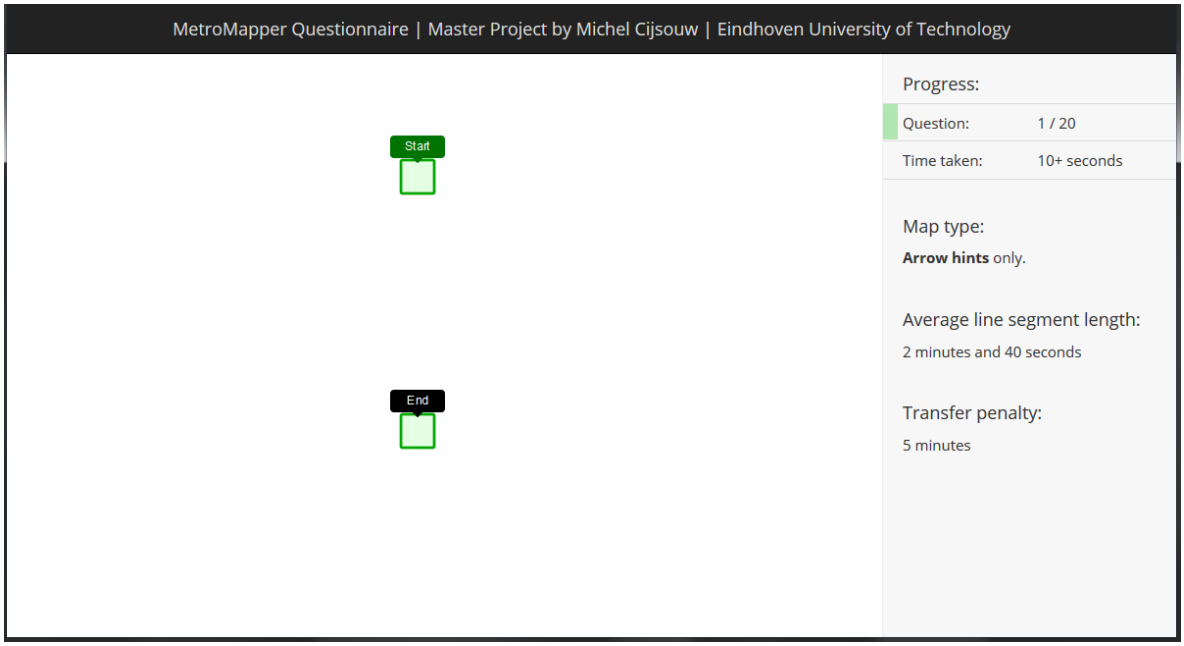


Figure C.2: First we show the positions of the starting station and destination station, so that users do not have to search for the stations before they can start planning their route. Shortly after the positions of the stations are shown we let the map fade in and we start the timer.

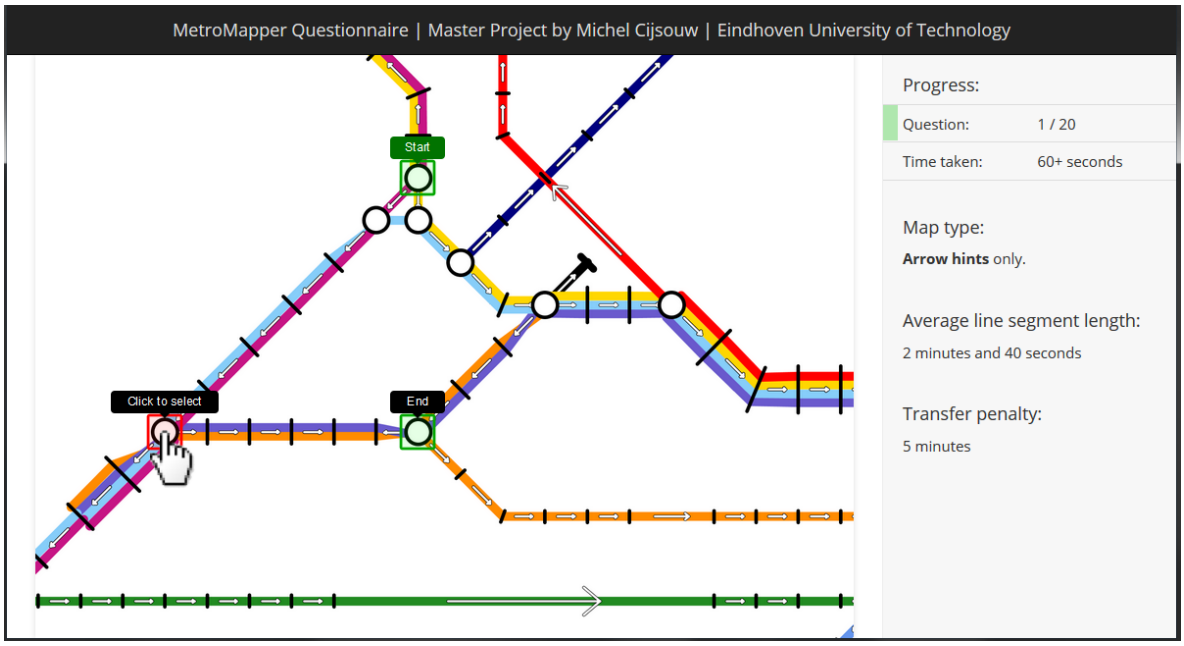


Figure C.3: Intersection stations that are part of the fastest route (according to the user) must be selected. When the mouse cursor is on a certain station a tooltip "Click to select" will pop up.

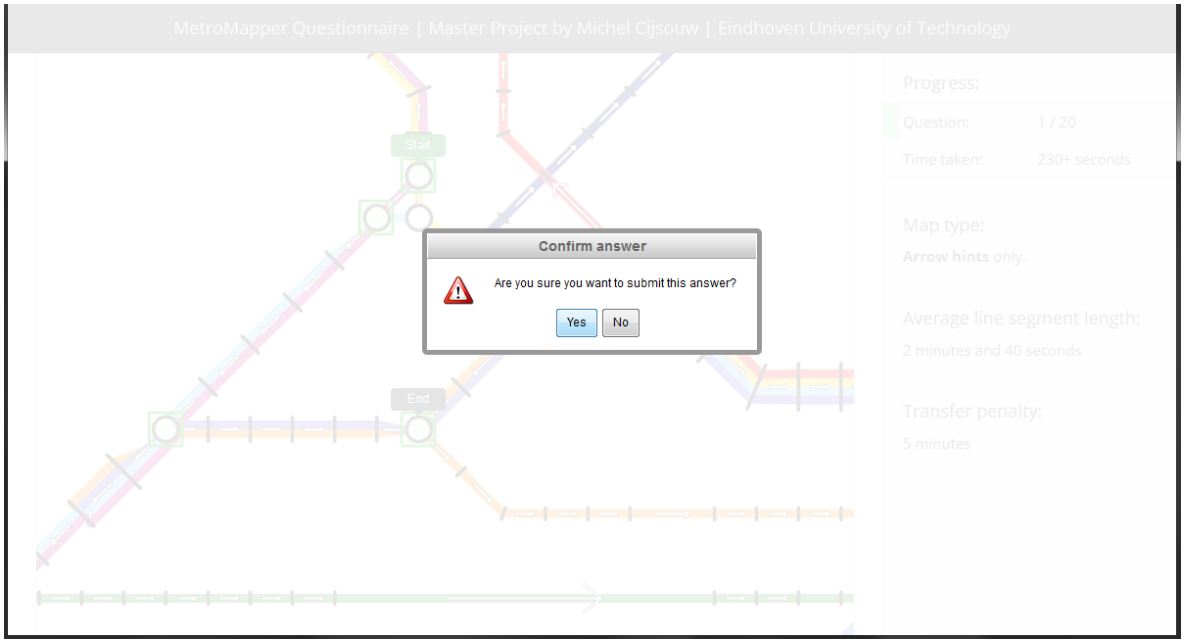


Figure C.4: After the last station (destination station) is clicked, the user is asked "Are you sure you want to submit this answer?" before actually submitting the question. While this pop-up is displayed the map on the background is occluded and the timer is paused. Users that click "No" will return to the map and the timer will resume.