Master Thesis in Geographical Information Science nr 119

Connectivity for Cyclists?
A Network Analysis of Copenhagen's Bike Lanes

Ane Rahbek Vierø

2020
Department of
Physical Geography and Ecosystem Science
Centre for Geographical Information Systems
Lund University
Sölvegatan 12
S-223 62 Lund
Sweden

Ane Rahbek Vierø (2020). Connectivity for Cyclists? A Network Analysis of Copenhagen's Bike Lanes.
Master's degree thesis, 30 credits in Master in Geographical Information Science
Department of Physical Geography and Ecosystem Science, Lund University

Connectivity for Cyclists?
A Network Analysis of Copenhagen's Bike Lanes

Ane Rahbek Vierø

Master thesis, 30 credits, in Geographical Information Sciences

Supervisors:

Mitch Selander
Department of Physical Geography and Ecosystem Science
Lund University

Ulrik Mårtensson
Department of Physical Geography and Ecosystem Science
Lund University

Acknowledgements

I would like to thank my supervisors for their guidance throughout the process and for Lund University for giving me the opportunity to study and explore the world of GIS and spatial data.
Thanks also to the City of Copenhagen and Frederiksberg Municipality for giving me access to their data, and to my colleagues at the City of Copenhagen for sharing their knowledge and expertise about cycling.

Abstract

Cycling has been identified as a central element of the solutions to some of the most pressing challenges for today's cities, such as poor air quality, rising CO_2-levels due to emissions from motorised traffic, traffic congestions, and sedentary lifestyles. A high-quality, safe, and widespread network of cycling infrastructure is a key part in the quest to encourage more people to choose the bicycle over the car. There is however a lack of methodologically sound, quantifiable, and consistent methods for evaluating networks of cycling infrastructure, partially due to issues with data availability and quality, inconsistent categorisations, and the fact that cycling as a means of transportation historically has received much less attention and funding, compared to other modes of transport. Meanwhile, there is a rich tradition of using network analysis within e.g. geography and transportation studies to describe, analyse, and model spatial networks. The purpose of this thesis is to examine how a spatial network analysis can be applied to describe and evaluate a network of cycling infrastructure, specifically the network in Copenhagen, Denmark. The analysis is centred on a range of traditional network metrics used for evaluating connectivity and accessibility, primarily based on the computation of shortest paths in a weighted graph, and applying mostly open source technologies such as PostgreSQL/PostGIS, Python, and NetworkX. From this analysis, it is evident that, despite its status as one of the world's best cities for cycling, there are substantial variations in connectivity and access within the Copenhagen network of cycling infrastructure. The analysis moreover shows that using different weighted graphs to model the network, with weights based on both geographical distance and other factors influencing the cycling experience, can give a fuller and more nuanced picture of how accessibility and connectivity vary throughout the city. Combined with data about e.g. demographics and the socio-economic status in the city's neighbourhoods, a network analysis of cycling infrastructure can thus highlight issues with areas with a disproportionately high or low network connectivity, identify overloaded network segments, and suggest where it might be beneficial to extend and strengthen the network.

Keywords: Geography, GIS, Network Analysis, Cycling, Connectivity

Contents

List of Figures

List of Tables

List of Equations

Abbreviations

ADT:	Abstract Data Type
APSP:	All-Pairs Shortest Paths Problem
CNR:	Connected Node Ratio
CSV:	Comma Separated Values
GIS:	Geographic Information Systems
LTS:	Level of Traffic Stress
ORDBMS:	Object-Relational Database Management System
SQL:	Structured Query Language
SSSP:	Single-Source Shortest Paths Problem

Glossary

Alpha: The alpha index α measures the actual number of cycles in a network, compared to the theoretical maximum number of cycles for that network.
Adjacency Matrix: A matrix used to store information about which edges connect to the same vertices.
Average Path Length: The average path length between vertices in a graph.
Beta: The beta index β describes the ratio between the number of edges to the number of vertices.
Connected Network: A graph is considered connected if there is a path from every vertex to every other vertex.
Connected Node Ratio: The number of intersections divided by the number of intersections plus the number of dangling vertices in a graph
Closeness Centrality: A measurement of reachability based on the average distance from a vertex to all other vertices in the graph.
Cycle Superhighway: A regional bike lane between Copenhagen and the neighbouring municipalities to facilitate longer distance trips and bicycle commuting.
Dangling vertex: A vertex which only is connected to one edge.
Directed network: A network in which some or all edges only allow movement in one direction.
Dual approach: A way of modelling networks in which e.g. street segments are transformed to vertices and edges are created between them if the street segments are connected.
Edge (link): The line features connecting the vertices in a graph.
Gamma: A measure based on the ratio between the number of actual, observed edges in a graph compared to the maximum possible number of edges.

Green Cycle Route: Bike routes with a focus on recreational values rather than speed and efficiency.
Incidence Matrix: A matrix for storing information about which vertices the edges in a graph connect to.
Intersection Density: The number of intersections between network segments per area unit.
Modifiable Areal Unit Problem: The problem that arises when the division of a geographical area into specific area units alter the results of the analysis and exaggerate or disguise the actual distribution of the studied phenomenon.
Network Efficiency: A measurement of how effectively movements through a network are.
Network Statistics: Statistical measurements that describe fundamental properties of a network.
Primal approach (L-Space): A way of modelling networks in which e.g. streets are represented by edges and street intersections and start/end points are represented by vertices.
Planar network: A planar graph is a graph where no edges intersect without having a vertex at the intersection.
Shortest Path: The shortest or cheapest way to move through the graph between a start and an end vertex.
Straightness Centrality: A measurement based on a comparison between the length of the shortest paths in the network and the Euclidian distance between the same vertex pairs measured as a straight line.
Street Density: The length of the edges in the network per area unit.
Stress Centrality: The number of shortest paths that passes through a network element.
Vertex Degree: The number of edges connected to a vertex.
Vertex (node): The point features comprising intersections and start and end points for edges in a graph.

Table of Symbols

Symbol	Definition
α	Alpha measures the actual number of cycles in a network, compared to the theoretical maximum number of cycles
β	Beta describes the ratio between the number of edges and the number of vertices in the graph
γ	Gamma measures the number of actual number of edges compared to the maximum possible number of edges
C_i	Clustering coefficient for a vertex
$\bar{C}$	Global clustering coefficient for the graph
d_{st}	The distance between vertex s and vertex t
E	Network efficiency
$C_D(v)$	Vertex degree for the vertex v.
$C_s(e)$	Stress centrality for the edge e
$\sigma_{st}(e)$	Number of shortest paths between vertex s and vertex t which contain the edge e
$C_c(s)$	The closeness centrality for vertex s
$C_s(s)$	Straightness centrality for vertex s
d_{st}^{Eu}	The straight Euclidian distance between vertex s and vertex t

Table 1. Symbols in order of appearance

1. Introduction

In an increasingly urbanized world, cycling is often brought out as a solution to issues ranging from congested roads, air and noise pollution, sedentary lifestyles, lacking mobility for those who cannot drive a car, as well as the looming problem of rising CO_2-levels. Copenhagen, Denmark is often lauded as one of the biking capitals of the world due to its extensive network of bike lanes and the fact that more than half of its population choose the bike for their daily commute (Kirschbaum, 2019; Colville-Andersen, 2017; City of Copenhagen, 2019a). In a city that is growing every year, both in size and population (Ekeroth, 2019), the current cycling infrastructure is however not sufficient, especially not since the city hope to have even more people biking in the future, in its attempt to become CO_2-neutral by 2025 (City of Copenhagen, 2016a). Moreover, Copenhagen has not always been a bike-friendly city and the current conditions should thus not be taken for granted (Henderson and Gulsrud, 2019).

The planning and prioritisation of cycling infrastructure in Copenhagen is today relying on a wide variety of factors and inputs, ranging from the city planner's background knowledge and experience of the city, traffic counts, citizen inputs, and political prioritisations (Henderson and Gulsrud, 2019, City of Copenhagen, 2017a). To ensure that no area is overlooked, and that funding is spent where most needed, it is however necessary with a comprehensive overview of the cycling network. The use of network analysis in relation to cycling infrastructure is often overlooked and underutilised. It does nevertheless have a large potential for improving the knowledge and quality of the network of bike lanes, since traditional metrics from network analysis, such as network density and connectivity, are important factors for the number of people choosing to bike (Schoner and Levinson, 2014). Cycling infrastructure in cities, which to a large extent simply consist of cycle lanes and paths, can however be hard to study, since they often are treated as an addition or extension to the road network, rather than a network in its own right. There are therefore often issues with data quality, missing links, and unconnected segments – both in the datasets and in the actual infrastructure (Topjian, 2019*).* Although the past years have witnessed a growing body of research into the application of network analysis on cycling infrastructure (Schoner and Levinson, 2014; Boisjoly and El-Geneidy, 2016; Kent and Karner, 2019), a knowledge gap still exists when it comes to adapting transportation research and network analysis to the context of cycling. There is thus not yet a sufficient understanding of how to approach cycling infrastructure as a system of interdependent parts, rather than as bike lanes evaluated on an individual basis.

Aim & Research Question

My aim with this thesis is to conduct an evaluation of the network of bike lanes in Copenhagen in order to firstly, examine the state of the current network, and secondly, to explore how GIS[1] and spatial analysis can be used to evaluate networks of cycling infrastructure. The evaluation will be based on a network analysis of the cycling infrastructure in Copenhagen, with a focus on determining network density and connectivity, as well as measuring the importance and centrality of individual network segments. The goal is to demonstrate how a network analysis can be used to, for example, identify areas that are not sufficiently connected to or covered by the existing network, compared to e.g. the city average or other cities, and in this way help close the knowledge gap on how to think more systematically about cycling infrastructure. To do this, the analysis will primarily make use of data describing the location and type of cycling infrastructure in Copenhagen.

A cyclist's experience of the quality and connectivity of a network of bike lanes is however not only influenced by simple network measures such as distance or network density. The analysis will therefore be conducted as an extended network analysis, which not only looks at traditional network metrics, but also attempts to take other factors into account, and evaluate the network based on the local context. This for example entails including indicators of the quality of the bike lanes and the biking experience in the analysis, to get a more comprehensive view of the network quality at both a local and city-wide level. As part of this, I will examine how various categorisations of network segments can be incorporated into a network analysis, to take into account that not all segments of a network are 'created equal' but rather have different characteristics. In this specific example, this for example includes the categorisation of the different types of bike lanes and routes in Copenhagen (see p.6-8) combined with various methods for ranking network elements, depending on how safe or stressful they are for cyclists (see p.10). To achieve this part, data on the road network, car traffic, and land use in Copenhagen will be used.

The analysis will finally correlate and compare potential variations in the network quality across the city with the differences in neighbourhood characteristics such as population density and average income, to look for any patterns in where the network of cycling infrastructure has a particularly high or low connectivity. This correlation will be tied to a wider discussion of how GIS and specifically network analysis can be applied

[1] GIS is in this context simply referring to a type computer software which can be used for storing, processing, analysing, and visualising geodata (Harrie, 2014).

within the planning of cycling infrastructure. The goal is accordingly to both outline new knowledge and ideas about how a network analysis can be performed and to produce applicable knowledge of the current state of the Copenhagen cycling infrastructure.

The overall research question, which this thesis attempts to answer is:

- *How can a GIS-based network analysis be utilised to evaluate cycling infrastructure?*

This question will be answered through the following secondary research questions:

  - What can the use of measures of connectivity and centrality reveal about a network of cycling infrastructure?
  - How can the computation of shortest paths be used to evaluate a network of bike lanes?
  - How and to what extent can characteristics and qualities beyond connectivity and distance be incorporated into a network analysis?
  - In what way can a network analysis of transportation infrastructure be used to inform urban planning efforts?

The hypothesis is that different metrics not only will affect how a network is perceived in terms of accessibility, network density, etc., but that the incorporation of additional variables also will enable a closer examination of variations across the network, compared to an analysis solely based on metric distance.

Outline

In the upcoming second chapter I will describe the study area and context for the research question, particularly the historical background and current conditions behind Copenhagen's status as a cycling city. While the knowledge presented here is not directly part of the analysis, it forms the necessary background and framework for the interpretation of the results. Chapter three will cover all the theoretical and analytical concepts used for the analysis, such as how to analyse transportation networks and the conceptualisation and study of networks based on graph theory. Both chapter two and three are based on a multidisciplinary literature study of materials from the fields of GIS, mathematics and graph theory, computer science, and urban planning. Chapter four will describe the methodology, including the tools applied throughout the analysis, the datasets which the analysis is based on, how data have been obtained and prepared, as well as the overall analytical approach. The outline of the analysis will be followed by a fifth chapter in which the results of the analysis will be presented. Finally, chapter six is dedicated to a discussion of what the results suggests about the state of the cycling network in Copenhagen, whether any differences in accessibility might be correlated to the characteristics of different areas, how the method might be applied within transportation and urban planning, methodological limitations, and ideas for future research.

2. Study Area

Copenhagen

Copenhagen is the capital of Denmark and has an urban population of approximately 736,000 inhabitants (Statistics Denmark, 2020a), a number which however has been steadily increasing for the past decades (City of Copenhagen, 2018a). Copenhagen has a population density of around 6,000 people per square kilometre, although the population density varies greatly within the city (The City of Copenhagen, 2016b). The city is divided into two separate municipalities, *The City of Copenhagen* and *Frederiksberg Municipality*, a legacy from the municipal reform from 1901. The vast majority of the city is part of the City of Copenhagen municipality, which complete encloses Frederiksberg. Combined, the two municipalities represent 12 neighbourhoods (see Figure 1), covering approximately 100 square kilometres (Statistics Denmark, 2020b).

Functionally, Copenhagen is closely integrated with the surrounding suburbs, which nevertheless are separated on the administrative level. The entire urban area is usually referred to as *Copenhagen metropolitan area*, with a population around 1.3 million inhabitants (Statistics Denmark, 2020c).

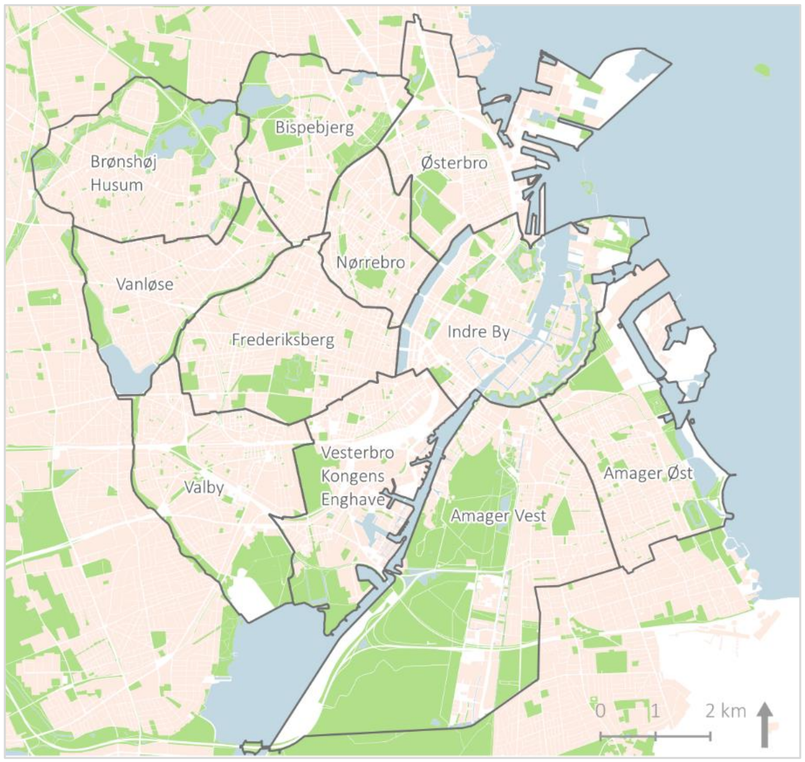

Figure 1. Neighbourhoods in the City of Copenhagen and Frederiksberg

Copenhagen can trace its history back to around year 900. The city is located on the eastern coast of Denmark, towards Sweden, and has since its start as a small fishing town grown both inland, towards west, as well as adding entire new areas of land claimed from the sea, a practice which continues today[2]. The location by the water has thus been formative for the growth and shape of Copenhagen, with a large part of the city only accessible from a small

[2] See for example the new project 'Lynetteholmen', which is an entirely new island constructed in the Copenhagen harbour: https://www.lynetteholmen.com/

number of bridges connecting Amager with Zealand and the rest of the city. The city centre is at the present time still dominated by buildings constructed around the sixteenth and seventeenth century, while many of the most central and densely populated neighbourhoods have maintained much of the original layout from their construction around the end of the nineteenth century (Samson, 2019). The city is thus still characterised by rather narrow and slightly winding streets and a fairly dense urban core, at least compared to the much younger cities in e.g. North America, with their more symmetrical grids and often much wider streets and longer blocks (Boeing, 2019). Copenhagen has however also in the past decades added entire new neighbourhoods to the city, some of which have faced extensive criticism for their more modern layout of the street network, which is claimed to accommodate car driving at the cost of cyclists and pedestrians (Corneliussen, 2002). These are not just curious historical details, but an important piece of information in an analysis of a network that to a large extent is shaped after the city's street grid.

Cycling in Copenhagen

Although much of Copenhagen's present bike lanes have been constructed within the past 30-40 years (Colville-Andersen, 2014), the city has a long history of cycling. Denmark's first bike lane was constructed in Copenhagen in 1892 as cycling became more and more popular (Dietrich, 2009), but the city was nevertheless heavily dominated by car traffic throughout the twentieth century. The City of Copenhagen has been working strategically to improve cycling conditions and increase the number of cyclists since the 1990's and adopted its first official cycling strategy in 2002. The strategy set up concrete targets for, among other things, the modal share of cyclists, improvement and extension of the physical infrastructure, as well as for the cyclists' experience of safety and comfort (City of Copenhagen, 2002). Today the City of Copenhagen (not including Frederiksberg) has around 382 kilometres of designated bike lanes, of which the majority are elevated and physically separated from car traffic by a curb (City of Copenhagen, 2019b). In 2018, 28% of all trips and 49% of all commuter trips in the municipality were done on a bike, which is just short of the official goal of a commuter modal share of 50% for bikes (ibid.). The high rates of cycling are however also reflected in the number of cyclists who experience the bike lanes as too narrow and crammed (ibid.), which underlines the importance of strategic improvements to the current network.

Although bike lanes are a seemingly simple type of physical infrastructure, the design solutions and standards differ widely between places, and sometimes even within the same city (Andersen, 2017). The bike lanes in Copenhagen are generally of a high standard, with raised curbs and a minimum width of 2.2 metres, although part of the network does consist of paths with a width of minimum 1.5 metres, which only are separated from motorised traffic by a painted white line (City of Copenhagen, 2013) (see Figure 3 and Figure 4).

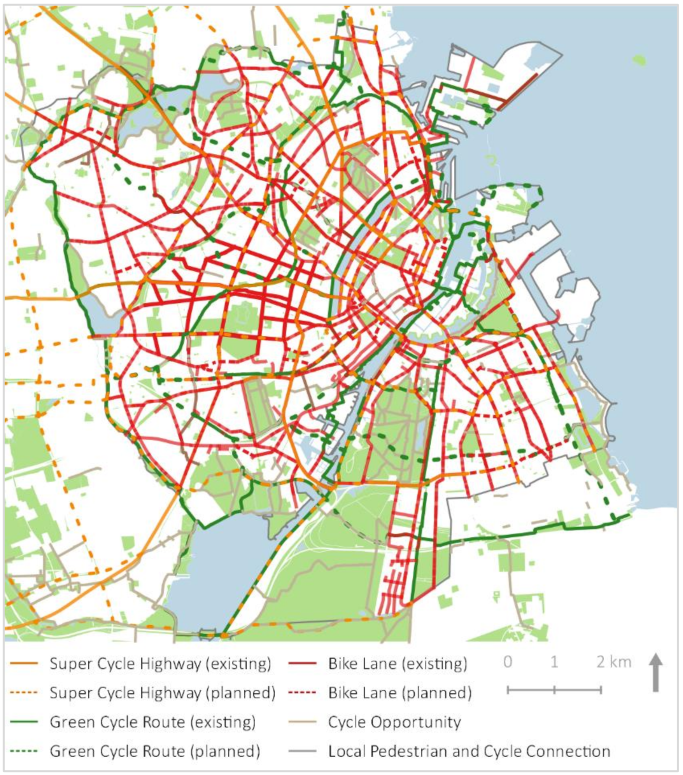
Super Cycle Highway (existing)
Super Cycle Highway (planned)
Green Cycle Route (existing)
Green Cycle Route (planned)
Bike Lane (existing)
Bike Lane (planned)
Cycle Opportunity
Local Pedestrian and Cycle Connection
0
1
2 km

Figure 2. The entire network of cycling infrastructure in Copenhagen

In Copenhagen, the bike lanes are usually one-directional and placed on both sides of a street, although two-directional bike lanes are used occasionally, for example for bike paths that do not follow a regular street (e.g. bike lanes in parks).

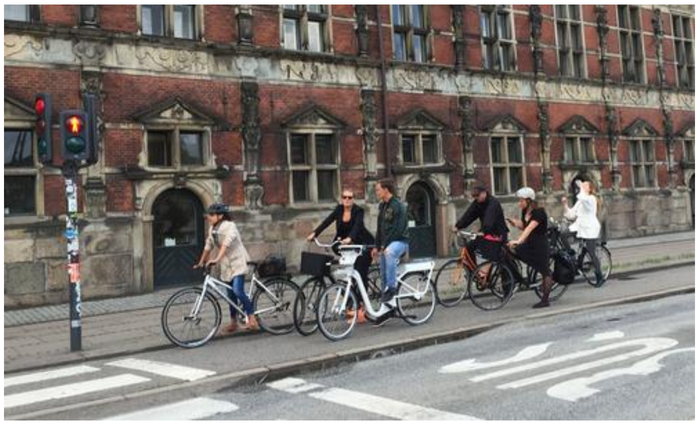

Figure 3. Bike lane with raised curb (Alpert, 2016)

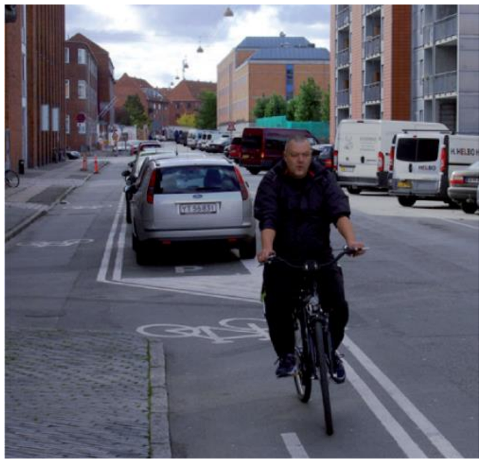

Figure 4. Bike lane marked with a painted line (City of Copenhagen, 2013)

Besides the designated bike lanes, the network includes so-called 'cycle streets', which are streets with mixed traffic, but where the cars must give way to bikes. The network moreover includes the *Green Cycle Routes* and the *Cycle Superhighways* (see Figure 2 and Figure 5). The Green Cycle routes are bike-friendly routes with a focus on recreational values rather than speed and efficiency, and which may or may not follow designated bike lanes (ibid.). The network of Green Cycle Routes spans around 63 kilometres, which also includes pedestrian areas as well as bridges for cyclists and pedestrians (City of Copenhagen, 2019b). The inclusion of pedestrian areas, bridges etc. is the main reason that this thesis refers to the network as 'cycling infrastructure', since the network consist of other types of infrastructure than just bike lanes – although bike lanes do form the backbone of the network.

The Cycle Superhighways are a regional project between the City of Copenhagen and the neighbouring municipalities. The project focusses on facilitating longer-distance commuting on bike between Copenhagen and the surrounding suburbs and towns, through constructing and maintaining a number of extra wide bike lanes on strategic stretches in the participating municipalities, making sure that intersections are optimised for cyclists, etc.

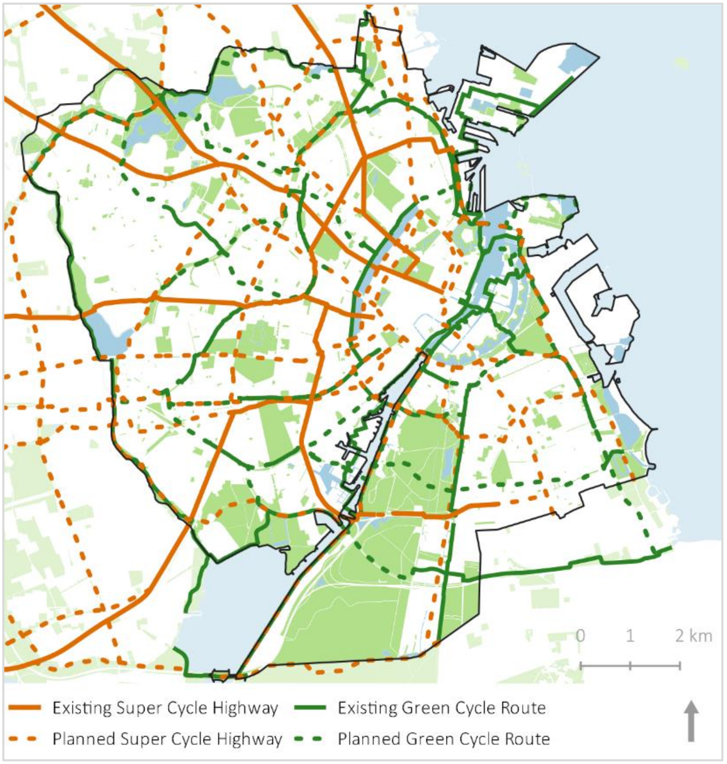
0 1 2 km
Existing Super Cycle Highway
Planned Super Cycle Highway
Existing Green Cycle Route
Planned Green Cycle Route

Figure 5. Cycle Superhighways and Green Cycle Routes in and around Copenhagen

The network of cycling infrastructure in Copenhagen is thus closely and deliberately connected to cycling infrastructure in surrounding municipalities. It could therefore be argued that analysing the network at a municipal scale does not capture the full extent and characteristics of the network of cycling infrastructure in the Copenhagen metropolitan area. In this way the analysis risk making what in geography is referred to as 'the modifiable areal unit problem', in which the division of a geographical area into specific districts can alter the results and potentially exaggerate or disguise the actual distribution of the studied phenomenon (Altaweel, 2018). While this is something to be kept in mind throughout the analysis, the cycling infrastructure in the various municipalities in and around Copenhagen are, in the planning process, mostly approached as separate networks, or at best as connected sub-graphs in the same network. Although interesting results undoubtedly could be obtained from approaching the entire region as one, connected network, I believe that it makes sense to consider the networks at a municipal level, since this is the spatial scale which most of the planning decisions are taking place on.

3. Theoretical Framework

In this chapter I will present the theoretical framework for the analysis. The chapter will begin with an outline of the perspective on transportation networks which will be applied throughout the analysis. In the following section I will give a brief presentation of what the spatial perspective entails for the methodology and the outcome of the analysis. Finally, the formal definition of a network, how it is abstracted, and the concepts and terms used to analyse and examine the network will be explained.

Accessibility & Equity in Transportation Networks

There are numerous different ways of answering the question of how to utilise a network analysis within urban planning. The section below will present the approach I will apply in the upcoming analysis, which primarily is concerned with how the network analysis can be used to highlight issues with inequity and lack of access.

The following will be an analysis of not just a network, but a transportation network, meaning that it is a system designed to overcome space and frictions such as distance, time, and topography (Rodrigue et al., 2009). When analysing transportation networks, focus can be on distance, accessibility, spatial interaction (actual flows of objects, people or information between places), or transportation/land use models (a more complicated framework for analysing relationships and feedback mechanisms between transportation flows and the supporting spatial structure) (ibid.). As mentioned above, the focus here will be on distance and accessibility.

Distance can be defined in numerous different ways, such as Euclidian distance, travel time, travel cost, etc. Accessibility is generally understood as a measure for the extent to which a location can reach or be reached from other locations (ibid.), but can otherwise be conceptualised quite differently, depending on the perspective. Some studies approach accessibility in a network as improving the overall efficiency and reachability of the transportation network, while others consider access from an equity perspective, with an aim of ensuring an equitable distribution of access throughout the network (Litman, 2020). Equity can again be understood either as so-called horizontal equity or as vertical equity. Horizontal equity means that everybody and every location preferable should have the same level of access (Bhuyan et al., 2019), which for example can be implemented as a completely uniform network density throughout the city. Others approach equity in terms of the outcome, also known as vertical equity, which entails that the focus is on directing and redistributing resources towards those most in need, not just on ensuring universal access (ibid.). Within the context of cycling this could for example mean providing a higher level of accessibility to areas or people, who are otherwise underserved in terms of transportation infrastructure or resources in general. Several studies have already evaluated cycling infrastructure within an equity and accessibility framework (e.g. Bhuyan et al., 2019 and Kent and Karner, 2019). The ambition in the following analysis is not to define one specific measure for an equitable configuration of the cycling network in Copenhagen, but it will be considered how the various network measures vary compared to common socio-economic metrics, such as income and education.

Planning or evaluating a network for cyclists in terms of accessibility also includes considering that a city's cyclists are a very heterogenous group – particularly in a city with as high a proportion of cyclists as Copenhagen. While some commute by bike, others use a bike for recreational purposes or exercise, just as people differ by age, cycling experience, etc. Even though cyclists generally are more sensitive to factors such as distance, noise, and feeling unsafe in traffic than people in cars or on public transportation, the willingness to tolerate detours, closely passing cars, or unregulated intersections vary considerably between different segments of cyclists (Schoner and Levinson, 2014; Boisjoly and El-Geneidy, 2016).

The diversity of cyclists and their preferences have been attempted captured in several different taxonomies for cyclists and cycling infrastructure (Kent and Karner, 2019; Zuo and Wei, 2019; Bhuyan et al., 2019). One of these is the Level of Traffic Stress (LTS) originally proposed by the Mineta Transportation Institute (Kent and Karner, 2019). LTS is method for classifying streets depending on their level of comfort for cyclists in order to quantify how cyclists experience a network and identify low stress connections (ibid.). With LTS, each street segment is classified on a scale from 1 to 4, where 1 corresponds to the lowest level of stress and 4 is the highest (ibid.). The classification levels correspond to a categorisation of cyclists into four groups, where 1 is those most vulnerable to traffic stress (e.g. children) and 4 are the most confident and experienced cyclists (Zuo and Wei, 2019). The exact methodology for how to classify street segments according to LTS varies from different studies and contexts (ibid.). Determinant factors are usually speed limit, street width, and the presence of bike lanes. The classification schemes have mostly been developed for a North American context, which usually differ significantly from the situation in Copenhagen when it comes to cycling. LTS will thus not be adopted directly in the evaluation of the network but will serve as an inspiration for how to evaluate different types of network segments as part of the network analysis. The exact classification of the Copenhagen network will be explained as part of the methodology (see p.35).

GIS, Geodata & Spatial Analysis

The following is a spatial analysis, which first and foremost entails that the research topic is approached from a spatial perspective, i.e. with a focus on the significance of location and spatial relationships such as distance and topology. In this case it also implies that GIS-software and other tools for working with spatial data will be an essential foundation for the analysis. A GIS-analysis makes sense for the research topic at hand, because the phenomenon being investigated – cycling infrastructure conceptualised as a network – at the most fundamental level is characterised by its spatial qualities, as for example proximity between elements, intersections of lanes, geographical density, etc. The analysis will thus first and foremost make use of spatial data. The term 'spatial data' covers a wide range of data types and formats, which however all share that the data must have a spatial component that allows you to determine the location of the object or area which the data describe. The exact data types and file formats used for this analysis will be described in detail in the next chapter (see p.29).

The fact that the analysis first and foremost will be based on quantitative, spatial data naturally has some implications for the type of knowledge the analysis can produce, and entails that e.g. more subjective and qualitative evaluations of the cycling network won't be an integral part of the analysis. The analysis is an examination of network characteristics and will be incorporating other factors than just metric distance, but it cannot give a full picture of how the network of cycling infrastructure is functioning or experienced by its users, even though this information to some extent do exist: the City of Copenhagen is on a regular basis conducting surveys among the city's cyclist, asking questions about their feelings of safety, satisfaction with bike lanes, experience with the cycling culture, etc. (City of Copenhagen, 2019b). This type of knowledge is an important element in a comprehensive understanding of the cycling infrastructure, but significantly lacks a more detailed spatial perspective, since the questions usually are not about specific elements of the network, but instead asks about the network as a whole. The conclusions which will be presented here should therefore not be interpreted as a complete examination of the network of cycling infrastructure, but as a new perspective and as a supplement to existing knowledge.

Network Analysis

The concept of a 'network' is used often and widely to refer to a quite diverse group of phenomena and within vastly different fields, from social network studies to electrical circuits (Brandes and Erlebach, 2005). Network theory is however also a field on its own, drawing on knowledge and methods from primarily mathematics and computer science. This approach to networks is also known as graph theory and focusses on the formal definitions and concepts used to model and study networks. Graph theory and the associated metrics have been chosen because they allow for describing the entire network in a quantifiable and systematic manner, and thus fits the purpose of developing a more systematic and comprehensive method for describing networks of cycling infrastructure.

In the following chapter I will introduce key concepts from graph theory used for describing and analysing networks, followed by an outline of the specific field of network analysis concerned with spatial networks. The chapter ends with some specific considerations for using network analysis and graph theory for an analysis of specifically networks of cycling infrastructure.

Spatial Network Data & Graph Theory

What is a Network?

Before commencing a network analysis, the first question to answer is of course what the concept of a 'network' covers. At the most general level, a network is "an object composed of elements and interactions or connections between these elements" (Brandes and Erlebach, 2005, p.7). Within mathematics, this concept is described as a *graph*. Network and graph will therefore be used synonymously in the following.
A graph consists of two types of elements: *vertices* (also referred to as nodes) and *edges* (sometimes known as links), where each edge connect a pair of vertices in the graph (see Figure 6). For a graph G = (V, E) the set of vertices is denoted V(G) and the set of edges E(G). The number of vertices in G is denoted *v* while *e* represents the number of edges (ibid.). If a vertex only is connected to one edge it is a *dangling* vertex (Tresidder, 2005).

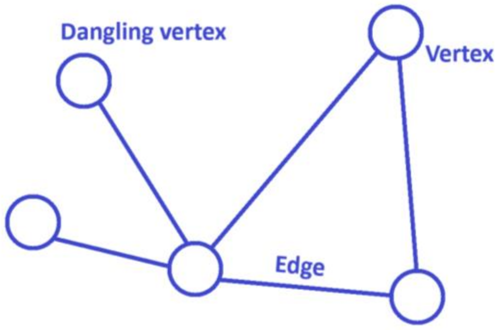

Figure 6. The structure of a graph

Two vertices joined by an edge are considered *adjacent* or *neighbouring* vertices (Brandes and Erlebach, 2005). For this specific application of graph theory, the edges describe actual segments of cycling infrastructure, whereas the vertices describe the segments' intersection or start and end points. This is the so-called *primal approach*, and arguably more intuitive way of representing a transportation network. A network can also be modelled according to the *dual approach*, in which street segments are considered vertices and an edge is created between two vertices, if the street segments represented by the vertices are connected (Lin, 2017). The primal approach to network representation corresponds to the *L-space* for networks, in which only adjacent vertices are considered having a direct connection. This is different than for example *P-space*, in which connections are assumed between all vertices if there is a path between them (see Figure 7) (Lin, 2017). The primal approach and the L-space modelling have been chosen for this analysis because the edges in this case are the fundamental structure of the network, not just abstract links between vertices, as it is in for example social networks, where the vertices are the initial elements.

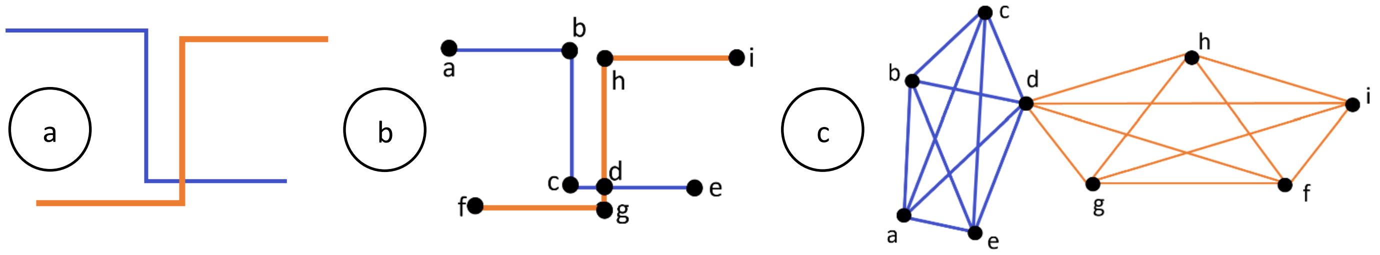

Figure 7. A network of bike lanes (a) modelled in L-space (b) and P-space (c) (based on Lin, 2017)

A network can either be directed or undirected (see Figure 8), describing respectively whether an edge connecting two vertices creates a connection in both directions or if it only allows movement in one direction. If a network is directed, an edge connecting two vertices *v* and *u* will be an outgoing edge for *v* and an incoming edge from *u*.

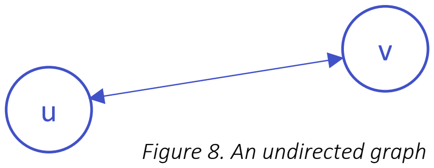

Figure 8. An undirected graph

The network of cycling infrastructure in Copenhagen is to some extent directed. It can however be complicated to determine whether a given segment in the network is directed or not: there are both two-way streets, which however only have a bike lane in one side; one way streets which might or might not have a bike lane in the same direction as the car traffic; mixed traffic streets which only are one-way for cars, and one-way streets with a bike lane that runs in the opposite direction of car traffic, thus allowing cyclists to cycle on the street in one direction and on the bike lane in the other direction (see Figure 9). Moreover, it is in many places common practice for cyclists to bike on streets against the allowed direction (even though it is illegal), and the available data from the city do not describe whether a given segment represent bike lanes on both sides or only one side/direction. The following analysis will therefore assume that the network is undirected.

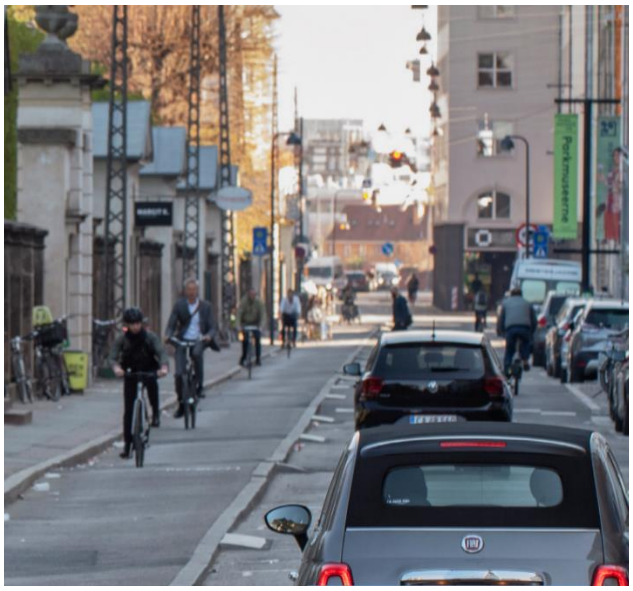

Figure 9. On Kronprinsessegade, the bike lane allows cyclists to move against the traffic on the otherwise one-directional street

Another aspect in which the graph representing the network is simplified compared to the physical network is the representation of parallel edges. The physical network of cycling infrastructure is actually a *multigraph*, meaning that there might be parallel edges between the same vertices. This is for example the case for all the streets which have bike lanes on both sides of the street. As just mentioned, this is however not the case for the dataset, and the graph used to represent the network will therefore be a *simple* graph, i.e. with no parallel edges (Brandes and Erlebach, 2005). It is moreover loop-free since there are no edges for which the start and end vertices are identical. A similar simplification is done considering the planarity of the network. A planar graph is a graph where no edges intersect without having a vertex at the intersection (Dill, 2004). In other words, there are no cases where an edge is intersecting another edge without actually being connected with that edge. For the cycling network in Copenhagen, this is the case for almost all locations, except for a few cycling bridges which requires a small detour in order to connect to the street and bike lanes under the bridge (see Figure 10). For the sake of this analysis the network will however be considered a planar network.

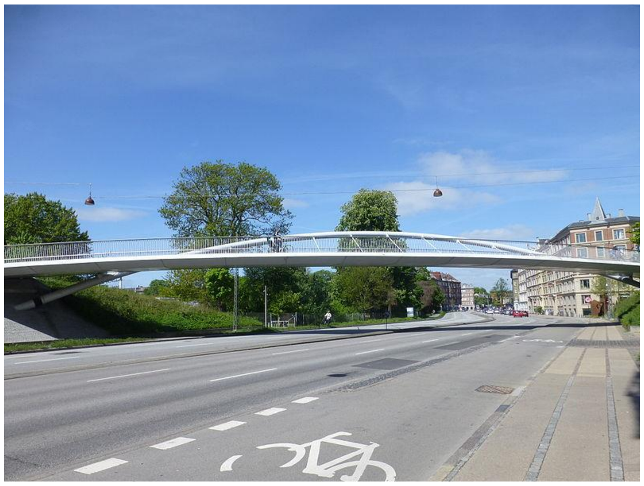

Figure 10. The cycle bridge Åbuen requires a small detour to connect to the bike lane below. Image from Jørgensen, 2015.

A graph is considered *connected* if there is a path from every vertex to every other vertex, i.e. there is no location in the network which cannot be reached from all other locations (disregarding the length and complexity of the path) (ibid.). Otherwise the network is disconnected (see Figure 11). Whether the network of cycling infrastructure is considered connected depends on how the physical network is converted into the abstraction of a graph. If smaller gaps between bike lanes are disregarded, if it is still possible to bike between the bike lanes, the network is connected[3]. A path is in this context a sequence of vertices and edges which together form a connection between vertex u and vertex v, given that $v \neq u$.

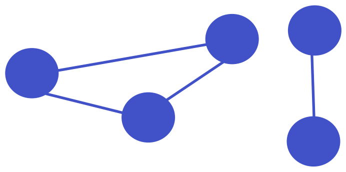

Figure 11. A disconnected graph

[3] Ignoring a few completely isolated segments which will not be included in the analysis. See p.31 for more about how the data have been cleaned and condensed.

To determine e.g. the shortest path (see p.18 for more on shortest paths), the metric distance between vertices can be used, given that each vertex has been assigned a location that enables distance computation. Shortest paths and distances can however also be based on vertex or edge *weights*, which describes a given property of the segment. The weights for a graph can represent e.g. distance, travel time, or other forms of cost, or it can be a measure of e.g. the strength or capacity of a connection (ibid.). The network will in this case be assigned weights representing the costs for the various segments (see p.35 for a further description).
Finally, the Copenhagen network of cycling infrastructure is considered a *complex* network. Complex networks are networks which are neither completely random nor completely regular[4], which is the case of most real-life networks (Lin, 2017).

Storing Network Elements

Common ways to store network elements and describe the connection between them are respectively an *incidence matrix* and an *adjacency matrix*. An incidence matrix B for an undirected graph is a matrix with v rows and e columns (with v being the number of vertices and e being the number of edges). If an entry $B_{i,j} = 1$ this means that the vertex i and the edge j are connected, if $B_{i,j} = 0$, they are unconnected. An incidence matrix can for example be used to quickly get an overview of the number of edges connected to each vertex (which is equal to the sum of the row corresponding to that vertex) (see Table 2 for an example). An adjacency matrix A is a matrix with v rows and v columns. For an undirected graph, if an entry $A_{u,v} = 1$ the vertices u and v are adjacent, i.e. there is an edge between them. If $A_{u,v} = 0$ the vertices are non-adjacent (Brandes and Erlebach, 2005). The adjacency matrix can be modified to store edge weights, so that $A_{u,v}$ stores the weight of the edge (Wagner, 2003). In that case, the lack of a valid number signifies that there is no edge between the vertices (Harrie, 2014b) (see Table 3 for an example).

[4] A regular graph is a graph in which all vertices have the same number of neighbours (Brandes and Erlebach, 2005)

Edges / Vertices	1	2	4	5	8	9	10	11	14	15	16	18	19	20	21	22	…
1	1																
2																	
3		1															
4		1															
7			1														
8			1														
9				1													
10																	
11																	
15					1												
16					1												
17						1											
18						1											
19							1										
20							1										
21								1									
…																	

Table 2. Segment of the incidence matrix for the graph.

Vertices	1	2	3	4	7	8	9	10	11	15	16	17	18	19	20	21	22	…
1																		
2																		
3				2														
4			2															
7						4												
8					4													
9																		
10																		
11																		
15											8							
16										8								
17													9					
18												9						
19															10			
20														10				
…																		

Table 3. Segment of the adjacency matrix for the graph, in this case storing the IDs of edges between vertices

Analysing & Evaluating Networks

With the formal conceptualisation of a network taken care of, I will introduce some of the core concepts and methods for analysing, evaluating, and comparing networks. As mentioned above, network analysis is a broad field, and the abstract concept of a graph is used to model many different types of phenomena. Not all concepts from graph theory will be meaningful or applicable for all types of networks or approaches, and it is therefore important to choose your analytical concepts carefully. In the section below I will limit the focus to the selected concepts for examining a network of cycling infrastructure.

When analysing a network, you are generally interested in identifying either some general, underlying characteristics for the whole network, or to identify specific segments or elements which stand out from the rest of the network. Measurements describing the entirety of the graph are referred to as *global* values, whereas measurements for specific segments are *local* values (Brandes and Erlebach, 2005). Measurements can furthermore be calculated either as single values (e.g. the average path length) or as distributions (e.g. the range of path lengths) (ibid.). Many of the measurements are based on the computation of shortest paths between vertex pairs. The concept of shortest paths and how they can be calculated will therefore be the first topic of this section.

Shortest Paths

As mentioned above, a common operation in network analysis is to find a path between two vertices, a path which usually is equal to the shortest possible path. The shortest path is normally defined as the one with the lowest costs, based on the weights assigned to the edges or vertices (Aby-Ryash and Tamimi, 2015). The shortest path is hence a relative intuitive concept, although the actual computation of shortest paths can be less straightforward.

The computation of shortest paths can be divided into two different approaches: the *single-source shortest paths problem* (SSSP) and the *all-pairs shortest paths problem* (APSP) (Brandes and Erlebach, 2005). For the single-source shortest paths problem, the shortest paths are calculated from a single source vertex s to all other vertices in the graph. For the all-pairs shortest paths problem, the problem is to find the shortest paths for all possible pairs of vertices in the graph (ibid.). The analysis performed here requires the shortest paths between all vertices and will therefore use the APSP approach. Several methods for solving the APSP exists. In this case, the Floyd-Warshall algorithm will be applied.

The Floyd-Warshall Algorithm

The Floyd-Warshall Algorithm for solving the APSP problem works by comparing all potential paths between all vertex pairs in the graph. The algorithm makes use of an adjacency matrix and furthermore requires a distance matrix D with the dimensions n*n which stores the distance between all vertex pairs. For the algorithm to produce the correct result, the graph must be connected and have no cycles of negative weights[5].

- First, all entries in the distance matrix are set to infinity
- For each vertex v, the entry in the distance matrix corresponding to the distance between the vertex v and itself, $D_{v,v}$ is set to 0
- For each edge (v,u) (i.e. each edge connected to vertex v) the distance matrix $D_{v,u}$ is set to the weight of edge (v,u)
- The next step is to check if there exists a vertex triple (v,w,u) for which the distance is shorter than for (v,u), in which case the distance matrix for $D_{v,u}$ is updated with the shortest distance value
- The check is performed in a triple loop over all vertices until the shortest distance between all vertex pairs have been found (Brandes and Erlebach, 2005).

For an example of how the Floyd-Warshall algorithm can be implemented, see Appendix XIII.

Connectivity

The first type of measurement to be introduced below are different ways of determining the *connectivity* of a graph. Connectivity could be said to be the main objective behind a transportation network (Dill, 2004). The computation of a networks' connectivity is usually done on a global scale (i.e. one measurement for the whole network), as an indication for how well connected the network is in general. This is in contrast to the

[5] A cycle, or circuit, is "a finite, closed path starting and ending at a single node" (Dill, 2004, p.6). A cycle of negative weight is a cycle which weights added together results in a negative number.

centrality indices introduced later (see p.22), which can be used as quantifications of how well the individual network elements are connected to the rest of the network.

Alpha, Beta & Gamma Indices

Some of the common measurements for network connectivity are the *alpha*, *beta,* and *gamma* indices. The alpha index α measures the actual number of cycles in a network, compared to the theoretical maximum number of cycles for that network. The alpha index ranges from 0 to 1. Simple networks and graphs with a tree structure will have an alpha index of 0, while higher alpha-values represent a more connected network (Rodrigue et al., 2009). The alpha index can be computed based on the number of edges (e) and the number of vertices (v) in a graph (Dill, 2004):

$$\alpha = \frac{e - v + 1}{2 * v - 5}$$

Eq. 1. Alpha Index

The beta index β describes the ratio between the number of edges to the number of vertices (ibid.). Simple networks have a beta index smaller than 1 while more complex networks have a beta index larger than 1 (Rodrigue et al., 2009). The beta index is, within network analyses of street networks, also known as the Link-Vertex Ratio (Dill, 2004). A perfect street grid has a beta index of 2.5, but the suggested value for street networks is usually around 1.4, which is neither too simple nor to gridded for an urban street network (ibid). The index can be computed as (Rodrigue et al., 2009):

$$\beta = \frac{e}{v}$$

Eq. 2. Beta Index

Finally, the gamma index γ is a measure based on the ratio between the number of actual, observed edges in a graph compared to the theoretical maximum number of edges (Rodrigue et al., 2009). Just like the alpha index, the gamma index range between 0 and 1, with a value of 1 representing a completely connected network, also known as strong connected, with direct links between all vertices (ibid.; Brandes and Erlebach, 2005). The gamma index can be computed as (Rodrigue et al., 2009):

$$\gamma = \frac{e}{3 * (v - 2)}$$

Eq. 3. Gamma Index

The alpha, beta, and gamma indices can be useful for an initial description of the connectedness of a graph, but two very different graphs might still have the same values for these three indices, and the they do thus not necessarily reveal a graph's more unique structural properties (ibid.). To achieve that, additional measurements are needed.

Density

A suggested measurement for the connectivity of a transportation network is the network density and clustering. Network density can be conceptualized in several different ways:
Intersection density is the number of intersections between network segments per area unit (e.g. square kilometre). The higher the number the higher the connectivity of the network is assumed to be (Dill, 2004).
Street density is correlated to intersection density but is measured as the network length per area unit, with higher values indicating a shorter distance between edges, which might suggest a higher connectivity (ibid.).
Connected node ratio (CNR) is calculated as the number of intersections (e.g. between bike lanes) divided by the number of intersections plus the number of dangling vertices. The CNR ranges between 0 and 1 with a higher CNR-value suggesting few dangling vertices, which usually indicates a higher connectivity (ibid.).

Clustering

The amount of clustering in a network can be quantified through the *clustering coefficient*. The clustering coefficient can both be computed as a local value for each vertex, or as a global value for the entire network (Lin, 2017). The clustering coefficient is actually measuring the edge density of a vertex's neighbours by testing for the existence of triangles in the network (Shanmukhappa et al., 2018) (see Figure 12).

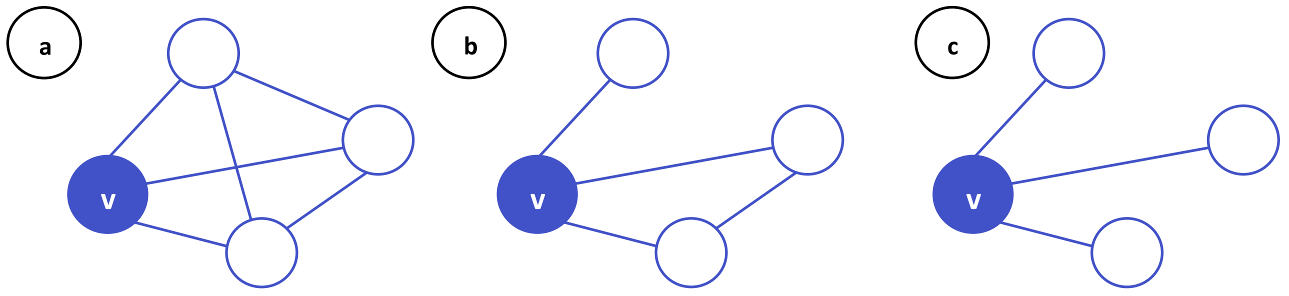

Figure 12. Various levels of clustering for the vertex v: Maximum clustering (a), medium clustering (b), no clustering (c) (based on Bisht, s.d.)

The clustering coefficient C_i for a vertex v_i can be computed based on the degree of the vertex d_i (see p.23) and the number of edges between the adjacent vertices (m_i) (Lin, 2017):

$$C_i(v_i) = \sum \frac{2 * m_i}{d_i * (d_i - 1)}$$ *Eq. 4. Local Clustering Coefficient*

There are several ways of determining the global clustering coefficient, but one method is simply to find the average of all local clustering coefficients for all vertices (Shanmukhappa et al., 2018):

$$\bar{C} = \frac{1}{v} \sum_{i=1}^{v} C_i$$ *Eq. 5. Global Clustering Coefficient*

Average Path Length

A network's *average path length* is another global measure that can reveal some of the network's underlying characteristics (Lin, 2017). The average path length is defined as below, with v denoting the number of vertices and d_{st} referencing the distance between two vertices s and t (Lin and Ban, 2017):

$$L_{ave} = \frac{1}{v(v-1)} \sum_{s,t \in V, s \neq t} d_{st}$$ *Eq. 6. Average Path Length*

Network Efficiency

A final method for evaluating the global connectivity of the network is *network efficiency*. The efficiency of a network naturally depends on the objective with the network and which function it serves but can be generalised as the effectiveness of movement through a network (Lin, 2017). The network efficiency can be computed as (ibid.):

$$E = \frac{1}{v(v-1)} \sum_{s,t \in V, s \neq t} \frac{1}{d_{st}}$$ *Eq. 7. Network Efficiency*

The computation of average path length and network efficiency look remarkably similar. While average path length is based on the distance between vertices in the chosen unit and returns an average in the same unit, network efficiency is based on the reciprocal distance between vertices and thus returns a result which, although influenced by the choice of unit, is not expressed in that unit.

Centrality

The computation of centrality is used to quantify the possibly "intuitive understanding" that in a network, some vertices or edges are more important for the network connectivity than others (Brandes and Erlebach, 2005, p.16). This variance of importance is usually computed as a so-called *centrality index* which assigns a level of centrality to every vertex or edge (ibid.). Although centrality indices are essential within network analysis, there is no single or clear-cut definition of what a centrality index is. Instead, a number of interpretations and methods for how to determine the relative importance of network segments exists, and the choice of method must depend on the real-life function of the network. One interpretation of centrality might be to identify the edges with the most activity/flow, or the vertices with the highest number of connections, while another approach might understand centrality as the vertices and edges which, if removed, would disconnect the network (ibid.). For all centrality indices it is however a requirement that they only depend on the structure of the graph and, for most centrality indices, that the graph is connected (ibid.).

Vertex Degree

A common and straightforward centrality index is the measurement of the *degree* of a vertex. The degree $C_D(v)$ for a vertex v in an undirected graph is simply the number of edges connected to the vertex v (ibid.). The degree centrality is a local measurement, since it only measures a vertex's immediate neighbours, and it does not tell you anything about how central or well-connected the vertex's neighbours are (ibid.; Shanmukhappa et al., 2018). The degree measurement is applicable for the analysis of transportation networks, even though there are natural limitations to the degree variation for specifically street networks: it is rare to encounter intersections with more than 3 intersecting roads, which results in a maximum degree of 6 for all vertices in the network (Lin, 2017).

Stress Centrality

The *stress centrality* $C_s(e)$ gives an indication of the amount of strain or 'work' done by each vertex or edge (Brandes and Erlebach, 2005). The index is based on the number of shortest paths that goes through a network element and is thus based on the assumption that flows always follow the shortest path. The stress centrality for edges is defined as below, where $\sigma_{st}(e)$ refers to the number of shortest paths containing the edge e, and s and t are the start and end vertices (ibid.):

$$C_s(e) = \sum_{s \in V} \sum_{t \in V} \sigma_{st}(e)$$ *Eq. 8. Stress Centrality*

Cyclists do not always choose the shortest path but are generally quite sensitive to distance and tend to value shorter distances over most other factors (Schoner and Levinson, 2014). The stress centrality measures the amount of flow through an edge in an "all-to-all scenario" (Brandes and Erlebach, 2005, p.29), i.e. a hypothetical situation where there for each vertex is exactly one person cycling to all other vertices. This is of course not an accurate picture of the actual flow of cyclists in a city. The stress centrality can however be a good indication of which edges are most important for the overall network connectivity, and moreover reveal how many shortest paths would be affected if an edge were changed or removed. Stress centrality can likewise be used as an indication of the resilience of the network: an even distribution of stress centrality indicates a resilient system where no single edge carries too much stress, whereas a high variety in stress centrality and a few elements with very high values indicates that the network is too reliant on a small number of critical vertices and edges (Lin and Ban, 2017).

Closeness Centrality

The *closeness centrality* is based on the average distance from a vertex to all other vertices in the network and can thus be a measure of the reachability of that vertex to or from other places in the network (Brandes and Erlebach, 2005). The distribution of closeness centrality can for example be used to illustrate various levels of accessibility in a network, with higher variations in closeness in less gridded street networks (Lin, 2017). The closeness centrality is computed as follows, with d_{st} denoting the length of the shortest path between vertex s and vertex t and v denoting the number of vertices in the network (ibid.):

$$C_c(s) = \frac{v-1}{\sum d_{st}}$$ *Eq. 9. Closeness Centrality*

The closeness centrality value actually measures the inverse distance to all other vertices. Vertices with a high closeness centrality value thus have the shortest distance to all other vertices (Neo4J, s.d.).

Straightness Centrality

The *straightness centrality* introduces the spatial distance in the calculation of centrality, rather than just shortest path, and is based on a comparison between the length of the shortest paths in the network compared to the Euclidian distance measured as a straight line (Lin, 2017). The straightness centrality can therefore be interpreted as a measurement of the efficiency of the network, in the sense that it measures how much longer the path in the network is, compared to a path that does not have to follow the structure of the network. This centrality index is thus only applicable for networks where it makes sense to distinguish between different types of distance, as it is the case for a cycling network. The straightness centrality for a vertex is defined as below, with d_{st}^{Eu} denoting the Euclidian distance between vertex s and vertex t:

$$C_s(s) = \frac{1}{v-1} \sum_{s,t \in V, s \neq t} \frac{d_{st}^{Eu}}{d_{st}}$$

Eq. 10. Straightness Centrality

Network Statistics

The measurements and indices introduced above are often summarised in the form of *network statistics*, which have the purpose of describing "essential properties" of the network, without having to present the detailed structure of the graph (Brandes and Erlebach, 2005, p.293). This is particularly relevant for local values, which might be difficult to interpret, especially for large and complex networks. The transformation of local values to global values or distributions can e.g. be done by summation, averaging, or finding minimum or maximum values (Brandes and Erlebach, 2005). It might furthermore be useful to normalise both local and global values, or in other words, to transform them to a relative value compared to the overall distribution of values for that specific measurement. A common normalisation method for most connectivity and centrality indices is to divide all local values with the maximum value for that index, resulting in all values being transformed to a value on a scale from 0 to 1 (ibid.). This normalisation makes it easier to assess whether a specific vertex for example has a relatively low or high degree compared to the distribution of values across the network. It should however be used carefully, if the objective is to compare values between different networks, since a normalised value no longer tells you anything about the actual value range. For comparison, non-normalised values are usually preferable. Comparisons across graphs should still be done carefully, since e.g. the maximum centrality value that a vertex can have is depending on the graph size (ibid.).

Differing centrality values might thus be more of an indication that two graphs differ in their size than an indication of variations in structure.

Spatial Network Analysis

There is a long history of network analysis in geography, for example for the study of urban street structures, transportation networks, and route planning (Lin, 2017; Dill, 2004). The interest in network analysis has been fuelled by the development of GIS and increasing computing powers, which allows for more advanced analysis of complex networks (Xie and Levinson, 2007). The theoretical framework presented above is applicable for spatial applications, but there are some specific considerations for spatial networks. This is for example, as previously described, the modelling of the network, particularly what the edges and vertices are made to represent. The fact that the graph in this case represents a physical structure moreover requires some extra considerations, such as how the physical landscape might place constraints on the network structure and explain some of the observed topological relationships. Spatial networks additionally exist in three dimensions, which particularly should be considered in places where elevation is a factor. Copenhagen is located in a very flat area and elevation will therefore not be a variable in this network analysis, but elevation can be an important factor for networks of cycling infrastructure in other locations.

Apart from elevation, a network analysis of cycling infrastructure moreover has some challenges which are less prevalent for regular street networks. Cycling infrastructure is for example more prone to suffer from missing links and unconnected networks and the datasets describing the networks often have a lower data quality and ambiguous classification of different types of cycling facilities (Schoner and Levinson, 2014; Lucas-Smith, 2019). It is therefore important to clarify the assumptions and limitations for the analysis since reliable data about the exact extent of the physical infrastructure might not exist.

4. Method

In the following chapter I will begin with an introduction to the different tools used during the analysis, followed by a description of the collection and subsequent cleaning of data. Subsequently, the central steps in the method used for the analysis will be presented.

Tools

Throughout the data collection, data cleaning, and analysis, I have made use of a wide range of tools and technologies. The following is thus only a presentation of the tools which have been most decisive for the design of the analysis.

A Spatial Database with PostgreSQL

The foundation for the storage and analysis of all data used in the analysis is a database in the open-source database software system PostgreSQL, also known as Postgres. Postgres is an *object-relational database management system* (ORDBMS) (Mansourian and Harrie, 2013), meaning that the system combines characteristics from the *relational data model* and the *object-based data model.* From the relational model, Postgres has adopted the idea that data are stored in 'relations' or tables which can be linked together with the use of unique 'keys' or ids, allowing the user to combine for example a table with the population data for a city's neighbourhoods with another table describing the neighbourhood's infrastructure. Furthermore, Postgres uses SQL (Structured Query Language) to communicate with the database, just like most relational databases (ibid.). From the object-oriented data model, Postgres has adapted the possibility to implement abstract data types (ADT), as for example spatial data. The implementation of ADTs in the database makes it possible to combine both geometric and attribute data in the same table (unlike the relational model where geometries and attributes are stored separately) (ibid.). This is one of the main benefits for using an ORDBMS for spatial applications, since the storage of spatial data in the table enables you to perform geometric and topological queries using the same methods – in this case SQL – as with the attribute data (ibid.).

To allow Postgres to support and query geographic objects, the extension *PostGIS* is required. With PostGIS, the database can store different feature types (e.g. points, linestrings, polygons etc.), support spatial and topological operations (such as area, distance, intersect, buffering), and define and change the data's spatial reference system.

A Postgres database with the PostGIS extension was the primary tool for the storage and pre-processing of the data, as well as for some of the initial queries used in the analysis (see below for details). The computation of shortest paths, which is the foundation of the majority of the centrality indices, was however done with the programming language Python, as outlined below.

Using Python for Spatial Analysis

Python is a general purpose open-source programming language often used in data analysis. Python is also widely used within spatial analysis due to its potential for automation and more customized solutions, compared to desktop software. Many desktop GIS therefore come with a build-in Python module, which allows you to run Python scripts as part of a spatial analysis, just as there exist numerous packages that supports using Python for spatial data.

In this analysis, Python has been used for some pre-processing of data, but is predominantly applied in the computation of shortest paths. The computation of shortest paths could potentially have been conducted in Postgres with the PostGIS extension pgRouting, which contains an implementation of the Floyd-Warshall algorithm for all pair shortest paths. This solution does however not allow you to inspect the actual shortest paths, only the distances or costs between all vertex pairs (pgRouting, 2017) and is therefore not adequate for the requirements for this analysis.

The solution to the APSP problem and the computation of the various connectivity and centrality indices are primarily based on the Python packages Pandas and NetworkX[6]. Pandas is a package based on NumPy[7] and is particularly useful for working with tabular data structure. I have in this context used it for e.g. the pre-processing of demographic and socio-economic data and for creating the adjacency matrices. NetworkX is a Python package developed specifically for modelling and studying graphs and has been utilised for the

[6] A Python package is a collection of different files/modules containing code defining functions, classes, variables, etc.
[7] NumPy is Python package developed for scientific computing based on an array data structure, as used in for example an adjacency matrix.

computation of shortest paths, with the use of the build-in function for the Floyd-Warshall algorithm and the package's predefined classes for graphs and their components of edges and vertices.
Additionally, a number of other packages and modules used for e.g. plotting and connecting to the database have been applied.

There are several reasons for using Python with NetworkX rather than e.g. a desktop GIS. First of all, the most common desktop GIS do not support advanced network analysis, like the computation of APSP or centrality indices – at least not without requiring additional licenses, which often are not available in the urban planning settings in which such an analysis frequently will take place. The goal was therefore to make use of open source tools, which do not have any immediate barriers from acquisition costs. An analysis based on several, lower-level tools and steps, rather than e.g. a pre-programmed function in a desktop software, moreover makes it easier to examine the interplay between data quality and structure and the final results, as well as allowing for an exploration of how a network analysis can be adapted to the specific research topic. Desktop GIS – in this case QGIS – has however also been used at various steps in the analysis and is also used for data visualisation.

Data Preparation

The following sections will firstly describe which datasets I have used to answer the research questions, what the data describe, and how they have been obtained. Subsequently, the most important steps taken to pre-process and clean the data, before the actual analysis, will be explained.

Data Collection

All data used for the analysis are made available by the City of Copenhagen and Frederiksberg Municipality, primarily as open data through the data portal for public data in Denmark, opendata.dk. The majority of the datasets are geodata and available as Shapefiles[8], but demographic data are available as tabular data from the City of Copenhagen's public database with city statistics, statistikbanken.kk.dk.

[8] A Shapefile is a nontopological data format developed by the GIS-company ESRI. Shapefiles are used for storing location and attribute information about spatial features and can contain either points, lines, or polygons (ESRI, 2016). The file format has become an industry standard for working with spatial data.

Bike lanes and routes in Copenhagen

Two different datasets describing bike lines have been obtained from respectively the City of Copenhagen and Frederiksberg Municipality. The most important information in the datasets are:

- Type of segment (e.g. bike lane, green cycling route, bike friendly street etc.)
- Whether the segment already has been established or only is planned for later construction
- Geographic extent of each segment

Roads

For the street network, several different datasets were used. The most important information in the datasets for the street network are:

- Type of street (e.g. local street, regional street, pedestrian shopping street, etc.)
- Whether the street is public, municipal, or state owned
- Speed limit, where available
- The geographic extent of the road surface (i.e. polygon data)
- Traffic counts (measuring the volume of traffic on a street segment)

Socioeconomic Data

To contextualise the findings of the analysis of network and centrality, several different datasets were used, describing for each neighbourhood:

- Neighbourhood boundaries
- Population density
- Average income
- Level of education
- Age distribution

Apart from the data described above, various datasets and base maps will be used for visualisation purposes.

Data Cleaning & Pre-Processing

Demographic Data

The demographic data describing the age distribution for each neighbourhood are made available as CSV-files containing a level of detail which are not necessary nor appropriate for the research topic. The first step in the pre-processing of data was thus a classification of the population data into age groups. The groups used here follows the classification of the population into children, young people, adults, and elderly normally used within traffic planning and cycling research (see e.g. Herby and Friis, 2014):

- 0 – 17 years
- 18 – 24 years
- 25 – 64 years
- 65 – 99 years

Cycling Infrastructure

The Shapefiles describing the cycling infrastructure in Copenhagen had a great number of missing attribute values, invalid geometries, duplicate lines and vertices, gaps between segments which actually are connected in the physical infrastructure, dangling ends at intersections etc. This would be an issue for most spatial analysis, but is possibly particularly problematic for a network analysis, in which it is of great importance whether two segments are connected or not, and where duplicate and overlapping edge segments will give a skewed value for the length of the network. The first part of the data cleaning was thus an extensive and mostly manual entering of missing values, removal of duplicate features, merging of mistakenly divided segments, etc. The pre-processing of the data did moreover also involve moving from a dataset with lines which ideally should represent the exact extent of the physical infrastructure to one which represents the more abstract idea of a graph. This for example means that while the original data did not connect two bike lanes separated by an intersection, although they in practice function as one segment, those segments should be connected in the final graph. The removal of gaps between segments can to some extent be automated, but in many instances require a visual inspection: a 10 meter gap between segments might for example be due to an intersection, meaning that the segments actually should be considered connected, or it might be a pedestrian-only street or canal dividing the bike lanes, in which case the segments should not be connected in the final graph.

The automated data editing steps can be summarised as follows:

- If a line segment *a* ends less than 3 meters from another line *b*, line *a* is extended so that it connects to line *b* (see Figure 13).

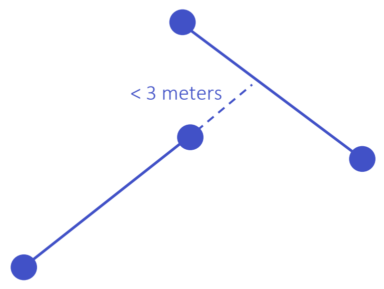

Figure 13. Unconnected edges are connected

- If two lines intersect and produces dangling ends, which are 3 meters or less, the ends are removed (see Figure 14).

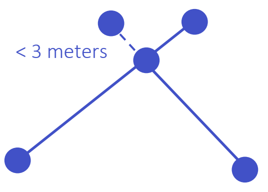

Figure 14. Dangling ends are removed

- The final dataset only contains segments classified as bike lanes, Cycle Superhighways, or Green Cycle Routes.
- The network both includes existing and planned/future segments for bike lanes and Cycle Superhighways, while planned segments of Green Cycle Routes are not included, since these planned segments might not actually allow for cycling at all at the moment. For bike lanes and Cycle Superhighways it is assumed that biking is still possible, regardless of whether the infrastructure has been established or only decided on – and since the ambition is to identify potentially overlooked areas, the distinction between existing or planned is of less interest.
- The segments of bike lane are assigned an attribute describing whether they are part of a Green Cycle Route and/or a Cycle Superhighway, taken as a sign of respectively recreational value or as an indicator that the segment has been optimised for efficient and comfortable cycling.
- All segments are furthermore assigned an attribute describing whether they run through or along the water or a park, taken as a sign of recreational value.

- Unconnected segments were identified. Only four, shorter unconnected segments were identified and removed from the final graph.
- Finally, the dataset was converted to a topologically valid graph of edges and vertices (see below for an elaboration and Figure 15 for a visualisation of the edges and vertices).

The data editing performed in Postgres can be seen in Appendix III.

Topology

Spatial data loaded into a Postgres database extended with PostGIS is not, as a starting point, topological, meaning that the dataset contains the location and extent of the geographical features, but it does not explicitly store any information about whether the features intersect, touches, overlap, etc. Since the Shapefiles, which the data originally are delivered as, are not topological either, no data is lost by loading the files into a Postgres database. Topology is however essential when the line features are to be converted into a graph, since a graph is based around the idea of shared vertices (Mikiewicz et al., 2017). To enable explicit storage of topological relationships, the PostGIS topology extension can be applied. The PostGIS topology extension is based around the ISO standard for topological networks which constructs networks from vertices, edges and faces[9] (ibid.). Before any topological relationship can be determined, the dataset must be both simple and valid, i.e. only one vertex can be located at a given coordinate pair and features cannot be self-intersecting. When the geometries in the Postgres table are converted into the topological format, four new tables are created containing respectively the faces, vertices, and edges, and a table providing the relationship between the new edges and the original data, where the other attributes are stored. The actual topological relationships need to be queried from the edge table, which stores information about the start and end vertex of each edge, as well as the ID of up to two edges connected to the same vertices. Exactly how many edges a given edge is directly connected to or how many edges connect to a given vertex is not explicitly stored but can be queried from the tables.

[9] Faces are used to represent polygons as bounding boxes but will not be used in this analysis.

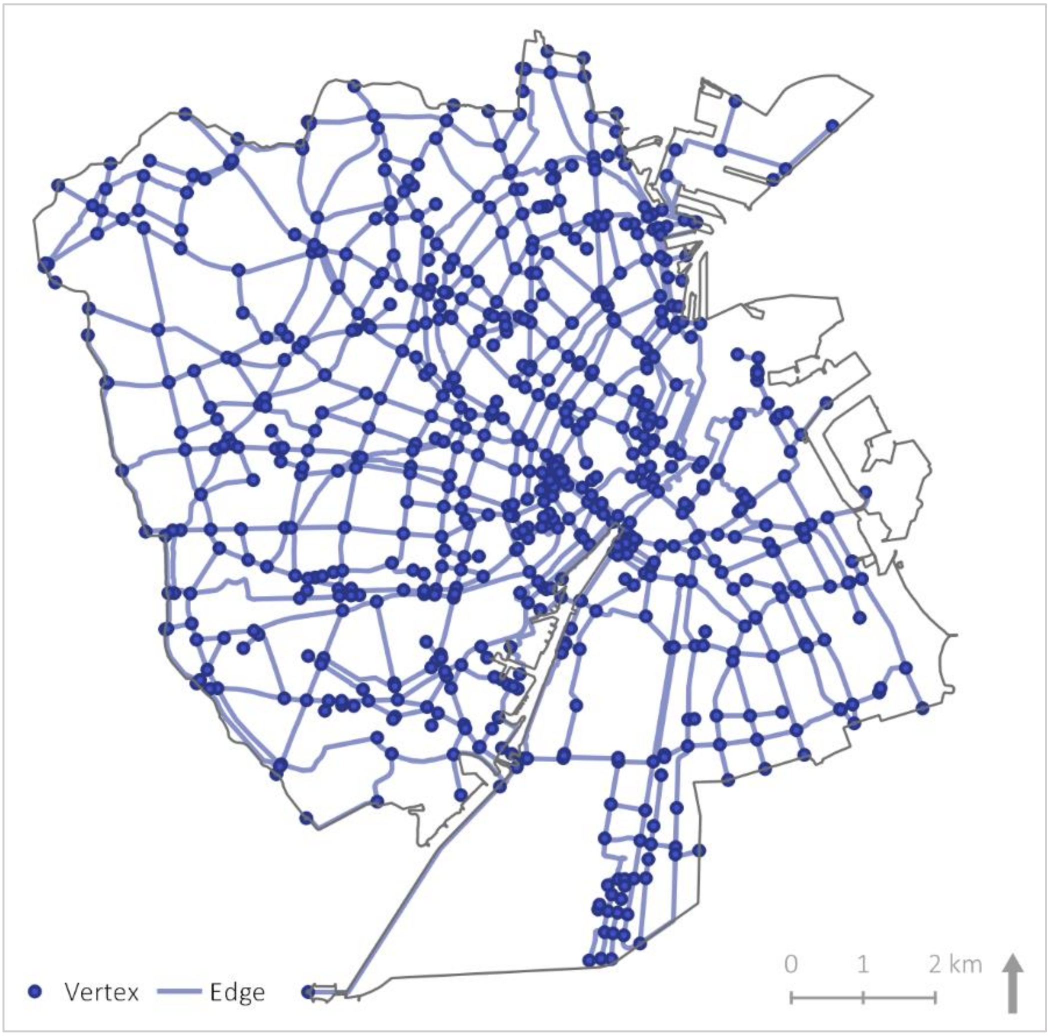

Figure 15. The network abstracted as vertices and edges

Data Analysis

The following section I will present firstly, how and according to which criteria the edges in the graph have been assigned weights, followed by an outline of how the various connectivity and centrality indices have been computed. Finally, I will describe the approach to contextualising and interpreting the results.

A Weighted Network

Assigning weights to a network of cycling infrastructure is not a straightforward task. Different attributes have various importance to different people, depending on both personal preferences – do you for example prefer a green and scenic route or to bike close to shops and urban life? – and well as the demographic group you belong to. Children and the elderly might e.g. be more sensitive towards high speed or a lack of separated bike lanes. Ideally, a weighted network analysis would be adapted according to the needs of a specific group, since the least cost path for e.g. commuters and school children cannot be expected to be the same.

Some approaches to analysing cycling networks, as for example the LTS method mentioned before, create a classification scheme for network segments, in which all segments within a class are considered to be equally suitable, and some classes might be completely omitted from the analysis. While this approach creates an intelligible and simple overview of the network it misses the nuances and cannot incorporate the fact that the experience of a given network segment is influenced by several and sometimes contradicting factors. A weighted network analysis on the other hand allows for differentiation based on a wider range of attributes summarised into a final weight for each segment.

For this analysis, two different approaches for assigning weights have been attempted to illustrate how varying preferences among cyclists influence how well-connected the network is in practice. One (W_{safety}) is designed with a general preference for segments with less traffic and slower speed, while the other ($W_{efficiency}$) gives more importance to efficiency, based on the Cycle Superhighways and the presence of physical bike lanes. As mentioned, cyclists are very sensitive to distance. It is thus important not to assign weights so high, that the paths of least cost actually are long detours in terms of physical distance. Additionally, a computation only based on segment lengths ($W_{distance}$) have been performed.

The edges in the graph have been assigned weights depending on the following factors:

- Whether the segment is part of a Green Cycle Route or a Cycle Superhighway
- Whether it is close to water or a park
- The type of street the segment is located on
- Whether the segment is a bike lane or not
- The speed limit for cars on the street where the segment is located
- The amount of car traffic and the number of trucks on the street
- The length of the segment

All segments have, as a starting point, been assigned a weight of 0. Weights are then either subtracted or added, depending of the edge's attributes. Since weights signify 'costs', positive attributes like green surroundings results in a smaller weight while negative attributes, like a high volume of traffic, increases the weight. Each edge weight is then multiplied with the edge length in 100 meters, to consider the fact that a long segment with a high weight should result in a higher cost than a shorter segment with the same attributes. Finally, all edge weights are increased with their length in meters, to take into account that distance is a determinant factor for cyclists.

The exact criteria and what weights they have resulted in can be seen in Appendix IV (see p.92).

Computing Connectivity & Centrality Indices

The connectivity and centrality indices selected for this analysis have all been computed based on the equations presented in chapter 3. I will not present every step in the methodology here, but only outline the general methodological approach. The various steps in the analysis and exact computations can be found in the appendices, which contain the Python code and SQL queries used for computing the indices.

The connectivity and centrality indices have been computed with the help of both SQL and Python scripts, which can be seamlessly combined using different Python packages, that allows connecting, querying, and uploading to a Postgres database from a Python environment. Most of the indices are based on the computation of shortest paths. In these instances, the shortest paths have been computed with the help of

NetworkX's built-in function for the Floyd-Warshall algorithm. This function has been combined and incorporated into a range of my own functions[10] for creating an adjacency matrix, computing the centrality and connectivity indices, retrieving path lengths, etc. (see Appendix VII – X).

When evaluating and assessing the network, the approach has been to, if possible, find local values for each individual network segment, but also to transform these local values to a global measure by finding e.g. the minimum, maximum, and average values, both for the whole network and for each neighbourhood. When appropriate (i.e. when the index is based on shortest path), indices have been computed for each of the three weighting schemes used for the shortest path computations.

To compare how the use of different weighting schemes affect reachability and flow through the network, maps showing how respectively $W_{efficiency}$ and W_{safety} differ from $W_{distance}$ at the neighbourhood level have been made. It is generally an important part of the methodology to make maps of the distribution and variation of the various indices: since it is a spatial network, a (visual) examination of geographical variations and any potential spatial autocorrelation is an important part of understanding how the network functions. Maps visualising network variations along with the area's physical and urban geography can furthermore help answer whether any of the variations in the network can be explained by the landscape or the physical shape of the network, such as differences between centre/periphery.

Contextualising a Network Analysis

The first and central part of the research question is focused on how to evaluate a network of cycling infrastructure based on graph theory. This focus point is however followed by an ensuing question on how such a network assessment can be used within urban planning, which in this thesis has been narrowed down to a question about how a transportation network might be evaluated in terms of equity in connectivity and accessibility. In order to demonstrate this, the Copenhagen network of cycling infrastructure and the variations in connectivity and centrality at the neighbourhood scale are compared and correlated to how the neighbourhoods differ when it comes to socio-economic characteristics and neighbourhood density. The purpose of this method is to determine firstly, whether the network fulfils the idea of horizontal equity within

[10] NetworkX does come with pre-defined function for several of the centrality indices, but they often do not return the results in the desired format and also offer less transparency into how the indices are calculated.

access to transportation networks, and secondly, to examine to what extent the network of cycling infrastructure lives up to, or conversely fails, the ideal of vertical equity – in other words whether some neighbourhoods appear to have a disproportionate high or low network quality, and whether this might be explained by some characteristic of that neighbourhood.

The choice of the neighbourhood as the analytical scale[11] means that the data on network variation is distributed on 11 datapoints, which imposes some limitations for the granularity and detail in the correlation between different parts of the network and the surrounding areas. The data do moreover not fulfil basic requirements for e.g. linear regression, such as for example constant variance. The neighbourhood variables and network variation have thus not been correlated through a rigorous statistical test but have instead been examined by plotting each pair of variables together, to visually examine whether any correlation might exist. This method does thus not allow for any firm claims about a, for Copenhagen, universal correlation between a neighbourhood variable and the quality of the network. It does however support an identification of neighbourhoods which stand out based on their network quality and can be used to indicate whether a poor network quality might be particularly problematic given the other characteristics of the neighbourhood.

[11] A choice which primarily has been determined by the fact that the neighbourhood is the most detailed scale for most publicly available socio-economic data in Copenhagen.

5. Results

In the following section I will present the result of the network analysis for each of the connectivity and centrality indices presented in chapter 3. The chapter will start out with the measures for overall connectivity, followed by the centrality indices. When relevant, the results for all 3 weighting schemes will be presented. The results will either be presented as the immediate results of the computation or as normalised values[12], depending on the unit of the results. For some indices, the non-normalised value represents a meaningful unit – e.g. the actual number of shortest paths passing through an edge – while it for others represent e.g. a fraction, in which case a normalised value might be just as informative.

Network Connectivity

The results of the connectivity indices presented in this section are so-called global values, i.e. they describe how well connected the graph is in general, in contrast to the local values for individual network elements. The connectivity indices are as a starting point computed on a city scale but, when relevant, the global values for each neighbourhood will be presented as well.

Comparing Edges to Vertices: Alpha, Beta & Gamma

In the graph representing the network of cycling infrastructure, the number of edges e = 997 and the number of vertices v = 704.

Alpha can thus be computed as:

$$\alpha = \frac{e - v + 1}{2 * v - 5} = \frac{997 - 704 + 1}{2 * 704 - 5} = 0.209$$

Eq. 11. Calculation of Alpha

This is a fairly low alpha value, which is a sign of a less connected network with few cycles and which is leaning towards a tree-like structure. This alpha-value is however a normal value for an urban street network and comparable with alpha-values for the regular street network (not bicycle network) for cities such as Stockholm and Toronto (Lin, 2017).

[12] Normalised values are based on the normalisation method described on p.26

Beta is calculated as:

$$\beta = \frac{e}{v} = \frac{997}{704} = 1.416$$

Eq. 12. Calculation of Beta

The Beta value for the network matches the suggested value for street networks (see p.20) and indicates a network which is neither completely simple nor perfectly gridded. The Beta value for the entire city does however cover substantial variations within the network, which for example can be detected at the neighbourhood scale, where e.g. Frederiksberg and Nørrebro have a more grid-like structure (with Beta values at respectively 1.9 and 1.8).

Gamma is calculated as:

$$\gamma = \frac{e}{3 * (v - 2)} = \frac{997}{3 * (704 - 2)} = 0.473$$

Eq. 13. Calculation of Gamma

Gamma values are always located in the range between 0 and 1, with 1 representing a completely connected network (strong connected). The Gamma value calculated here confirms the initial picture of a network structure somewhere in the middle between a simple and a strong connected network.

How Dense is the Network?

Intersections are defined as vertices with more than two connected segments, meaning that there is more than one entry for the vertex in the adjacency matrix. According to this definition, the number of intersections is 575 while there are 129 so-called dangling vertices which only are connected to one edge (see Figure 16).

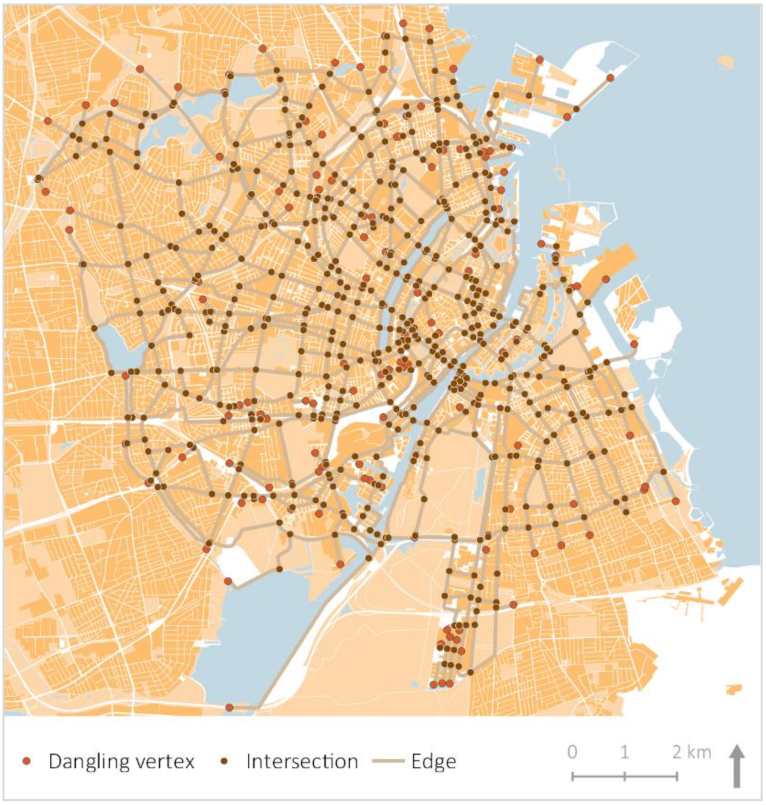

Figure 16. Dangling vertices and intersections

With an area of 102.17 km^2 for the entire city, the intersection density is 5.63 intersections per square kilometre. This does however cover some considerable variations between the different neighbourhoods:

Neighbourhood	Number of intersections	Intersection density (per km^2)	Normalised intersection density
Copenhagen	575	5.70	0.48
Amager Øst	38	3.87	0.32
Amager Vest	79	4.08	0.34
Bispebjerg	33	4.83	0.40
Brønshøj-Husum	26	2.97	0.25
Frederiksberg	44	5.05	0.42
Indre By	106	10.22	0.86
Nørrebro	49	11.93	1.00
Østerbro	60	6.10	0.51
Valby	41	4.44	0.37
Vanløse	31	4.63	0.39
Vesterbro-Kongens Enghave	67	7.92	0.66

Table 4. Intersection Density

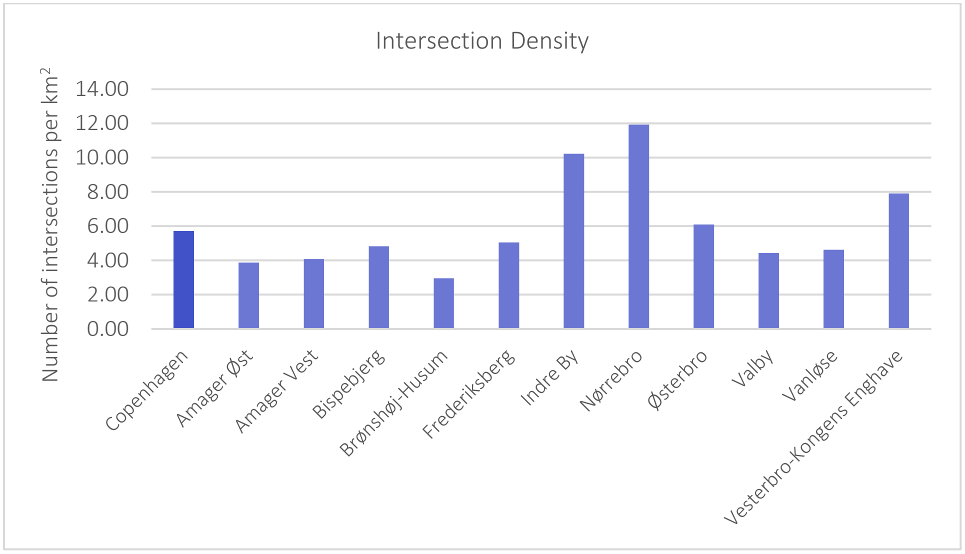

Figure 17. Neighbourhood intersection density

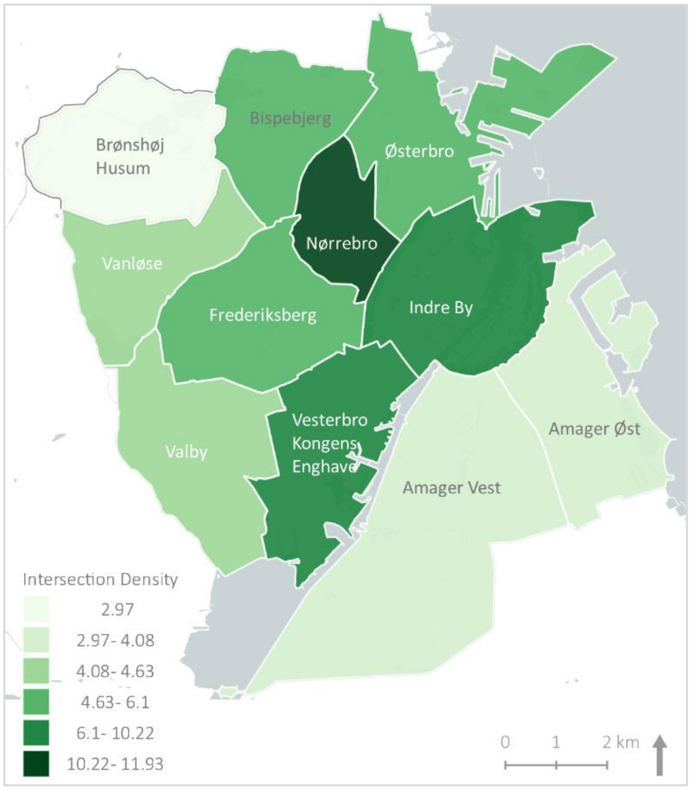

Figure 18. Intersection density as number of intersections per square kilometre (Natural Breaks Classification)

Part of the geographical variation in the intersection density is to be expected: since the network is connected and have no subgraphs, it is no surprise that there will be more dangling vertices in the peripheral neighbourhoods. An important consideration in the interpretation of intersection density compared to the number of dangling vertices is moreover that dangling ends located at the border of the study area might not be dangling at all, but actually connect to edges located outside the municipality. Despite these reservations, there are clearly considerable variants in intersection density across the city.

Street density is measured as the network length per area unit, which in this case is square kilometres. The entire network has a length of 373.68 km and a street density of 3.66 km per km^2. As with intersection density there are however also substantial variations between different parts of the city:

Neighbourhood	Length of network (km)	Street density (km/km^2)	Normalised street density
Copenhagen	373.68	3.66	0.62
Amager Øst	26.74	2.72	0.46
Amager Vest	55.22	2.85	0.48
Bispebjerg	26.17	3.83	0.65
Brønshøj-Husum	22.18	2.54	0.43
Frederiksberg	34.47	3.96	0.67
Indre By	48.21	4.64	0.79
Nørrebro	24.21	5.89	1.00
Østerbro	39.00	3.96	0.67
Valby	35.22	3.83	0.65
Vanløse	25.02	3.74	0.63
Vesterbro-Kongens Enghave	36.23	4.28	0.73

Table 5. Street Density

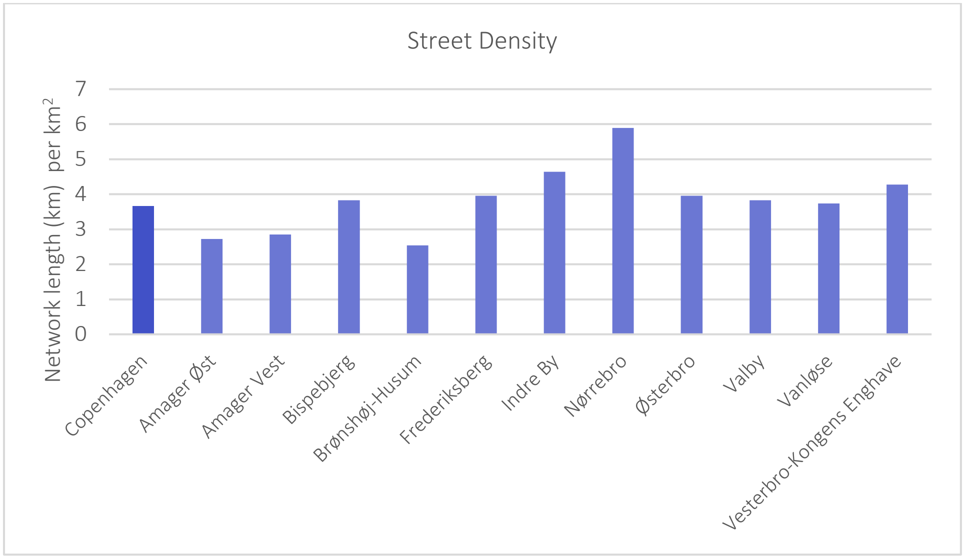

Figure 19. Neighbourhood street density

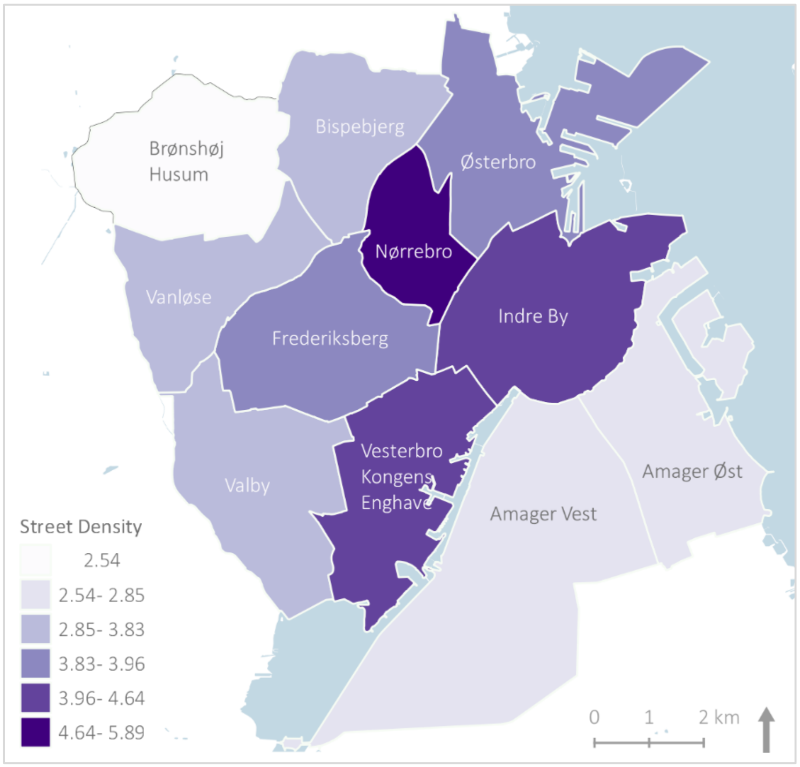

Figure 20. Street density as kilometre network per square kilometre (Natural Breaks Classification)

A high value for street density is an indication of a denser part of the network. This both implies that there is more biking infrastructure in this part of the city, but can also suggest higher connectivity, since the network segments will be closer together than in areas with a lower street density. If Figure 18 and Figure 20, depicting respectively intersection and street density in the different neighbourhoods, are compared, it becomes clear that the two types of densities in this case are correlated.

The connected node ratio (CNR) can be used to describe the extent of dead-ends and missing links in a graph. The CNR for the entire network is computed as follows:

$$CNR = \frac{intersections}{intersections + dangling\ vertices} = \frac{575}{575 + 129} = 0.82 \qquad \textit{Eq. 14. CNR}$$

This is a relatively high CNR-value (CNR is ranging from 0 to 1), which indicates high connectivity in the sense that the network has few dangling vertices relative to the total number of vertices. Normally, values over 0.7 are preferred for street networks, which entails that the network of cycling infrastructure in Copenhagen is comparable to a well-connected, regular street network.
That the graph has a relatively high CNR-value is maybe not surprising, given that the graph has no sub-graphs, and that Copenhagen is known for its well-developed network of cycling infrastructure, which connects all parts of the city and does not suffer from the same degree of fragmentation and piecemeal developments as many other cycling networks do (Vassi and Vlastos, 2014).

Is the Network Clustered?

The network has very little clustering, with no vertex having more than one connecting edge between their adjacent vertices (see p.21 for the definition of clustering), and only 79 vertices having any clustering at all (see Figure 21). Among the vertices with any clustering, the minimum local clustering coefficient $C(v)_{min}$ = 0.1, the maximum local clustering coefficient $C(v)_{max}$ = 1.0, and the average local clustering coefficient $C(v)_{ave}$ = 0.25. With a total number of 704 vertices, the global clustering coefficient, as an average of all clustering coefficients is $C = 0.03$.

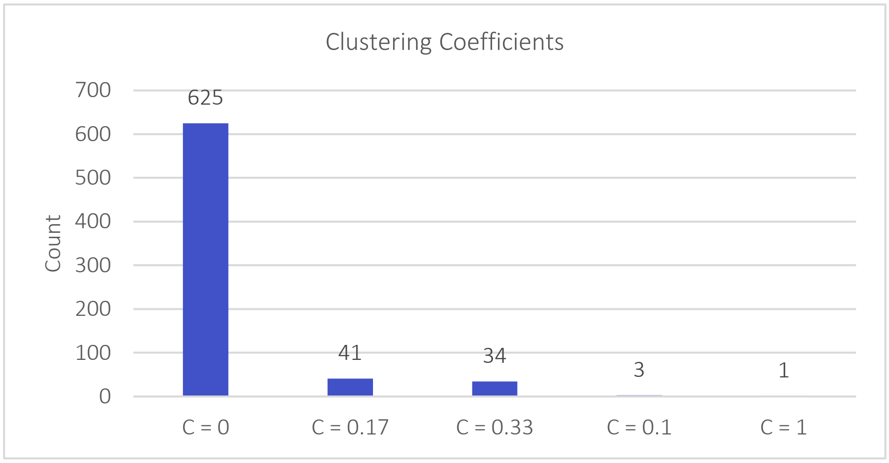

Figure 21. Distribution of clustering coefficients

The lack of clustering is in this case not necessarily an indication of the spacing between vertices or the overall connectivity in the network, but is most likely a symptom of the structure of the street network along which most of the cycling infrastructure is located. In the definition of clustering used here, a square gridded network with right angles will have no clustering, while a street network with a less gridded structure and pointed angles between edges do form the triangles measured by the clustering coefficient (see Figure 22 and Figure 23 below).

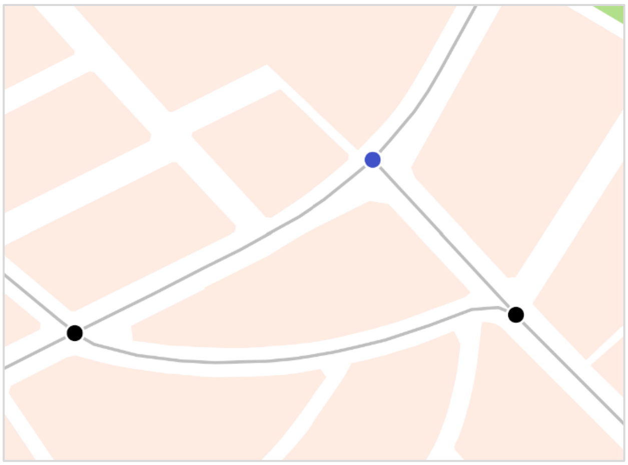

Figure 22. Example of vertex (purple) with clustering

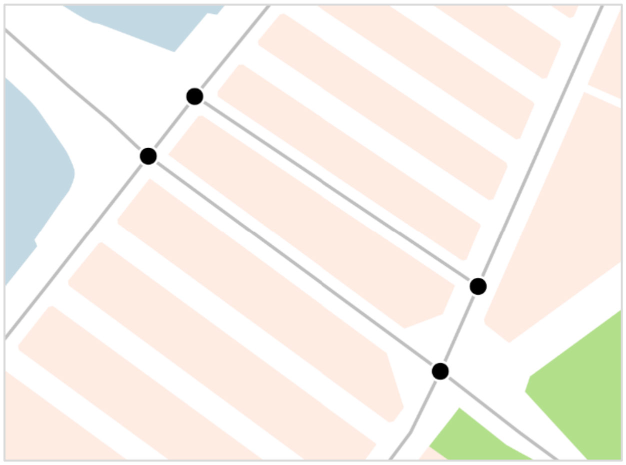

Figure 23. Example of vertices with no clustering

Average Cycling Distance Between Vertices

The concept of average path length describes the average length of the shortest path between all vertex pairs. The average path length for the graph is influenced by the weighting scheme used for the computation of shortest paths, since different weighting schemes result in different shortest paths. If the weights had solely been based on some other metric than distance – e.g. traffic volume – the average path length can be presented as a different unit that physical distance. In that case, the average path length could reveal for example the average amount of traffic or pollution a cyclist would encounter. In this case, where both distance and other factors have been combined, the average path length is however computed as the physical distance of the paths.

For the shortest paths based on the three different APSP solutions, the average path lengths, based on equation 6 are as follows:

Weighting Scheme	Average Path Length (km)	Total Path Length (km)
$W_{efficiency}$	6.14	3,037,996
W_{safety}	6.22	3,077,751
$W_{distance}$	5.7	2,820,630

Table 6. Average Path Lengths

That the average path length between a vertex pair increases, as costs other than just distance is added to the shortest path computation, is of course to be expected. The differences in average path lengths for the three weighting schemes moreover shows that the weighting scheme for efficient paths (but still with a preference for less traffic and green areas) indeed do give a shorter path length, on average, than the weighting scheme designed to avoid streets with high speed and a lot of traffic. This suggests that cyclists in Copenhagen must make a trade-off between shorter distances and any desire to avoid streets with heavy traffic or high speed.
The differences between the three weighting schemes and their influence on path lengths can also be examined from the following figures, which illustrate the variations in the total path length between each vertex to all other vertices.

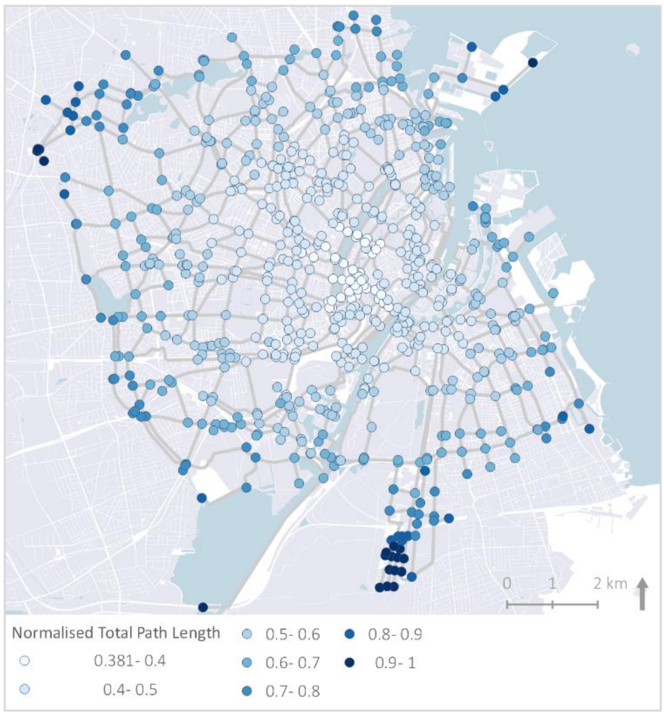

Figure 25. Vertices visualised based on the length to all other vertices ($W_{efficiency}$)

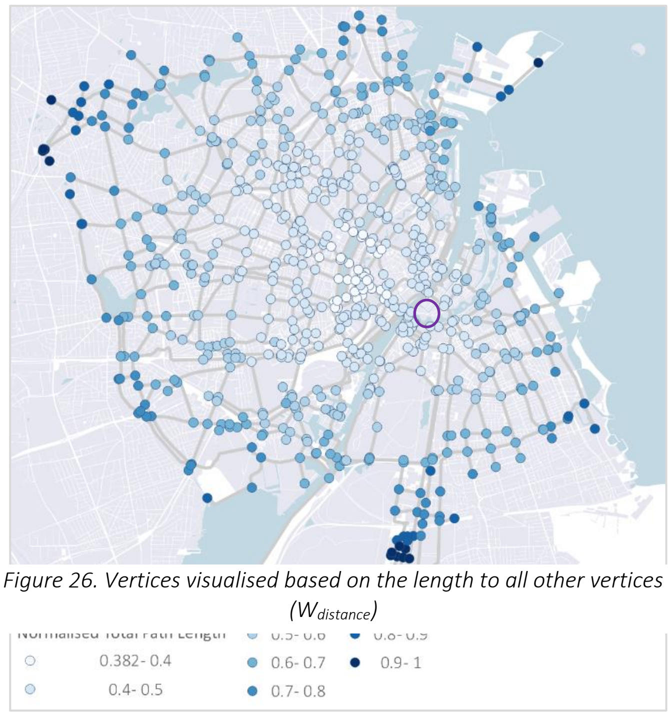

Figure 26. Vertices visualised based on the length to all other vertices ($W_{distance}$)

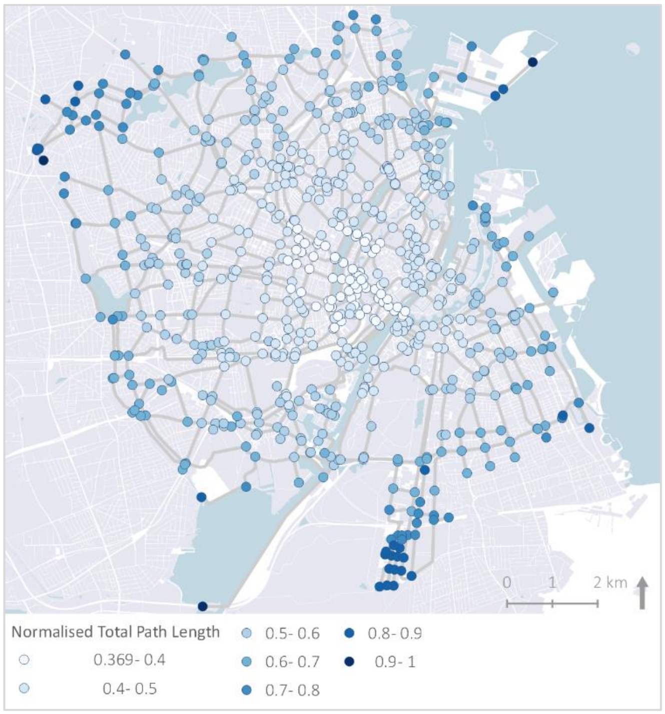

Figure 24. Vertices visualised based on the length to all other vertices (W_{safety})

When looking at the maps of the vertices visualised based on the length to all other vertices, the differences might not immediately seem significant (see Figure 25, Figure 24 and Figure 26). The maps can nevertheless illustrate where the distance to other vertices increases, when other costs than simply distances are introduced, or identify vertices which, despite of a relatively central location, are far away from other vertices when it comes to path lengths (e.g. the encircled vertex on Figure 26). That a vertex appears isolated in terms of path lengths does not necessarily imply that it is hard to access, since it is possible to cycle outside of the dedicated cycling infrastructure in many parts of the city – but it can indicate a gap or a missing link in the network for cyclists, who do not feel safe cycling without e.g. separated bike lanes.

Evaluating Network Efficiency

Based on equation 7, the network efficiency for the graph is as follows:

Weighting Scheme	Network Efficiency *(based on distance in km)*
$W_{efficiency}$	0.270
W_{safety}	0.268
$W_{distance}$	0.281

Table 7. Network Efficiency

The values for network efficiency are not necessarily interesting on their own and vary depending in the unit used to measure the length of the shortest path. Network efficiency is however useful for comparing how well different graphs functions in terms of the flow through the network and can likewise be used to identify whether e.g. a new network segment will improve the overall network connectivity. As expected, the network efficiency is best if only distance is considered as a cost, and then decreases as more costs are added. Just as for average path length, the variations in network efficiency between the weighting scheme confirms that the weighting scheme for efficiency indeed does results in a more efficient network compared to the scheme for safety.

To sum up, the connectivity indices presented above indicate a relatively well-connected network, judging from the CNR-value, and a network which is neither completely simple nor entirely gridded or very complex, based on the values for alpha, beta, and gamma. While these measures have been developed to describe graphs as a general concept, they must be interpreted in the specific context of street networks, or more specifically in the context of networks of cycling infrastructure. A street network will for example usually not be very complex or have high values alpha-values, since alpha-values approaching the maximum value of 1 indicates a network with edges between all vertices, which is not practically possible in a street network at the size of Copenhagen. Similarly, the CNR-value for Copenhagen might actually be very high if it was compared to other networks of cycling infrastructure, since they oftentimes are fragmented and scattered across a city (Vassi and Vlastos, 2014). The variations in network density do however also indicate that there are substantial differences in the network connectivity between neighbourhoods, and that the access to cycling infrastructure is not evenly distributed throughout the city.

Centrality: Comparing Network Elements

The results presented in this section are, as a starting point, local values, meaning that they describe properties for an individual network element. The local values will, for most centrality indices, be visualised on maps, and otherwise be transformed into global values by finding the minimum, maximum, median, and average for each centrality index. Global values for each neighbourhood can be found in Appendix I. The centrality indices have, when possible, been computed for edges rather than vertices, but e.g. closeness or straightness centrality can only be calculated for vertices.

Vertex Degree & Edge Intersections

The vertex degree can be considered a measurement of reachability for the individual vertex, but also functions as a global measurement for how dense or sparse the graph is. As mentioned before, a street network has some physical limitations which rules out high degree values. For the Copenhagen network of cycling infrastructure, the minimum degree is 1 (representing dangling vertices), the mean is 2.83, and the maximum degree is 5, which however only occurs at 4 vertices (see Figure 27). The median degree value, with 313 out of 704 vertices, is 3, representing three-way junctions.

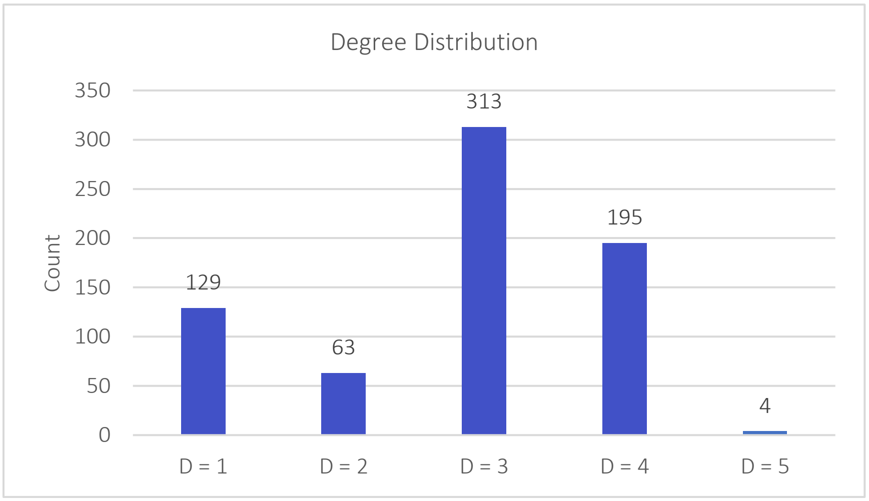

Figure 27. Vertex degree distribution

Stress Centrality: Identifying Central Edges

Stress centrality $C_s(e)$ for a graph describes the number of shortest paths which passes through a network element. Since the shortest paths depend on the weights assigned to the network elements, this centrality index varies between the three weighting schemes.

The minimum stress centrality for the Copenhagen network is 0, meaning that there are several edges which are not part of any shortest paths. This is possible because shortest paths are based on vertex pairs – but does of course not imply that the given edge is not actually used by cyclists. In the interpretation of the result it is important to remember that stress centrality is a theoretical construction, assuming that there is exactly one cyclist moving between each possible pairing of vertices.

Weighting Scheme	Minimum Stress Centrality	Average Stress Centrality	Median Stress Centrality	Maximum Stress Centrality
$W_{efficiency}$	0	4,764	1,988	45,640
W_{safety}	0	4,763	2,003	44,270
$W_{distance}$	0	4,621	2,845	33,378

Table 8. Stress Centrality

The stress centralities presented in Table 8 shows that when other costs than just distance are introduced in the computation of shortest paths, the paths tend to concentrate on a smaller number of edges, resulting in higher average and higher maximum stress centralities. It is furthermore evident that there is a rather uneven distribution of stress centrality across the network. This is the case for all three weighting schemes, but as can be seen on the maps below (see Figure 28, Figure 29, and Figure 30), the stress centrality indices for the weighted shortest paths have a more uneven distribution than the stress centralities only based on the distance between vertices. The high stress centrality index for a small number of edges indicates that the network in some areas have a low resilience, since a high number of shortest paths are depending on a small number of network elements. Given that the network is planar, higher values for stress centrality in the centre of the graph is to be expected, but as evident on the visualisations below, a central location does not explain all of the high values and uneven distribution. Although the stress centrality indices should not be confused with numbers for actual traffic flow, this uneven distribution of stress centrality matches the current situation of the city, where some centrally placed bike lanes and cycle bridges experience very high numbers of cyclists (City of Copenhagen, 2016c).

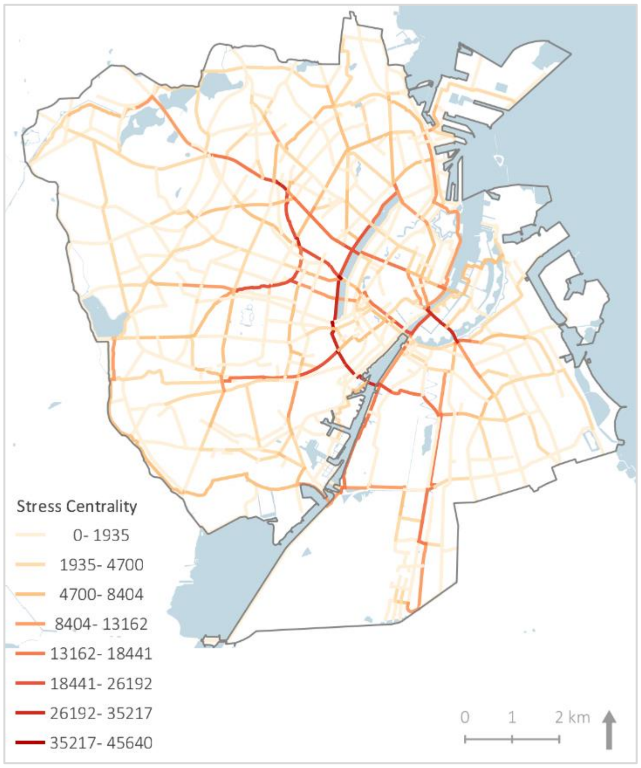

Figure 28. Stress Centrality based on $W_{efficiency}$ (Natural Breaks Classification)

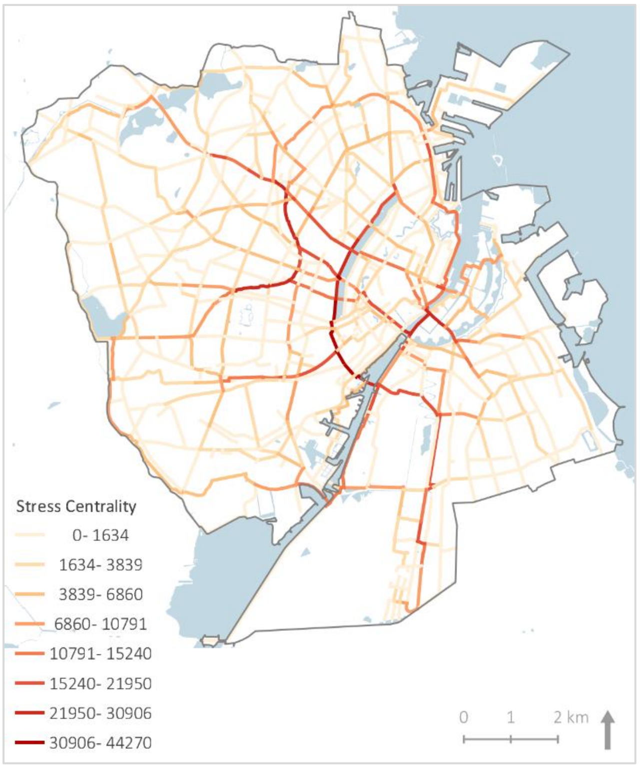

Figure 29. Stress Centrality based on W_{safety} (Natural Breaks Classification)

The uneven distribution of stress centralities is to some extent a consequence of the city's geography, where the water and canals intersecting the city result in a lot of pressure on bridges, particularly the bridges connecting Amager and Islands Brygge with the rest of the city. The City of Copenhagen has for several of the new bridges been positively surprised by the high number of daily users (City of Copenhagen, 2017c; FAOD, 2019). Given the central role which the bridges play in connecting different parts of the network, it is however not surprising that several of the cycling bridges are among the edges with the most daily cyclists (City of Copenhagen, 2019b).

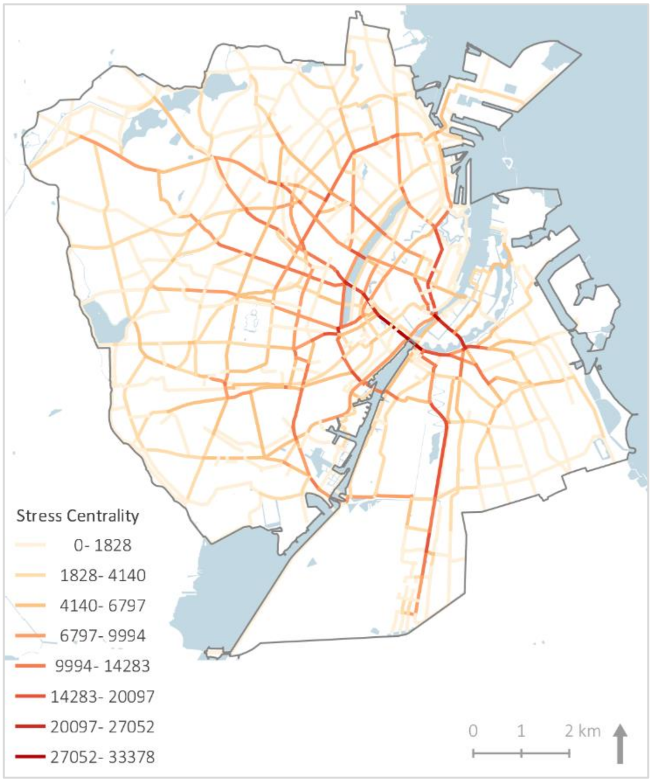

Figure 30. Stress Centrality based on $W_{distance}$ (Natural Breaks Classification)

As evident from Figure 31 and Figure 32, the differences in stress centrality between the weighting scheme based solely on distance and respectively efficiency and safety follow the same pattern. Although the average stress centrality is higher for $W_{efficiency}$ and W_{safety} at a city scale, this is not true for all neighbourhoods. For some neighbourhoods, as for example Bispebjerg, adding more factors such as traffic volumes or speed limits results in a higher average stress centrality, while it for other neighbourhoods result in a lower average stress centrality.

The variation in stress centrality needs to be interpreted carefully, since the sum and distribution of stress centrality values depend on, for example, whether paths tend to go through many short edges or few, longer edges. The average value does furthermore not necessarily say anything about the dispersion of paths between edges. In the case of Bispebjerg both minimum, mean and median values for $W_{efficiency}$ and W_{safety} are however higher than for $W_{distance}$, indicating that more paths go through that neighbourhood when other factors are added to the computation of paths. The maps seen below can in this way be used as an indication of areas where the shortest paths are changed considerably by the respective weighting schemes.

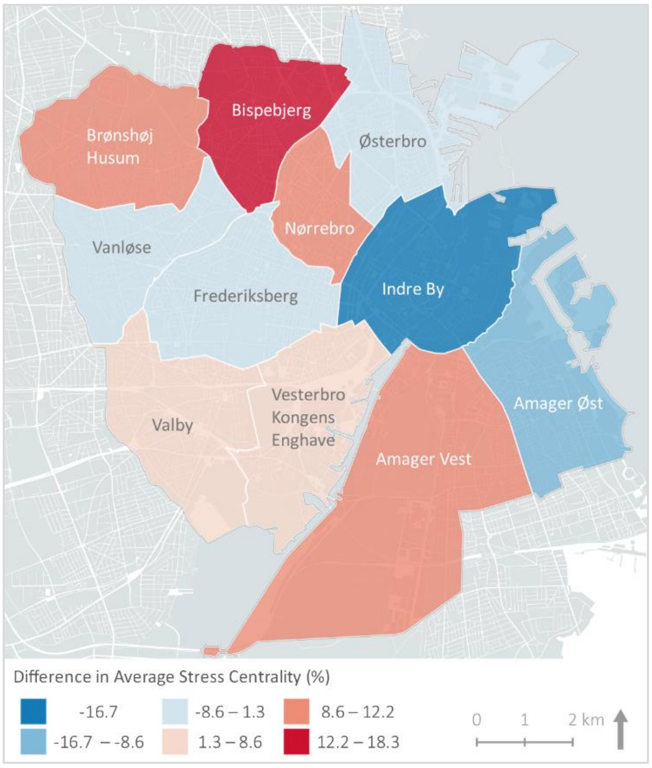

Figure 31. Differences in average Stress Centrality for $W_{efficiency}$ compared to $W_{distance}$

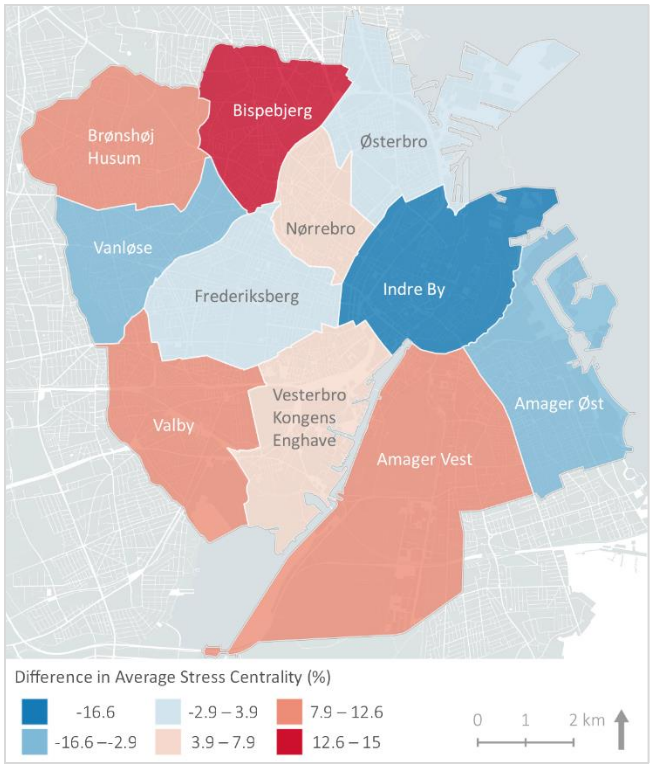

Figure 32. Differences in average Stress Centrality for W_{safety} compared to $W_{distance}$

Closeness Centrality: How Easy are Vertices to Reach?

Closeness centrality $C_c(v)$ for a vertex measures reachability based on the average distance from a vertex to all other vertices in the network. The computation of closeness centrality for the network confirms the picture that reachability and access for the networks decreases as more of the factors which can influence the experience of cycling are introduced in the analysis (see Table 9). As explained earlier (see p.24), the closeness centrality value represents the inverse distance to all other vertices. In this case this entails that higher values are preferred, since they indicate that a vertex is close to the other vertices in the network.

Weighting Scheme	Minimum Closeness Centrality	Average Closeness Centrality	Median Closeness Centrality	Maximum Closeness Centrality
$W_{efficiency}$	0.091	0.172	0.169	0.238
W_{safety}	0.090	0.170	0.166	0.237
$W_{distance}$	0.094	0.184	0.181	0.256

Table 9. Closeness Centrality, based on the distance in kilometres

The differences in closeness centrality computed with shortest paths based on respectively $W_{efficiency}$ and W_{safety} does not appear to be significant. On the other hand, the use of $W_{distance}$ results in higher closeness centralities in general, although the difference between the weighted and unweighted closeness centralities vary across the network (see Figure 33, Figure 34, and Figure 35). This does not necessarily mean that vertices in areas with a smaller variation between the three different computations of closeness centralities are less influenced by the factors used for assigning costs. Instead, the lack of variation in closeness centrality for vertices in some neighbourhoods might reflect that there often only is little if any change in the paths around those vertices. This is most likely due to the sparsity of the network there and hence a lack of alternative routes, particularly if long detours are not considered (see Figure 36 and Figure 37 below).

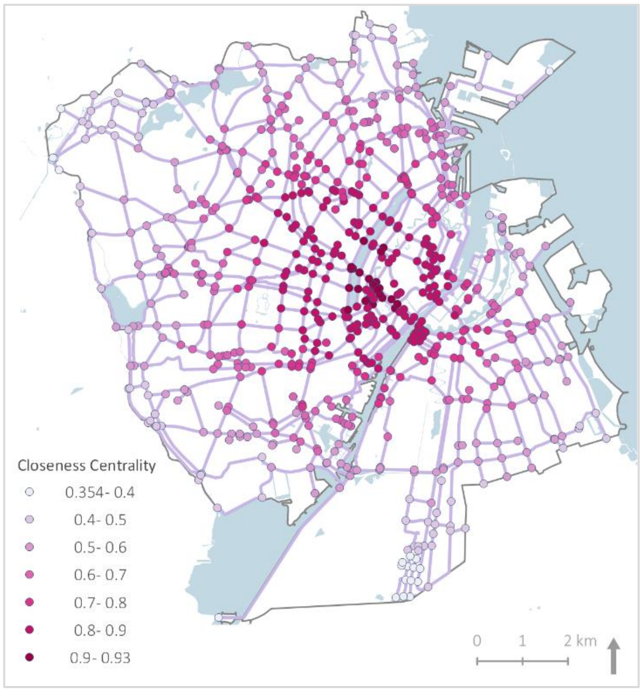

Figure 33. Closeness Centrality based on $W_{efficiency}$ (normalised values)

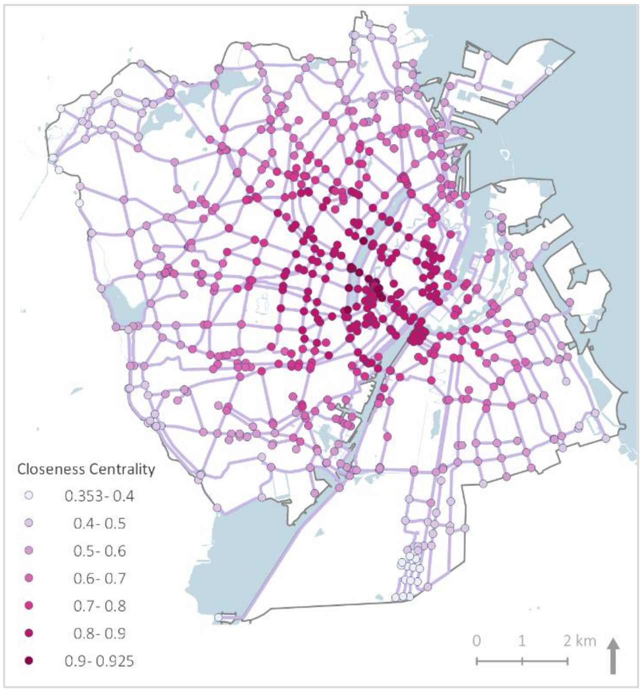

Figure 34. Closeness Centrality based on W_{safety} (normalised values)

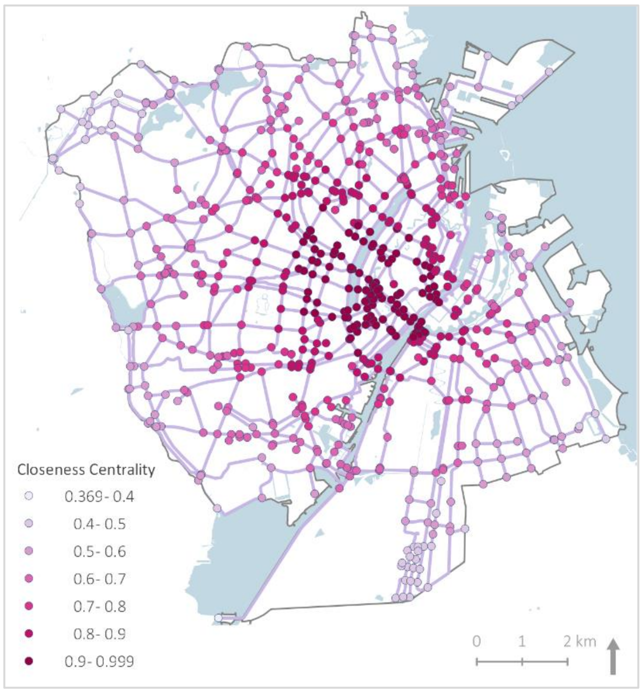

Figure 35. Closeness Centrality based on $W_{distance}$ (normalised values)

As can be seen in Figure 36 and Figure 37, the differences between $W_{efficiency}$ and W_{safety} compared to $W_{distance}$ vary between neighbourhoods, suggesting that the introduction of additional factors for some neighbourhoods result in longer detours and a larger drop in reachability than in others.

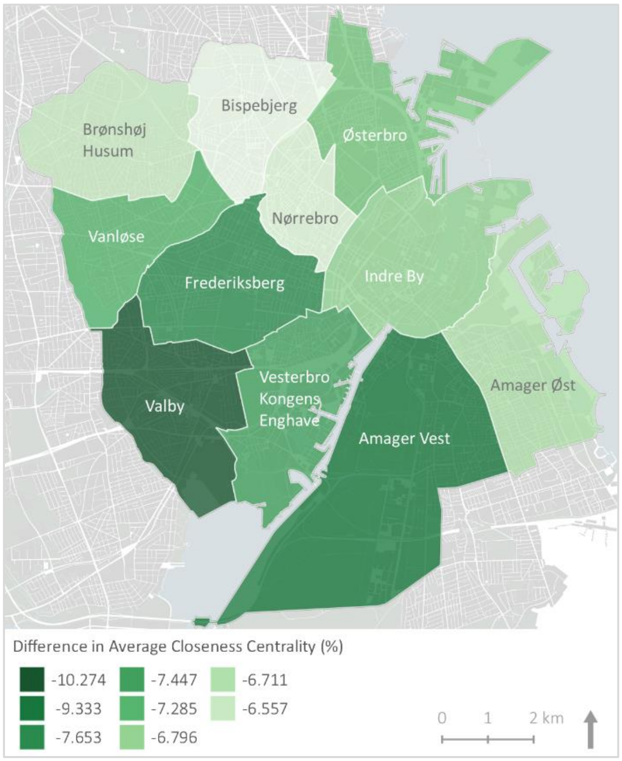

Figure 36. Differences in average Closeness Centrality for $W_{efficiency}$ compared to $W_{distance}$

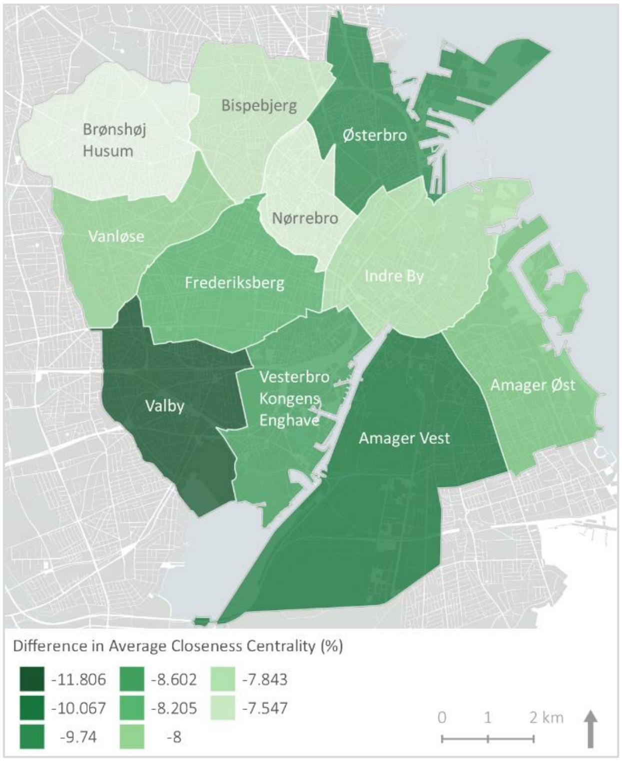

Figure 37. Differences in average Closeness Centrality for W_{safety} compared to $W_{distance}$

Straightness Centrality: How Long are the Detours?

Straightness centrality $C_s(v)$ is a measure for reachability based on a comparison of the length of the shortest path between two vertices and the distance between them based on a straight line. Higher values indicate that the shortest path is closer to a straight line, whereas lower values indicate longer detours. It is naturally not surprising that the shortest paths optimised solely according to distance results in a higher average straightness centrality (See Table 10). The straightness centrality index can however still be useful for identifying locations where adding other considerations than just distance appear to result in longer detours (see Figure 38, Figure 39Figure 40). Straightness centrality can additionally identify locations that, despite a central geographical location, have low accessibility and thus requires long detours to reach. Some of the vertices with very low straightness centralities (see encircled area on Figure 40) have indeed already been identified as an area needing better connectivity and the municipality is therefore planning to add a connection across the water (Køhler, 2020).

Weighting Scheme	Minimum Straightness Centrality	Average Straightness Centrality	Median Straightness Centrality	Maximum Straightness Centrality
$W_{efficiency}$	0.584	0.769	0.779	0.843
W_{safety}	0.579	0.760	0.771	0.840
$W_{distance}$	0.614	0.817	0.827	0.874

Table 10. Straightness Centrality

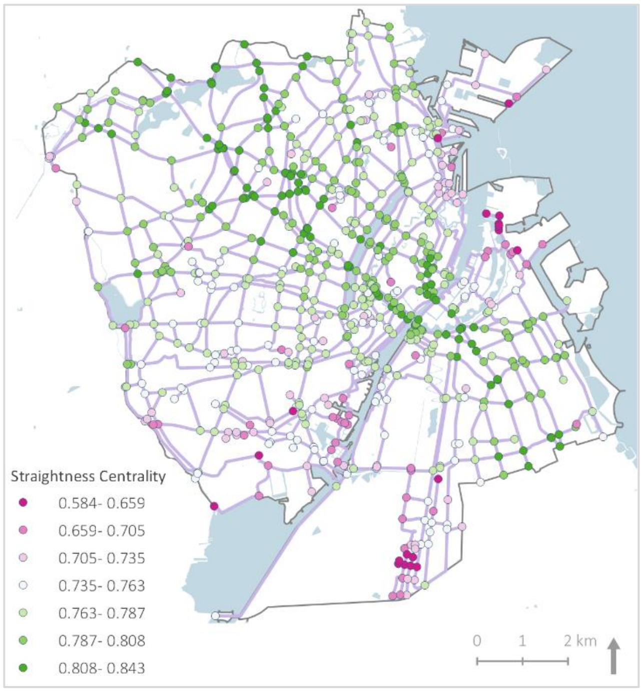

Figure 38. Straightness Centrality based on $W_{efficiency}$ (Natural Breaks Classification)

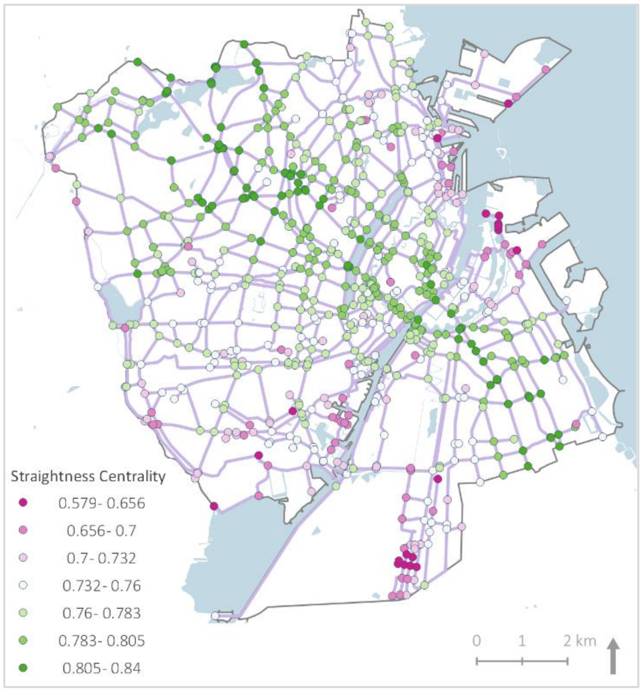

Figure 39. Straightness Centrality based on W_{safety} (Natural Breaks Classification)

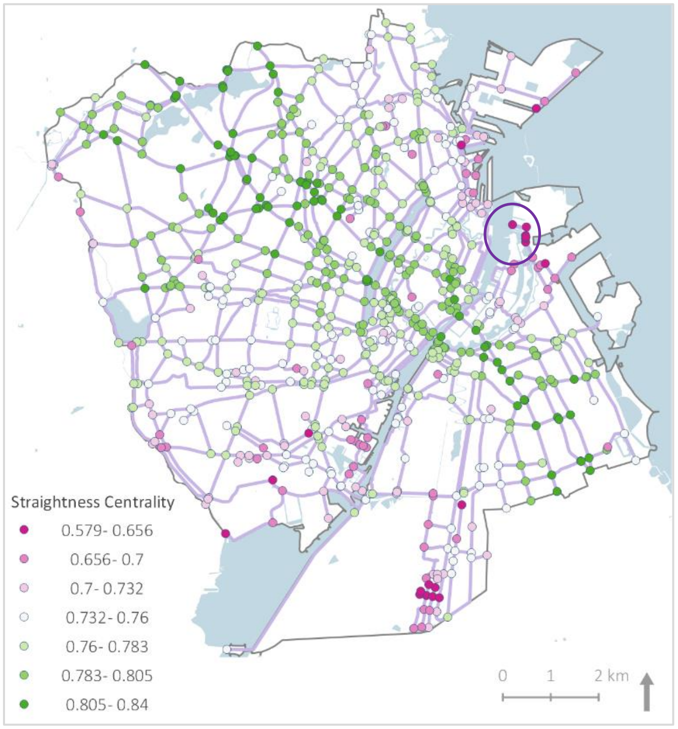

Figure 40. Straightness Centrality based on $W_{distance}$ (Natural Breaks Classification)

Looking at the maps below, which compare the differences between straightness centralities based on $W_{distance}$ and respectively $W_{efficiency}$ and W_{safety} (Figure 42 and Figure 41), confirms the picture about changes in reachability when other factors than distance are introduced. While the straightness centralities generally are lower when shortest paths are based on more than distance, the drop in straightness centrality, implying longer detours, is more pronounced for some neighbourhoods than others. Particularly the neighbourhood Valby seems to experience a noteworthy drop in straightness centrality, which could indicate that the most direct route used in a lot of shortest paths is suffering from factors such as a lot of car traffic or high speed, which will make the weighted shortest path computations find longer detours to avoid those costs.

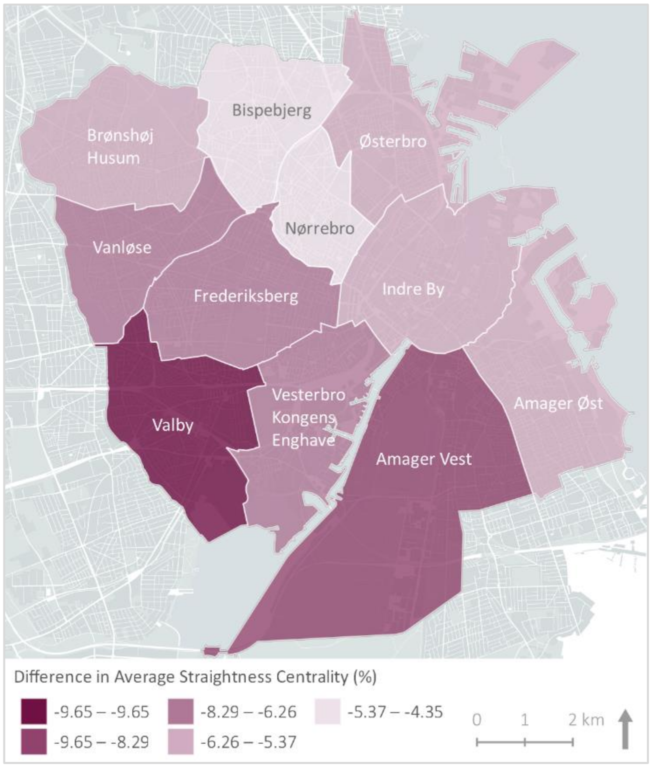

Figure 42. Differences in average Straightness Centrality for $W_{efficiency}$ compared to $W_{distance}$

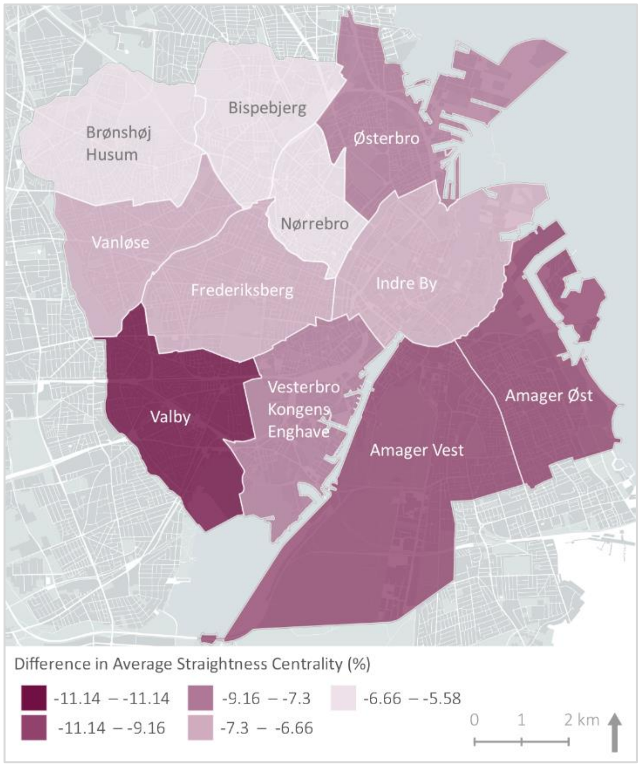

Figure 41. Differences in average Straightness Centrality for W_{safety} compared to $W_{distance}$

To sum up, the centrality indices presented above indicates a network which have some considerable internal variations, despite generally having a high level of connectivity, especially interpreted in the context of street networks. The average degree of the network of cycling infrastructure is similar to that of a regular urban street network, confirming the picture of a network of cycling structure that fulfils the purpose of connecting different segments – in other words, most edges do actually connect to other edges. Combining this metric with the global range for stress, closeness, and straightness centrality however confirms what was also shown by the neighbourhood variations in network density: there are noteworthy variations in reachability and connectivity within the network.

When it comes to stress centrality, the large value range indicates a network which in many places is depending on a few central edges and thus have a low resilience. This tendency is reinforced when other factors are introduced into the computation, suggesting that if cyclists want to avoid e.g. segments with high speed and/or prefer to bike along green areas and on separated bike lanes, the dependency on a small number of network segments becomes even greater.

The closeness centralities throughout the network not only reveals the more obvious observations, such as the fact that network elements in the exterior neighbourhoods indeed are further away from all other segments than those in the middle of the network. The closeness centralities based on the weighted shortest paths also indicate that the reachability drops much more for some neighbourhoods than others if additional preferences among cyclists are taken into account. This implies that some areas are hard to reach if you want to avoid e.g. high-speed streets. That the closeness centralities do change for some areas is however also an indication that there is more than one potential route to a given vertex. Consequently, the lack of change in closeness centrality for other vertices might not mean that they shortest paths leading to them are not weighted differently when more factors are added to the shortest paths computation, but might simply be due to the fact that there is no other possible path in this area.

This picture is finally confirmed by the straightness centrality. The value range and variations here both indicates that, although the landscape in Copenhagen plays an important role for the reachability for some areas, there are network elements which, independent of the topography, requires much longer detours than others, particularly if additional factors are added to the computation of shortest paths.

6. Interpretation & Discussion

In this final chapter I will start out with a contextualisation and interpretation of the results in relation to the city's neighbourhoods and their respective differences in demography and socio-economic characteristics, followed by a discussion of how the results of the analysis can be interpreted and applied. Finally, I will discuss some of the assumptions and limitations for the analysis and resulting conclusions, along with ideas and potentials for further research. Through this discussion, the research question of how a network analysis can be used to inform urban planning decisions will be answered, along with considerations for how a network analysis can incorporate a broader range of factors and variables to more closely reflect different types of cyclists' experiences of the network.

Examining Neighbourhood Variations

From the results presented above, it is clear that connectivity and centrality indices vary between neighbourhoods, and that, although the topography in and around Copenhagen definitely plays a role in the uneven accessibility, not all variations can be explained by the landscape. Although Copenhagen, when compared with other European capitals, is a fairly small city, the city's neighbourhoods not only have distinct differences in terms of the density of cycling infrastructure, but also when it comes to factors such as population density, the average income, educational level of the residents, age distribution, etc.

The analysis of the network has so far looked at accessibility in the network both defined as general, global metrics and through more local variations. These local variations will, in the following section, form the foundation for an interpretation of how the network of cycling infrastructure performs in terms of equity, i.e. to what extent network variations coincide with differences between neighbourhoods when it comes to for example income and population density. The data on neighbourhood variations is, due to the choice of the neighbourhood as the analytical scale, distributed on only 11 datapoints and do furthermore not fulfil the basic requirements for e.g. linear regression. The results below will thus not be presented in the form of any quantified correlation, but instead use plots to detect and discuss the extent to which any correlations between neighbourhood demographics and centrality and connectivity indices exist.

All connectivity and centrality indices as well as all neighbourhood data have been plotted together to look for any sign of a positive or negative correlation. Only some of the variables showing some type of correlation will be presented here, but all plots can be found in Appendix II. The variables used in the plot below are, when based on shortest paths, all based on the shortest paths computed only with distance as cost.

Neighbourhood Characteristics

The maps presented on this and the following page show how the Copenhagen neighbourhoods differ in terms of population density, proportion of population with a long education, and average income (see Figure 43, Figure 44 and Figure 45). Population densities are to a large extent results of the neighbourhoods being dominated by different types of building stock, with a higher share of dense, multi-storey buildings in the centrally located neighbourhoods. Population density is also influenced by the proportion of real estate used for commerce and offices (e.g. in the case of Indre By) and how much of the neighbourhood is taken up by green areas (e.g. in the case of Amager Vest).

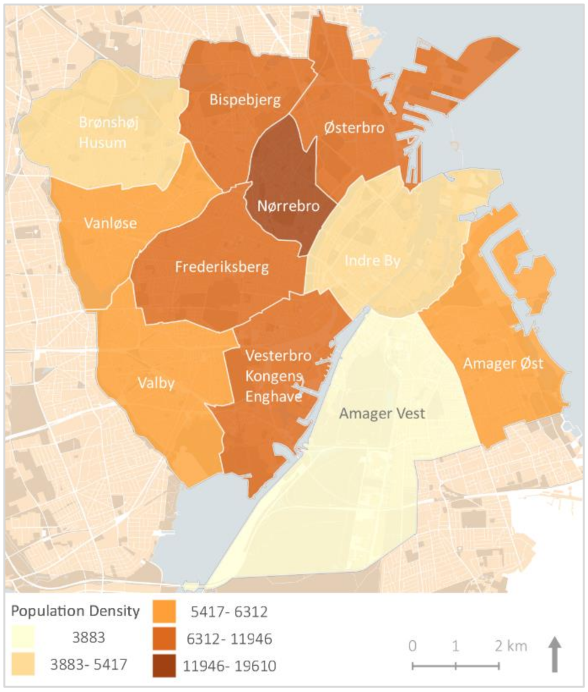

Figure 43. Neighbourhood population density

Although there is no perfect correlation between income and education, Figure 44 and Figure 45 give a good indication of the pattern of socio-economic stratification in Copenhagen, which also is present in similar patterns for factors such as health and life expectancy (City of Copenhagen, 2017b).

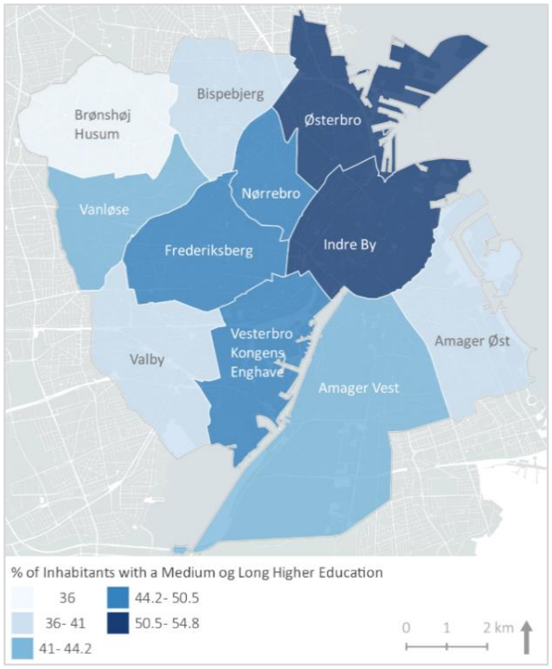

Figure 44. Share of population with a medium/long education

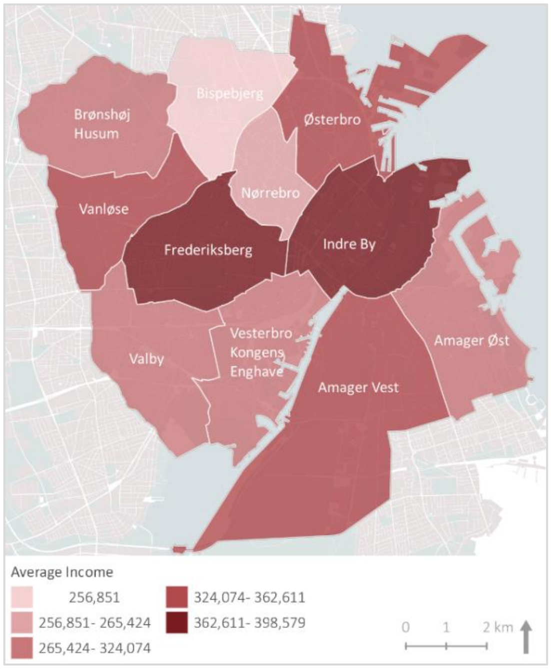

Figure 45. Average income (Dkr.) across Copenhagen

Correlating Neighbourhood Characteristics with Network Variations

Network Density & Centrality Indices

When looking at the correlation between neighbourhood characteristics and network variations, it is important to keep in mind that, just like the average income and the resident's level of education is positively correlated, so are several of the centrality and connectivity indices. That some centrality indices, like closeness and stress centrality will be correlated is somewhat intuitive, at least in a planar and connected network, like the one analysed here. That street density and closeness centrality, as well as intersection density and stress centrality (see Figure 46 and Figure 47), appear to be positively correlated, where neighbourhoods[13] with a higher network density also tend to have higher closeness and stress centralities is however not a given. Instead, it indicates that neighbourhoods, which already are located centrally in the network, at the same time also are in denser and thus more connected areas. This furthermore means that high values for stress centrality cannot be explained with a lack of alternative routes, since neighbourhoods where many paths pass

[13] Neighbourhoods are labelled with their neighbourhood ID (1: Indre By, 2: Østerbro, 3: Nørrebro, 4: Vesterbro-Kongens Enghave, 5: Valby, 6: Vanløse, 7: Brønshøj-Husum, 8: Bispebjerg, 9: Amager Øst, 10: Amager Vest, 11: Frederiksberg)

through, also have a high network density and thus a high number of bike lanes and routes (although it might still be the case that areas with high stress centralities have few alternatives that fulfil the preferences of cyclists in terms of safe and low stress conditions).

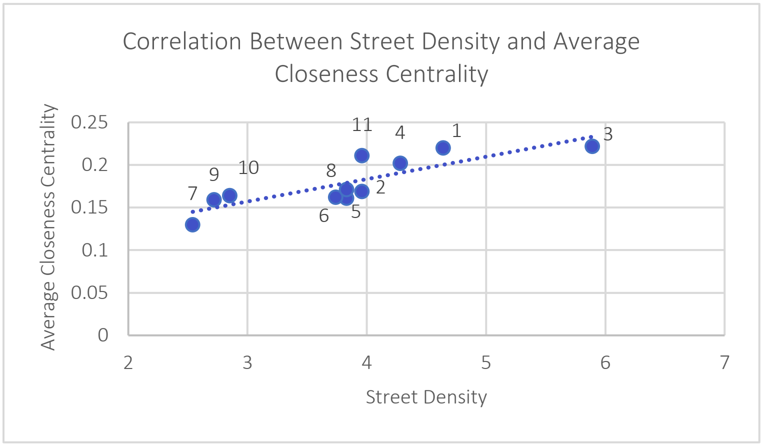

Figure 46. The correlation between street density and average closeness centrality

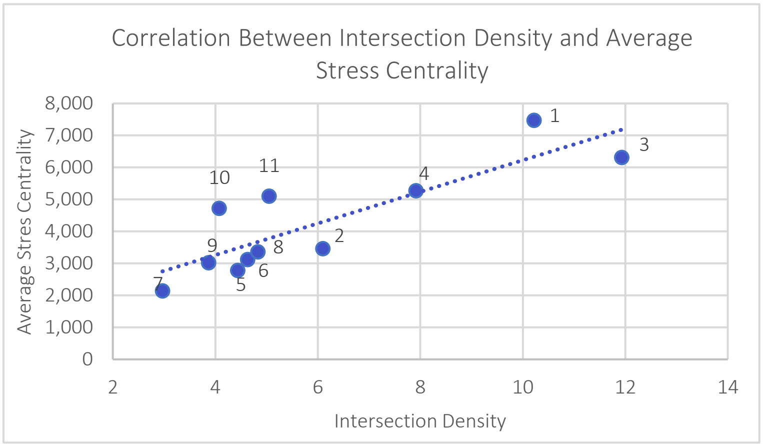

Figure 47. The correlation between intersection density and average stress centrality

When looking at the connection between neighbourhood statistics and network indices, the first conclusion is that there is no perfect linear relationship between any demographic variable and any of the centrality or connectivity indices. There does however still appear to be some type of positive correlation between population and network density, and between several network indices and the share of a neighbourhood's population with a long higher education.

While discussing the relationship between variables it is important to remember that correlation does not imply causation: even though two variables might follow a similar pattern, that does not necessarily mean that an increase in one variable directly causes an increase in the other. Detecting correlation, even without claims of causality, can nevertheless still be used in a discussion of whether some areas are underserved by the current infrastructure, and whether these areas, according to ideas of vertical equity should have access to an above average quality of transportation infrastructure (see p.9 for a description of equity within transportation planning).

Population & Network Density

As can be seen on Figure 48 and Figure 49, there appears to be a positive correlation between population and network density. The network is thus denser and have a better connectivity in areas with more residents. Exceptions are Indre By (1) and Frederiksberg (11) which appears to have respectively a higher and lower network density than their population densities warrants. That Indre By, being the city centre, has a high network density makes sense, given that the neighbourhood experiences a lot of traffic from other places and has a high number of businesses. It is however interesting that Frederiksberg, which belongs to a different municipality that than the other ten neighbourhoods, appear to have a considerably lower intersection density relative to the neighbourhood's high population density.

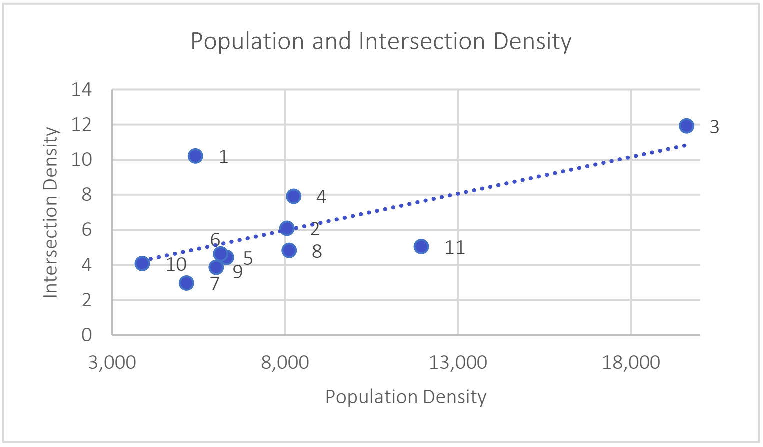

Figure 48. Correlation between population density and intersection density

Plotting population density with network density also shows that, although for example the neighbourhood Brønshøj-Husum (7) has a much lower network density than most other neighbourhoods, there are also much fewer potential cyclists based on the neighbourhood's low population density.

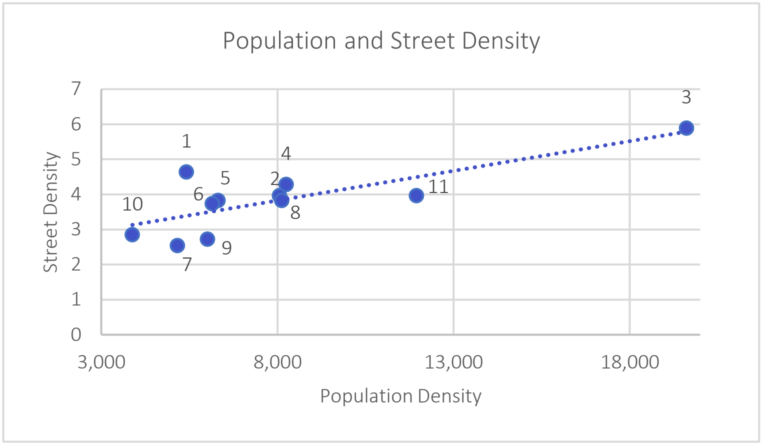

Figure 49. Correlation between population density and street density

Educational Level & Network Quality

Network density is not only correlated with population density, but also with the share of the population with a longer higher education (see Figure 50). Likewise, the level of education among a neighbourhood's residents appear to be correlated with the average stress and closeness centrality, in the sense that people with a longer education tend to live in central areas with a denser and more connected network (see Figure 50, Figure 51, and Figure 52), indicating that they have better access to cycling infrastructure.

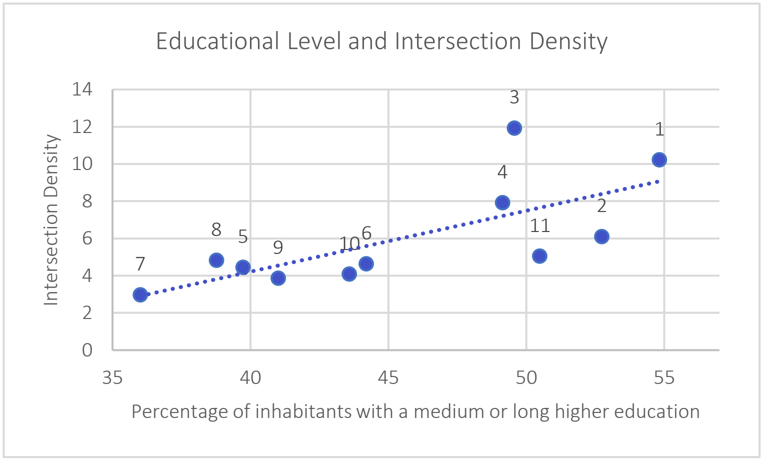

Figure 50. Correlation between educational levels and intersection density

The average income for a neighbourhood does, for most neighbourhoods, also appear to have a weak positive correlation with closeness and stress centrality, although the relationship is more ambiguous (see Appendix II for all plots).

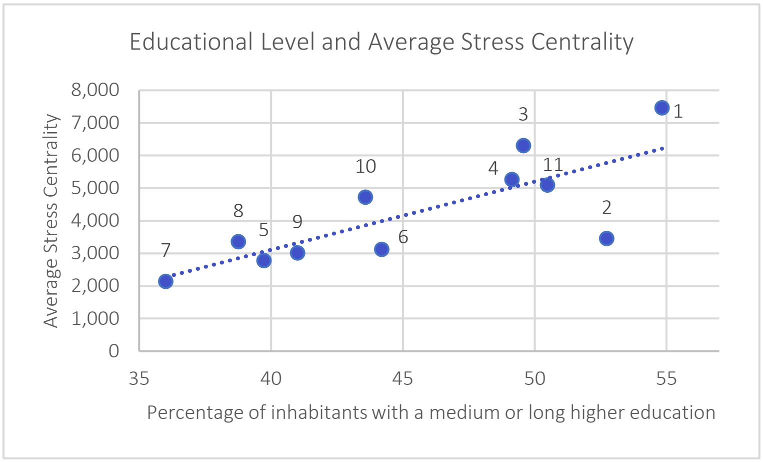

Figure 51. Correlation between educational level and average stress centrality

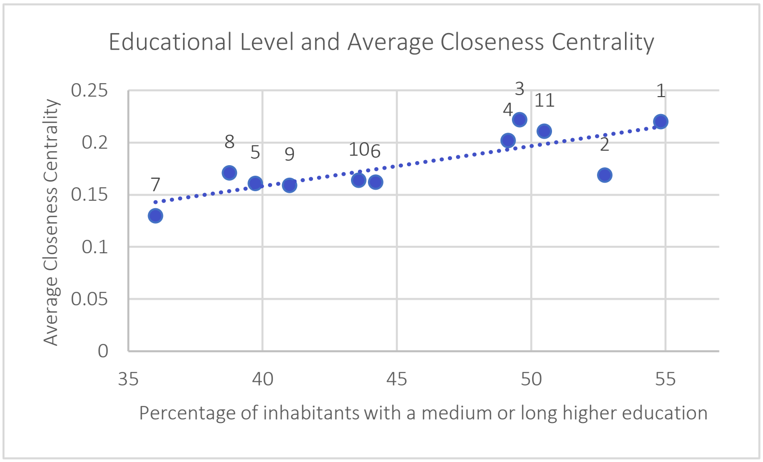

Figure 52. Correlation between educational level and average closeness centrality

Plotting the selected neighbourhood variables with the area's average connectivity and centrality indices have thus shown that the level of network density coincides with population density, and that the people with a longer higher education tend to live in more central areas in the city/network, which generally have a higher level of connectivity and accessibility. What this means in terms of transportation planning and the ideals of equity within access to transportation infrastructure will be discussed further in the following section, in which I will look at the wider implications and potential interpretations of the results of the network analysis.

Interpretation & Application of Results

The purpose of this research project has first of all been to explore how a network analysis based on graph theory can be used to examine and assess the quality and structure of a network of cycling infrastructure. The evaluation of the network is however not only interesting in itself, but also as a potential tool for the planning and evaluation of future improvements and extensions to the network. In the following section I will summarise what the results says about the Copenhagen network compared to other cities, outline suggestions for how to interpret the results in an urban planning context, and discuss how the methodology might be applied in the planning process.

From the results presented above it is clear that a network analysis both can reveal some general features of a network, but also is well-suited for identifying areas and elements which stands out from the rest of the network. When looking at the network in its entirety, the Copenhagen network of cycling infrastructure is mostly well-connected, does not suffer from fragmentation, and has a structure which in several ways resembles that of a regular street network. Particularly the fact that the city's cycling infrastructure is not connected into subgraphs but functions as one, integrated network distinguishes it from most other cycling networks. Other than that, it is however hard to make any direct comparison between the Copenhagen network and networks in other cities based on the network indices. Very few comprehensive analyses of cycling networks have been made (Schoner and Levinson, 2014), partly because the cycling infrastructure in many places is so dispersed and poorly connected that a network analysis hardly is meaningful (U.S. Department of Transportation, 2018; Furth and Noursalehi, 2015). Those that do exist moreover use custom-made methods, making any direct comparison difficult[14], and results available for comparison are often based on different assumptions and types of data. The Copenhagen network of cycling infrastructure for example appears to have a higher network density than the network in Amsterdam, but the computation of density in Amsterdam is, opposed to those computed for Copenhagen in this analysis, only based on segregated bike lanes and not routes and other types of paths (Niedderer et al., 2017). In many other studies[15] the regular street network for motorised vehicles is, due to a low number of actual bike lanes and routes, used as a proxy for the cycling network, which however makes it impossible to compare results with an analysis only including bike lanes and routes. Finally, the approach here both uses positive and negative

[14] This is for example the case for the otherwise meticulous methods developed and used by People For Bikes.
[15] See for example Tresidder, 2005; Kent and Karner, 2019; Dill, 2004.

characteristics, to either subtract or add costs. This is contrary to many other classifications of cycling networks that mostly, if not only, operates with negative factors (Zuo, and Wei, 2019; Bhuyan et al. 2019). Although it can be interesting to describe a network at a generalised level, despite challenges when it comes to inter-city comparison, the most interesting and useful part of a network analysis like the one conducted here might be its ability to reveal internal differences and disparities within the network.

It has already been concluded that there are substantial variations in network density and connectivity, and that accessibility, understood as how easy it is to reach or be reached from other parts of the network, therefore is unevenly distributed throughout the city. It is moreover evident that the variations in network density and connectivity do not appear to be completely random, but instead are correlated with the location in the network, with higher rates of access in the city centre and lower values in the peripheral neighbourhoods. This pattern is additionally related to both population density and the share of the residents with a longer higher education. A longer education is in Denmark not only correlated to higher income, but also to a longer life expectancy, better health, and a higher participation in the democratic process (Statistics Denmark, 2005; Ministry of Health, 2014; Vilhelmsen, 2016; Ejlertsen, 2020; Nielsen, 2019). Interpreted in terms of transportation equity, this means that the current network not only fails in terms of horizontal equity, i.e. there is not a uniform level of access, but even more in terms of vertical equity, since some of the neighbourhoods that are doing worse when it comes to many of the traditional variables for socio-economic status and well-being have a lower quality network. These neighbourhoods are additionally not covered by the city's expanding subway system and might also have a lower access to cars, since there usually is a clear connection between income and car ownership (City of Copenhagen, 2018b).

While it could be argued that the variation in e.g. network density is understandable and natural, since it fits the variation in population density across the city, the population density is, due to the popularity and attractiveness of many of the central and densely build neighbourhoods, strongly connected to other variables such as income, education, health, etc. Therefore, a seemingly logical pattern in the distribution of network density inadvertently results in a lower access to cycling infrastructure for less privileged groups. It should nevertheless still be noted that these differences are taking place within a city which overall has a very high quality of cycling infrastructure compared to other cities (Colville-Andersen, 2017). That some neighbourhoods have a lower network quality should thus not be interpreted as meaning that they have a low quality – only that there are internal differences, which might be useful to consider when planning future expansions and improvements of the network.

When comparing neighbourhoods, it is also important to remember that the network has been weighted based on a range of different factors, which have been combined to form a single score for each edge. This can conceal other variations between neighbourhoods, e.g. in terms of the number and length of routes going through green areas or the amount of car traffic. The connectivity and centrality indices presented here are therefore only one perspective on network quality and needs to be combined with other forms of assessment and evaluation. With that in mind, the computation of for example stress centrality might still be a useful tool for identifying network segments which might need to be expanded, or for suggesting where new bike lanes and routes should be considered to relieve some of the pressure on the most used bike lanes. Similarly, adding potential future bike lanes to the dataset will allow you to visualise how different solutions will improve connectivity in an area. The method of assigning weights to edges based on their different characteristics can furthermore be adapted to different cyclist profiles, in order to identify how the network is experienced by for example cyclist who feel unsafe without separated bike lanes, cyclists who want to avoid polluted streets, and so on.

At the moment, the planning of new cycling infrastructure is relying on a range of different sources to identify where the network is in need of improvement, such as urban planners' personal knowledge about the network and inputs from citizen and neighbourhood organisations. Common to these different sources is however that they rely on personal experiences from people who are interested in cycling infrastructure. This method might be feasible for a relatively small city as Copenhagen but will pose a challenge for larger cities where the distance between the municipal planning unit and some parts of the city might be greater. The dependence on personal knowledge and citizen input can furthermore be problematic for neighbourhoods where the citizens do not have the same resources to make their claims heard, or where there is less of a tradition for cycling and thus less awareness of what could be improved. In such situations, a more systematic network analysis can provide an additional perspective.

Methodological Considerations

In this final section before the conclusion, I will discuss some of the limitations and assumptions for the analysis and results, along with ways in which the analysis could be adapted and extended with further research.

When assessing the results presented above, it is important to remember that the analysis is based on several core assumptions, which to some extent can limit the validity or accuracy of the results. The most central assumptions are those describing the fundamental structure of the network, namely that it is planar, undirected, and simple. As described in chapter 3, this is not entirely accurate. A network analysis based on a more detailed dataset which accounts for non-planar, parallel, and directed edges would give a higher accuracy and give a better fit between the abstracted graph and the actual, physical network.

The model of the graph furthermore does not include several factors which can have a big impact on the experience of cycling and potentially influence the cyclists' choice of route. These are for example the cost of turning, particularly left turns which requires you to cross the street twice[16], the presence and timing of traffic lights, and the state and width of bike lanes. All these variables can be incorporated into a network analysis, if the data are available, and would potentially give a more realistic view of the actual paths chosen by the city's cyclists.

The comparison of neighbourhood variations in network connectivity and density with various neighbourhood characteristics could similarly be extended and incorporate more variables, such as the distribution of businesses and industries across the city, the rate of car ownership in different neighbourhoods, and access to public transport. The comparison and correlation of neighbourhood characteristics with network variables should however be done carefully, since there is a high degree of correlation, and potentially causality, between both neighbourhood characteristics and network indices.

A smaller analytical unit than the neighbourhood might potentially also reveal a different pattern in the distribution of network variation. The analytical unit could moreover be made to incorporate connecting

[16] This way of turning left at intersections is also known as 'the Copenhagen left' and is by some considered both cumbersome and hazardous (Hembrow, 2010). See Copenhagen, 2019c for a description of this way of turning left.

bike lanes and routes in neighbouring municipalities. This could reveal issues with gaps and missing links between segments on each side of the municipal border and give a more accurate account of the number of dangling vertices.

A final consideration is the way in which the different factors are used to assign cost to the edges. In the method used for this analysis, distance is the determinant factor for the cost assigned to an edge. This means that the overall reachability of course goes down when more factors are introduced, and might make it harder to decipher the influence of the different variables compared to the influence of the length of a segment. While distance is crucial for modelling the route choices of cyclists, a network analysis without distance as a cost would to a greater extent reveal network segments with particularly good or bad cycling conditions. Similarly, it could be considered to separate the positive and negative factors used for assigning costs into two different weighting schemes. This might aid in the interpretation of the results and for example ensure that e.g. the presence of a park or water nearby does not mask that a segment simultaneously have a lot of traffic and no separated bike lane. The weighting schemes could furthermore be adapted to produce a larger difference between results from different weighting schemes, and in this way more clearly display how the network is experienced based on different preferences. This approach could potentially be extended to assigning infinite cost to edges with particularly high traffic counts or high speed, which would make it clear whether any areas become inaccessible if a cyclist is uncomfortable with this type of street. While some other network analysis of cycling infrastructure actually do exclude high stress segments[17], this does however not match the actual experience of cycling in Copenhagen, which, both in theory and practice, allows for cycling on almost all street segments, and even have bike lanes along some highways. The data used here furthermore only includes segments which officially are part of the cycling infrastructure of bike lanes and routes (present and planned), and all segments are thus considered bikeable. A network analysis based on the entire street network might however have to modify this assumption. It is in this regard important to remember that an analysis like the one performed here needs to be adapted to the local context, since assumptions made here might not hold for other places.

[17] See e.g. Zuo and Wei, 2019 or Kent and Karner, 2019.

7. Conclusion

The analysis and results presented above have all aimed to answer the research question of how a GIS-based network analysis can be used for evaluating cycling infrastructure. My objective with this question has been to firstly, explore how a network analysis can be used to describe and assess a network of cycling infrastructure at a general level, and secondly, to examine in what ways such a network analysis might also be used to identify areas and network segments that stand out, for example due to lower than average connectivity.

The results and methods applied throughout the analysis have shown that measures of connectivity and centrality can reveal the core structure and state of a network. It does this by describing to what extent e.g. the connectivity between network elements, network density, and ratio between edges and vertices make the cycling infrastructure function as a proper transportation network, that allows you to move through the city without too many constraints and detours. Connectivity and centrality indices can also be used to identify issues with low network resilience in places where a network is too reliant on a small number of network segments. The computation of such indices can furthermore highlight problems with disparities and inequalities in the network, e.g. in terms of access to cycling infrastructure, network density, and distance and reachability to or from other parts of the network.

To achieve these results, the concept of shortest path is essential, both for the computation of overall connectivity and for identifying the centrality of different network elements. The use of shortest path computations for network analysis is particularly useful when the shortest path is not only taken to mean physical distance, but instead the path of 'lowest costs', since this allows for the incorporation of a wider variety of variables that all influence the experience of cycling. It is in this regard nonetheless important to keep in mind that the shortest path is a theoretical construct, which does not necessarily reflect the actual paths of cyclists. A computation of the connectivity and centrality indices used in this analysis, but based on actual paths, would thus be an interesting idea for future research.

The use of weighted shortest paths based on a range of different variables describing each network segment have demonstrated that shortest path computations can incorporate many more variables than simply physical distance, if data are available. Adding more variables will give a different understanding of the network and potentially provide a more realistic and nuanced depiction of how a network is experienced by cyclists, who rarely only are concerned about distance. If the goal is to understand how a specific characteristic is influencing the connectivity and costs of moving through the network, a different approach than the one used here is however needed.

With that said, the analysis has clearly demonstrated the potentials of a network analysis as a tool when planning and designing better cycling infrastructure. A network analysis like the one I have conducted here can for example provide a comprehensive overview of how a network functions as a whole, which is otherwise hard to achieve for larger networks, and can identify areas with low connectivity and access. While it is straightforward to determine e.g. how many kilometres of bike lane a neighbourhood has, the length and number of bike lanes and routes do not necessarily tell you how well connected the bike lanes are, whether they connect properly to surrounding areas, how easy the area is to reach, if they are disproportionately affected by car traffic, and so on. A network analysis can therefore be used to detect otherwise overlooked areas, identify missing links, model the consequences of new extensions, and moreover provides standardised and quantifiable measurements for the quality of the network. The City of Copenhagen already has specific targets and measurements used to track and guide the development for e.g. the share of the population commuting by bicycle, satisfaction with bike lanes, safety, etc. As a network of cycling infrastructure becomes increasingly developed and established, it might be useful to similarly establish concrete goals for the network as a whole, such as e.g. a minimum network density for a neighbourhood, the maximum cost of cycling across the city, a maximum amount of dangling ends per square kilometre, etc. Such metrics can help ensure a consistent network quality across the city and prevent that some areas fall behind as the cycling infrastructure is improved and extended. Even though some of the missing links, issues with network resilience, and disparities in access and connectivity might already be known, a quantifiable metric moreover makes it easier to measure and communicate progress and improvements.

This thesis has primarily been an exploration of a method, with the aim of showing the potentials of network analysis for understanding and communicating the state of cycling infrastructure. The analysis has shown that network analysis is a highly flexible approach that can be adapted to specific needs and preferences, for example when it comes to the weighting of network segments. The results are however only as good as the underlying data, and data availability, quality, and accuracy are reoccurring challenges when it comes to planning for cycling. The analysis I have performed here could thus easily be improved if better and more accurate data were available, particularly concerning the fundamental assumptions about the network structure and directionality. Future research could also examine how to use the preferences of actual cyclists to model the weights and cost, and ideally incorporate more data about the condition and characteristics of network segments. With a longer time perspective, developing the methods, functions, and equations used here into broadly available plugins and packages that can be run with common GIS-technologies would make it easier and more accessible to perform detailed and comprehensive network analyses, and hopefully result in more comparable metrics. Finally, I have shown that the connectivity and accessibility for cyclists do not need to be deduced or assumed based on regular street networks build for motorised traffic, but that networks of cycling infrastructure can be treated as independent networks, with meaningful results.

8. References

Aby-Ryash, H. and Tamimi, A., 2015. Comparison Studies for Different Shortest Path Algorithms. *International Journal of Computers and Applications*, 14(8), p.5979-5986

Alpert, D., 2016. *Copenhagen uses this one trick to make room for bikeways on nearly every street,* viewed 1 March 2020 <https://ggwash.org/view/43010/copenhagen-uses-this-one-trick-to-make-room-for-bikeways-on-nearly-every-street>

Altaweel, M., 2018. *The Modifiable Areal Unit Problem and GIS*, viewed 01 March 2020 <https://www.gislounge.com/modifiable-areal-unit-problem-gis/>

Andersen, M., 2017. *We scored the bike networks in 299 U.S. cities – here's what we found,* viewed 01 March 2020 <https://peopleforbikes.org/blog/we-scored-the-bike-networks-in-299-u-s-cities-heres-what-we-found/>

Bisht, J., s.d. *Clustering Coefficient*, viewed 06 March 2020 <https://en.wikipedia.org/wiki/Clustering_coefficient#/media/File:Clustering_coefficient_example.svg>

Boeing, G. Urban spatial order: street network orientation, configuration, and entropy. *Applied Network Science,* 4(67)

Boisjoly, G. and El-Geneidy, A., 2016. Are we connected? Assessing bicycle network performance through directness and connectivity measures, a Montreal, Canada case study. *Paper to be presented at the 95th Annual Meeting of the Transportation Research Board.* Washington D.C., USA.

Brandes, U., and Erlebach, T., eds., 2005. *Network Analysis. Methodological Foundations*. Berlin: Springer

Bhuyan, I., Chavis, C., Nickkar, A., and Barnes, P., 2019. GIS-Based Equity Gap Analysis: Case Study of Baltimore Bike Share Program. *Urban Science,* 3(42)

City of Copenhagen, 2002. *Cykelpolitik 2002 – 2021*. Copenhagen: City of Copenhagen

City of Copenhagen, 2013. *Focus on cycling. Copenhagen Guidelines for the Design of Road Projects*. Copenhagen: City of Copenhagen

City of Copenhagen, 2016a. *CPH 2025 Climate Plan - Roadmap 2017–2020*. Copenhagen: City of Copenhagen

City of Copenhagen, 2016b. *Faktaark fra Velfærdsanalyse*. Copenhagen: City of Copenhagen

City of Copenhagen, 2016c. *Trafikken i København: Trafiktal 2010-2014*. Copenhagen: City of Copenhagen

City of Copenhagen, 2017a. *Cykelstiprioriteringsplan 2017 – 2025*. Copenhagen: City of Copenhagen

City of Copenhagen, 2017b. *Statusrapport 2017for sundhedspolitikken 2015-2025*. Copenhagen: City of Copenhagen.

City of Copenhagen, 2017c. *Bedre landingsplads til Inderhavnsbroen,* viewed 19 June 2020 <https://www.kk.dk/nyheder/bedre-landingsplads-til-inderhavnsbroen>

City of Copenhagen, 2018a. Status på København 2018. Copenhagen: City of Copenhagen

City of Copenhagen, 2018b. *Besvarelse af spørgsmål vedr. beboerlicenser og indkomster,* viewed 09 June 2020 <https://www.kk.dk/sites/default/files/brsek_-_svar_til_knud_holt_nielsen_om_indkomster_og_bilejerskab_samt_konsekvenser_af_aendret_takster_for_beboerlicenser_-_25052018.pdf>

City of Copenhagen, 2019a. *Cykelredegørelse 2019*. Copenhagen: City of Copenhagen

City of Copenhagen, 2019b. *Cykelregnskab 2018*. Copenhagen: City of Copenhagen

City of Copenhagen, 2019c. *Cycling rules in Copenhagen*, viewed 12 June 2020 <https://international.kk.dk/artikel/cycling-rules-copenhagen>

Colville-Andersen, M., 2014. *Anniversary of the Modern Copenhagen Cycle Track*, viewed 01 March 2020 <http://www.copenhagenize.com/2008/06/copenhagen-lanes-celebrate-25-years.html>

Colville-Andersen, M., 2017. The 20 Most Bike-Friendly Cities in the World, From Malmö to Montreal. *Wired,* viewed 20 February 2020 <https://www.wired.com/story/world-best-cycling-cities-copenhagenize/>

Corneliussen, C. J., 2002. Hård kritik af Ørestads udformning. *Jyllands Posten*, viewed 03 March 2020 <https://jyllands-posten.dk/indland/kbh/ECE3414383/H%C3%A5rd-kritik-af-%C3%98restads-udformning/>

Dietrich, O. W., 2009. *Cykelsti,* viewed 01 March 2020 <http://denstoredanske.dk/It,_teknik_og_naturvidenskab/Teknik/Vejbygning/cykelsti>

Dill, J., 2004. Measuring Network Connectivity for Bicycling and Walking. *In 84th Annual Meeting of the Transportation Research Board.* Washington, DC, paper 04-001550

Ejlertsen, M., 2020. Akademikere styrer den offentlige debat. *Magisterbladet,* viewed 10 June 2020 <https://www.magisterbladet.dk/aktuelt/2020/marts/akademikere-styrer-den-offentlige-debat>

Ekeroth, T., 2019. København vokser med 100.000 de næste 12 år, viewed 20 February 2020 <http://www.danskekommuner.dk/Nyhedsarkiv/2019/Marts/12/Kobenhavn-vokser-med-100000-de-naste-12-ar/>

ESRI, 2016. What is a Shapefile?, viewed 12 March 2020 <https://desktop.arcgis.com/en/arcmap/10.3/manage-data/shapefiles/what-is-a-shapefile.html>

FAOD, 2019. *Lille Langebro hitter hos cyklister og fodgængere*, viewed 19 June 2020 <https://www.faod.dk/lille-langebro-hitter-hos-cyklister-og-fodgaengere/>

Furth, P. and Noursalehi, P., 2015. Evaluating the Connectivity of a Bicycling Network. *In 94th Annual Meeting of the Transportation Research Board*. Washington, DC

Harrie, L. (ed), 2014a. *Geografisk informationsbehandling. Teori, metoder och tillämpningar*. Lund: Studentlitteratur

Harrie, L. 2014b. *Lecture Notes in GIS Algorithms*. Lund: Lund University

Hembrow, D., 2010. *The "Copenhagen Left" and merging of cyclists with cars turning right: Dangerous and inconvenient junction design in Denmark*, viewed 12 June 2020 <http://www.aviewfromthecyclepath.com/2010/07/not-really-so-great-cycle-path-design.html>

Henderson, J. and Gulsrud, N. M., 2019. *Street Fights in Copenhagen. Bicycle and Car Politics in a Green Mobility* City. London: Routledge.

Herby, J. and Friis, A., 2014. *Cykelregnskab for Region Hovedstaden*. Holte: Incentive

Jørgensen, L., 2015. *Åbuen*, viewed 13 April, 2020 <https://commons.wikimedia.org/wiki/File:%C3%85buen_01.JPG>

Kent, M. and Karner, A., 2019. Prioritizing low-stress and equitable
bicycle networks using neighbourhood-based accessibility measures. *International Journal of Sustainable Transportation,* 13(2), p.100-110

Kirschbaum, E. 2019. Copenhagen has taken bicycle commuting to a whole new level. *LA Times,* viewed 20 February 2020 <https://www.latimes.com/world-nation/story/2019-08-07/copenhagen-has-taken-bicycle-commuting-to-a-new-level>

Køhler, S., 2020. Får København en ny cykelbro mellem Refshaleøen og Kastellet?, *Magasinet KBH,* viewed 21 May 2020 <https://www.magasinetkbh.dk/broforbindelse-refshaleoen-kastel>

Lin, J., 2017. *Spatial analysis and modeling of urban transportation networks*. Doctoral thesis. Stockholm: KTH Royal Institute of Technology

Lin, J. and Ban, Y., 2017. Comparative Analysis on Topological Structures of Urban Street Networks. *ISPRS International Journal of Geo-Information*, 6, article 295

Litman, T., 2020. *Evaluating Accessibility for Transport Planning. Measuring People's Ability to Reach Desired Goods and Activities*. Victoria: Victoria Transport Policy Institute.

Lucas-Smith, M., 2019. *Is the OSM data model creaking?*, viewed 08 March 2020 <https://www.cyclestreets.net/blog/2019/09/22/sotm2019/>

Mansourian, A. and Harrie, L., 2013. *Lecture Notes in Geographical Databases*. Lund: Lund University

Mikiewicz, D., Mackiewicz, M. and Nycz, T., 2017. *Mastering PostGIS*. Birmingham: Packt Publishing

Ministry of Health, 2014. *Ny undersøgelse: Stor ulighed i sundhed,* viewed 10 June 2020 <https://www.sum.dk/Aktuelt/Nyheder/Forebyggelse/2014/Marts/Stor-ulighed-i-sundhed.aspx>

Neo4J, s.d. *The Closeness Centrality algorithm,* viewed 20 May 2020 <https://neo4j.com/docs/graph-algorithms/current/labs-algorithms/closeness-centrality/>

NetworkX, 2015. *edge_betweenness_centrality*, viewed 06 March 2020 <https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.algorithms.centrality.edge_betweenness_centrality.html>

Niedderer, K., Clune, S., and Ludden, G. (eds), 2017. *Design for Behaviour Change: Theories and practices of designing for change*. London: Routledge

Nielsen, H., 2019. Hvad betyder en uddannelse, hvis vi ikke har en god planet at leve på? *Jyllands Posten,* viewed 10 June 2020 <https://jyllands-posten.dk/indland/ECE11417018/hvad-betyder-en-uddannelse-hvis-vi-ikke-har-en-god-planet-at-leve-paa/>

pgRouting, 2017. *pgRouting Manual. Release 2.2.4 (pgrouting-2.2.4),* viewed 08 April 2020 < https://docs.pgrouting.org/pdf/en/pgRoutingDocumentation-2.2.4.pdf>

Rodrigue, J., Comtois, C. and Slack, B., 2009. *The Geography of Transport Systems*. 3rd ed. New York: Routledge

Samson, J., 2019. Københavns udvikling. *Faktalink*, viewed 01 March 2020 <https://faktalink.dk/titelliste/kobenhavns-byudvikling>

Schoner, J. and Levinson, D. 2014. The missing link. Bicycle Infrastructure Networks and Ridership in 74 US Cities. *Transportation,* 41(6), p.1187-1204

Shanmukhappa, T., Ho, I. W., and Tse, C. H, 2018. Spatial analysis of bus transport networks using network theory. *Physica A*, 502 p.295–314

Sommer, C., 2014. Shortest-Path Queries in Static Networks. *ACM Computing Surveys*, 46(4), article 45

Stan, 2019. *Statistical vs. Computational Efficiency*, viewed 07 March 2020 <https://mc-stan.org/docs/2_22/stan-users-guide/statistical-vs-computational-efficiency.html>

Statistics Denmark, 2005. *Privatøkonomi og uddannelse*. Copenhagen: Statistics Denmark

Statistics Denmark, 2020a. *Population at the first day of the quarter by region and time,* viewed 02 March 2020 <https://www.statbank.dk/statbank5a/SelectVarVal/saveselections.asp>

Statistics Denmark, 2020b. *Area 1. January by time and region*, viewed 02 March 2020 <https://www.statbank.dk/ARE207>

Statistics Denmark, 2020c. *Population 1. January by urban and rural areas and time,* viewed 02 March 2020 <https://www.statistikbanken.dk/BY1>

Topjian, T., 2019. Why We Need to Dream Bigger Than Bike Lanes. *Citylab,* viewed 20 February 2020 <https://www.citylab.com/perspective/2019/10/micromobility-urban-design-car-free-infrastruture-futurama/600163/>

Tresidder, M., 2005. *Using GIS to Measure Connectivity: An Exploration of Issues.* Field Area Paper. Portland: Portland State University

Vassi, A. and Vlastos, T., 2014. A review and critical assessment of cycling infrastructure across Europe. In: Marchettini, N., Brebbia, C., Pulselli, R. and Bastianoni, S. (eds), 2014. *The Sustainable City IX. Urban Regeneration and Sustainability*. Southampton: WIT Press, p.757-768

Vilhelmsen, J., 2016. *De længst uddannede lever 6 år mere end de ufaglærte*. Copenhagen: Arbejderbevægelsens Erhvervsråd

Wagner, D., 2003. *Discrete Mathematics for CS,* viewed 06 March 2020 <https://people.eecs.berkeley.edu/~daw/teaching/cs70-f03/Notes/hypercube.pdf>

Xie, F. and Levinson, D., 2007. Measuring the Structure of Road Networks. *Geographical Analysis*, 39, p.336–356

Zuo, T., and Wei, H., 2019. Bikeway prioritization to increase bicycle network connectivity and bicycle-transit connection: A multi-criteria decision analysis approach. *Transportation Research*, Part A 129, p.52–71

9. Appendices

Appendix I: Neighbourhood Centrality Indices

Table 11. Neighbourhood Centrality Indices based on $W_{efficiency}$

Neighbourhood	Minimum Stress Centrality	Average Stress Centrality	Maximum Stress Centrality	Minimum Straightness Centrality	Average Straightness Centrality	Maximum Straightness Centrality	Minimum Closeness Centrality	Average Closeness Centrality	Maximum Closeness Centrality
Amager Øst	46	2780	14979	0.649	0.782	0.842	0.108	0.149	0.189
Amager Vest	0	5384	31224	0.585	0.748	0.834	0.091	0.15	0.216
Bispebjerg	94	4112	28224	0.716	0.804	0.843	0.132	0.162	0.195
Brønshøj-Husum	0	2411	9339	0.712	0.795	0.83	0.095	0.122	0.163
Frederiksberg	0	5053	36887	0.705	0.774	0.822	0.147	0.196	0.23
Indre By	0	6404	45640	0.584	0.775	0.826	0.128	0.206	0.238
Nørrebro	0	7174	45640	0.703	0.794	0.823	0.177	0.209	0.237
Østerbro	21	3505	17657	0.603	0.762	0.818	0.1	0.158	0.204
Valby	0	3041	14503	0.646	0.746	0.79	0.103	0.146	0.189
Vanløse	8	3046	15070	0.682	0.783	0.83	0.106	0.151	0.19
Vesterbro-Kongens Enghave	59	5624	41184	0.608	0.745	0.79	0.117	0.188	0.231

Table 12. Neighbourhood Centrality Indices based on W_{safety}

Neighbourhood	Minimum Stress Centrality	Average Stress Centrality	Maximum Stress Centrality	Minimum Straightness Centrality	Average Straightness Centrality	Maximum Straightness Centrality	Minimum Closeness Centrality	Average Closeness Centrality	Maximum Closeness Centrality
Amager Øst	49	2619	11233	0.631	0.744	0.84	0.106	0.147	0.186
Amager Vest	0	5408	28602	0.579	0.742	0.832	0.09	0.149	0.213
Bispebjerg	90	3953	25238	0.71	0.792	0.84	0.13	0.159	0.194
Brønshøj-Husum	0	2447	9668	0.709	0.792	0.828	0.095	0.122	0.163
Frederiksberg	0	5307	35949	0.704	0.77	0.819	0.146	0.195	0.229
Indre By	0	6408	44270	0.579	0.766	0.816	0.126	0.204	0.237
Nørrebro	0	6852	44270	0.695	0.788	0.82	0.176	0.207	0.236
Østerbro	0	3536	17791	0.592	0.746	0.803	0.098	0.154	0.202
Valby	0	3090	13626	0.643	0.736	0.781	0.103	0.144	0.187
Vanløse	9	3036	12557	0.679	0.779	0.827	0.101	0.15	0.189
Vesterbro-Kongens Enghave	0	5596	36940	0.606	0.74	0.786	0.117	0.186	0.229

Table 13. Neighbourhood Centrality Indices based on $W_{distance}$

Neighbourhood	Minimum Stress Centrality	Average Stress Centrality	Maximum Stress Centrality	Minimum Straightness Centrality	Average Straightness Centrality	Maximum Straightness Centrality	Minimum Closeness Centrality	Average Closeness Centrality	Maximum Closeness Centrality
Amager Øst	38	3018	17429	0.678	0.824	0.875	0.113	0.159	0.203
Amager Vest	0	4726	30242	0.626	0.81	0.874	0.094	0.164	0.235
Bispebjerg	77	3359	14506	0.758	0.839	0.87	0.138	0.171	0.208
Brønshøj-Husum	6	2144	8874	0.804	0.839	0.868	0.107	0.13	0.173
Frederiksberg	56	5099	23537	0.747	0.824	0.855	0.158	0.211	0.247
Indre By	0	7472	33378	0.614	0.817	0.878	0.135	0.22	0.256
Nørrebro	26	6311	23985	0.728	0.832	0.861	0.192	0.222	0.253
Østerbro	0	3461	13467	0.639	0.803	0.848	0.105	0.169	0.215
Valby	0	2779	13520	0.709	0.818	0.858	0.113	0.161	0.2
Vanløse	120	3125	12534	0.712	0.832	0.869	0.12	0.162	0.199
Vesterbro-Kongens Enghave	20	5270	23245	0.669	0.794	0.84	0.129	0.202	0.25

Appendix II: Pairwise Correlation of Network Indices and Neighbourhood Characteristics

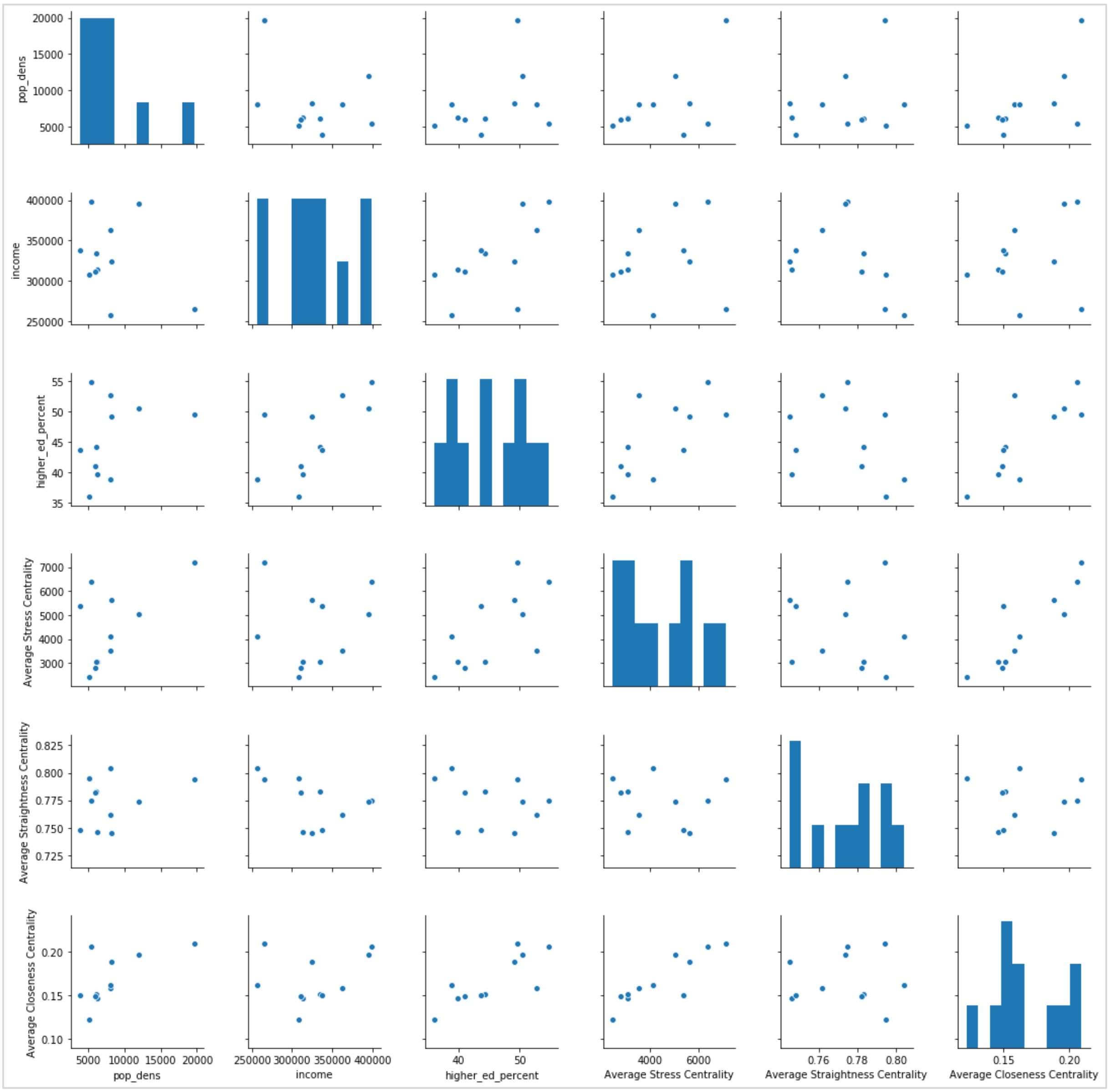

Figure 53. Seaborn pairwise comparison plot based on $W_{efficiency}$

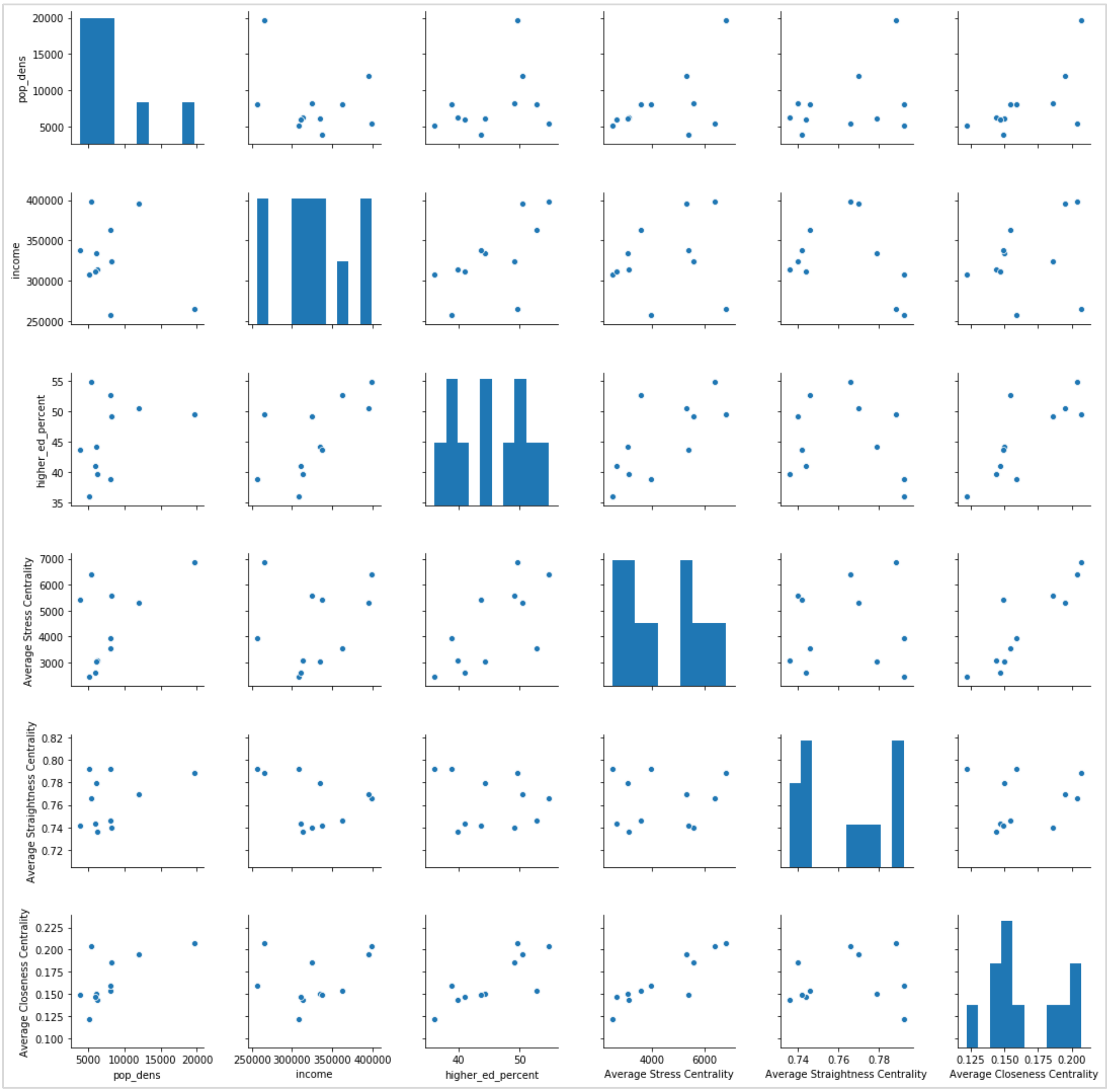

Figure 54. Seaborn pairwise comparison plot based on W_{safety}

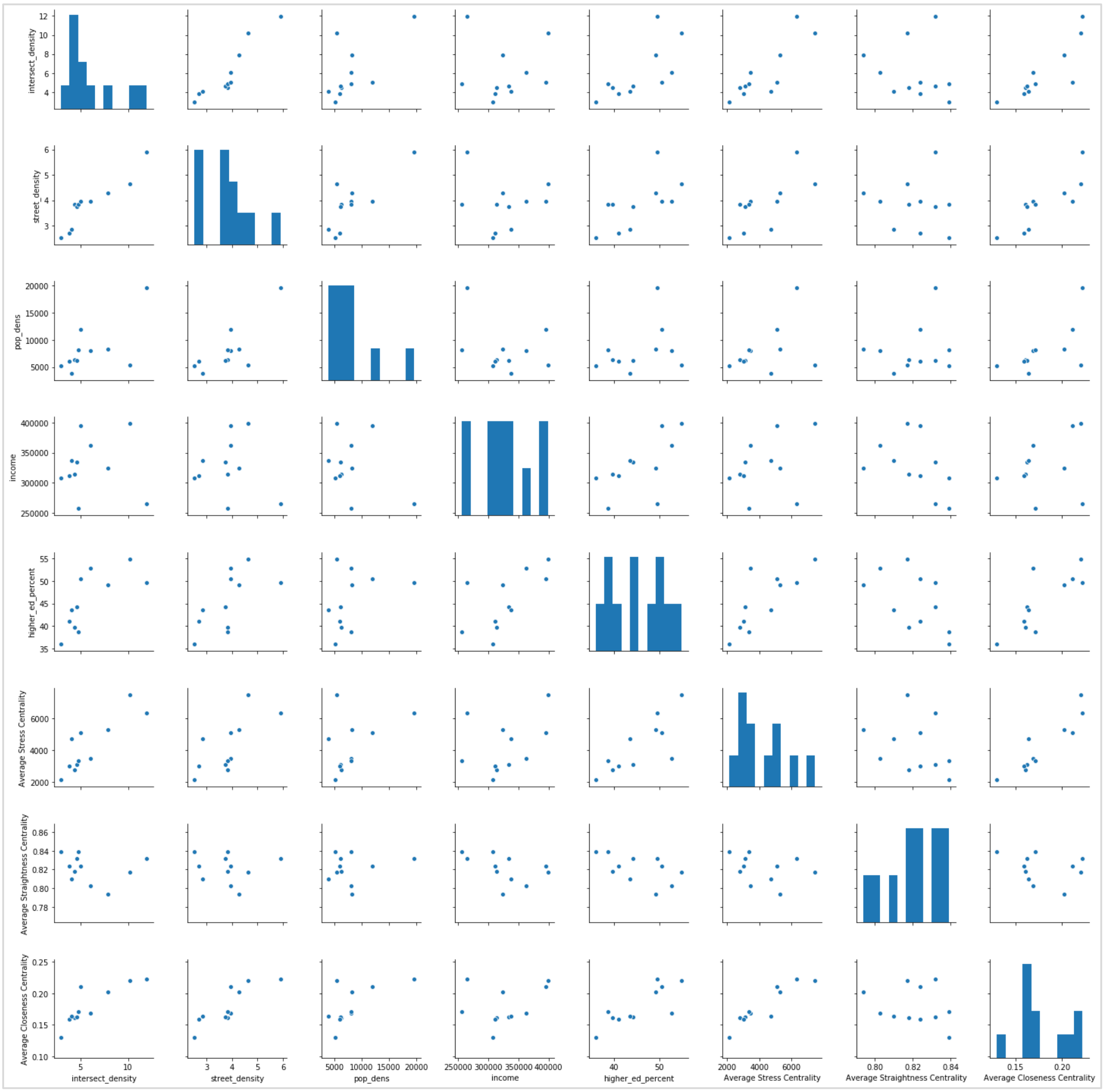

Figure 55. Seaborn pairwise comparison plot based on $W_{distance}$

Appendix III: Pre-processing of Network Topology with PostGIS

```sql
CREATE DATABASE network_analysis
ENCODING = UTF8
WITH OWNER = postgres;

-- Table with age distribution and population density --

CREATE TABLE age_density
(n_name character varying(50) NOT NULL UNIQUE PRIMARY KEY,
Total_0_17 float,
Percent_0_17 float,
Total_18_24 float,
Percent_18_24 float,
Total_25_64 float,
Percent_25_64 float,
Total_65_99 float,
Percent_65_99 float,
Total_pop float,
Area float,
Pop_dens float );


\copy age_density FROM 'Neighbourhoods_age_density.csv' DELIMITER ',' CSV HEADER;

-- Table with income and educational levels

CREATE TABLE income_education (n_name character varying(50) NOT NULL UNIQUE PRIMARY KEY,
income integer,
elementary integer,
percent_elementary float,
high_school integer,
percent_highschool float,
vocational integer,
percent_vocational float,
short_further integer,
percent_short float,
medium_higher integer,
percent_medium float,
bachelor integer,
percent_bachelor float,
long_higher integer,
percent_long float,
total_16_66 integer
);

\copy income_education FROM ‘Neighbourhoods_income_education.csv' DELIMITER ',' CSV HEADER;


-- Shapefiles loaded using PostGIS Bundle
-- Updating spatial reference for shapefiles

select UpdateGeometrySRID('all_bikelanes', 'geom', 25832);
select UpdateGeometrySRID('all_roads', 'geom', 25832);
select UpdateGeometrySRID('traffic_counts', 'geom', 25832);
select UpdateGeometrySRID('road_surface', 'geom', 25832);
select UpdateGeometrySRID('neighbourhoods', 'geom', 25832)
select UpdateGeometrySRID('parks', 'geom', 25832);
select UpdateGeometrySRID('water', 'geom', 25832);

-- Adding Frb. to Neighbourhoods layer--

CREATE TABLE single_neighs AS
SELECT gid, navn, areal_m2, area_km2, neigh_number, (ST_DUMP(geom)).geom::geometry(Polygon,25832) AS geom FROM ne
ighbourhoods;

CREATE TABLE frb (like single_neighs);
INSERT INTO frb(geom)(SELECT St_MakePolygon(st_interiorringn(st_union(geom),1)) AS geom FROM neighbourhoods);

-UPDATE frb SET gid = 11, navn = 'Frederiksberg', neigh_number = 11;

INSERT INTO single_neighs SELECT * FROM frb;

ALTER TABLE single_neighs RENAME TO neighbourhoods;

-- Adding roadid to table traffic counts
ALTER TABLE traffic_counts ADD COLUMN roadid VARCHAR;
UPDATE traffic_counts SET road_id = road_surface.vejid FROM road_surface
WHERE ST_Intersects(traffic_counts.geom, road_surface.geom);

-- Saving bike lanes as separate table
CREATE TABLE regular_bikelanes AS SELECT * FROM all_bikelanes_old WHERE kategori = 'Bike lane';

-- Assigning road_id to bikelanes
UPDATE bikelanes SET vejid = all_roads.road_id FROM all_roads
WHERE ST_Intersects(bikelanes.topogeom, all_roads.geom) AND vejid IS NULL;
```

```sql
-- Saving Cycle Superhighways and Green Cycle Routes as separate table --
CREATE TABLE super_green AS SELECT * FROM all_bikelanes_old WHERE kategori IN ('Super','Green');

-- layer with bikelanes edited in arcmap and qgis
-- all segments joined and then split at intersection
-- spatial join with most overlapping road surface
-- reloaded as bikelanes


-- Specifying bikelanes which are part of green cycle routes of cycle superhighways
ALTER TABLE bikelanes ADD COLUMN green VARCHAR;
ALTER TABLE bikelanes ADD COLUMN super VARCHAR;

-- super_green has been split into singleparts using QGIS
CREATE TABLE super_buffer AS
SELECT ST_Union(ST_Buffer(geom,15)::geometry(Polygon,25832)) AS geom
FROM super_green_single WHERE super_green_single.kategori = 'Super';

WITH regular_super AS
(SELECT * FROM bikelanes AS bl, super_buffer AS sb WHERE ST_Within(bl.geom, sb.geom))
UPDATE bikelanes SET super = 'super' from regular_super rs WHERE bikelanes.id = rs.id;

CREATE TABLE green_buffer AS
SELECT ST_Union(ST_Buffer(geom,15)::geometry(Polygon,25832)) AS geom
FROM super_green_single WHERE super_green_single.kategori = 'Green';

WITH regular_green AS
(SELECT * FROM bikelanes AS bl, green_buffer AS gb WHERE ST_Intersects(bl.geom, gb.geom))
UPDATE bikelanes SET green = 'g' FROM regular_green rg WHERE bikelanes.id = rg.id;

-- Specifying bikelanes which are part of closer to the water or parks
ALTER TABLE bikelanes ADD COLUMN park VARCHAR;
ALTER TABLE bikelanes ADD COLUMN water VARCHAR;

WITH bikelanes_water AS
(SELECT bl.gid FROM bikelanes AS bl, water AS w WHERE ST_DWithin(bl.topogeom, w.geom, 15))
UPDATE bikelanes SET water = 'w'
FROM bikelanes_water bw WHERE bikelanes.gid = bw.gid;

WITH bikelanes_park AS
(SELECT bl.gid FROM bikelanes AS bl, parks AS p WHERE ST_DWithin(bl.topogeom, p.geom, 15))
UPDATE bikelanes SET park = 'p'
```

```sql
FROM bikelanes_park bp WHERE bikelanes.gid = bp.gid;

-- Checking for non-valid and non-simple geometries
SELECT * FROM bikelanes WHERE ST_IsValid(geom) =  FALSE;

SELECT * FROM bikelanes WHERE ST_IsSimple(geom) = FALSE;

-- Deleting null geometries from regular bikelanes
DELETE FROM bikelanes WHERE geom IS NULL;


---Installing the topology extension
CREATE EXTENSION postgis_topology;

-- Creating new empty topology
SELECT topology.CreateTopology('network_topology', 25832, 1, FALSE);

-- Creating new column for TopoGeometry
SELECT topology.AddTopoGeometryColumn('network_topology','public','bikelanes','topogeom','MULTILINE');

-- Filling TopoGeometry column with snapping threshold 1 meter --
DO $$DECLARE r record;
BEGIN
  FOR r IN SELECT * FROM bikelanes LOOP
    BEGIN
      UPDATE bikelanes SET topogeom = topology.toTopoGeom(geom,'network_topology',1,2)
      WHERE id = r.id;
    EXCEPTION
      WHEN OTHERS THEN
        RAISE WARNING 'Loading of record % failed: %', r.id, SQLERRM;
    END;
  END LOOP;
END$$;

-- Validating topology
SELECT * FROM topology.ValidateTopology('network_topology');

--Topological errors where fixed using e.g.:
SELECT ST_NewEdgeHeal('network_topology',186,610);

-- Identify unconnected edges --
SELECT * FROM network_topology.edge_data WHERE edge_id = abs_next_left_edge AND edge_id = abs_next_right_edge;
```

Appendix IV: Factors & Corresponding Weights

Weighting Scheme for Efficiency & Separated Bike Lanes ($W_{efficiency}$)

Speed	Weight
0	-20
15	-20
30	-15
40	-10
50[18]	0
60	20
70	25
90	30

Class	Weight
Highway	30
Distribution street	10
Regional street	20
Shopping street	0
Neighbourhood street	0
Unknown	0

Status	Weight
Path[19]	0
Private street	0
Main road	10
Municipal street	0

Attributes	Weight
Green Cycle Route	-15
Cycle Superhighways	-25
Park	-10
Water	-10
Bike lane	- 40

Traffic counts, cars (thousands)	Weight
0	0
0.1 - 3	0
3 - 10	5
10 - 17	10
17 - 27	15
27 - 43	20
43 - 67	25
67 - 89	30

Traffic counts, trucks (thousands)	Weight
0.001 – 0.1	5
0.1 – 0.5	10
0.5 – 1.2	15
1.2 – 1.9	20

[18] Default speed limit in urban areas in Denmark.
[19] Paths are in some cases significantly more bike friendly than streets, for example in the base of separate bike lanes constructed away from the street. Other paths are however not exclusively used by cyclists and might have a surface which is less suitable for bike.

Weighting Scheme for Lower Traffic Speed & Fewer Cars (W_{safety})

Speed	Weight
0	-30
15	-20
30	-15
40	-10
50	0
60	25
70	30
90	40

Class	Weight
Highway	30
Distribution street	10
Regional street	20
Shopping street	0
Neighbourhood street	0
Unknown	0

Status	Weight
Path	- 10
Private street	0
Main road	10
Municipal street	0

Attributes	Weight
Green Cycle Route	-15
Cycle Superhighways	-15
Park	-10
Water	-10
Bike lane	- 20

Traffic counts, cars (thousands)	Weight
0	0
0.1 - 3	0
3 - 10	5
10 - 17	10
17 - 27	20
27 - 43	30
43 - 67	35
67 - 89	40

Traffic counts, trucks (thousands)	Weight
0.001 – 0.1	10
0.1 – 0.5	15
0.5 – 1.2	20
1.2 – 1.9	25

Appendix V: Adding Weights to Network Segments

```sql
--This scripts weighs the different network segments based on their attributes

ALTER TABLE bikelanes ADD COLUMN total_weight_safety NUMERIC;
ALTER TABLE bikelanes ADD COLUMN total_weight_efficiency NUMERIC;

ALTER TABLE network_topology.edge_data ADD COLUMN length_ NUMERIC;
UPDATE network_topology.edge_data SET length_ = ST_Length(geom);

CREATE TABLE safety_weights AS
SELECT gid, vejid FROM bikelanes;

-- Update safety_weights based on class
ALTER TABLE safety_weights ADD COLUMN weight_class NUMERIC;
UPDATE safety_weights SET weight_class = 0;
UPDATE safety_weights SET weight_class = weight_class + 30 FROM all_roads
WHERE vejid = road_id AND class = 'Highway';

UPDATE safety_weights SET weight_class = weight_class + 10
FROM all_roads WHERE vejid = road_id AND class = 'Distribution street';

UPDATE safety_weights SET weight_class = weight_class + 20
FROM all_roads WHERE vejid = road_id AND class = 'Regional street';

-- Update safety_weights based on road status
ALTER TABLE safety_weights ADD COLUMN weight_status NUMERIC;
UPDATE safety_weights SET weight_status = 0;
UPDATE safety_weights SET weight_status = weight_status + 10
FROM all_roads WHERE vejid = road_id AND status = 'Main road';
UPDATE safety_weights SET weight_status = weight_status - 10
FROM all_roads WHERE vejid = road_id AND status = 'Path';

-- Update safety_weights based on speed
ALTER TABLE safety_weights ADD COLUMN weight_speed NUMERIC;
UPDATE safety_weights SET weight_speed = 0;
UPDATE safety_weights SET weight_speed = weight_speed - 30
FROM all_roads WHERE vejid = road_id AND speed = 0;

UPDATE safety_weights SET weight_speed = weight_speed - 20
FROM all_roads WHERE vejid = road_id AND speed = 15;

UPDATE safety_weights SET weight_speed = weight_speed - 15
```

```sql
FROM all_roads WHERE vejid = road_id AND speed = 30;

UPDATE safety_weights SET weight_speed = weight_speed - 10
FROM all_roads WHERE vejid = road_id AND speed = 40;

UPDATE safety_weights SET weight_speed = weight_speed + 25
FROM all_roads WHERE vejid = road_id AND speed = 60;

UPDATE safety_weights SET weight_speed = weight_speed + 30
FROM all_roads WHERE vejid = road_id AND speed = 70;

UPDATE safety_weights SET weight_speed = weight_speed + 40
FROM all_roads WHERE vejid = road_id AND speed = 90;


-- Update safety_weights based on traffic counts (cars)
ALTER TABLE safety_weights ADD COLUMN weight_cars NUMERIC;
UPDATE safety_weights SET weight_cars = 0;
UPDATE safety_weights SET weight_cars = weight_cars + 5
FROM traffic_counts WHERE vejid = road_id AND cars_weekday <= 10000 and cars_weekday >= 3000;

UPDATE safety_weights SET weight_cars = weight_cars + 10
FROM traffic_counts WHERE vejid = road_id AND cars_weekday <= 17000 and cars_weekday > 10000;

UPDATE safety_weights SET weight_cars = weight_cars + 20
FROM traffic_counts WHERE vejid = road_id AND cars_weekday <= 27000 and cars_weekday > 17000;

UPDATE safety_weights SET weight_cars = weight_cars + 30
FROM traffic_counts WHERE vejid = road_id AND cars_weekday <= 43000 and cars_weekday > 27000;

UPDATE safety_weights SET weight_cars = weight_cars + 35
FROM traffic_counts WHERE vejid = road_id AND cars_weekday <= 67000 and cars_weekday > 43000;

UPDATE safety_weights SET weight_cars = weight_cars + 40
FROM traffic_counts WHERE vejid = road_id AND cars_weekday > 67000;

-- Update safety_weights based on traffic counts (trucks)
ALTER TABLE safety_weights ADD COLUMN weight_trucks NUMERIC;
UPDATE safety_weights SET weight_trucks = 0;
UPDATE safety_weights SET weight_trucks = weight_trucks + 10
FROM traffic_counts WHERE vejid = road_id AND trucks <= 100 and trucks >= 1;

UPDATE safety_weights SET weight_trucks = weight_trucks + 15
```

```sql
FROM traffic_counts WHERE vejid = road_id AND trucks <= 500 and trucks > 100;

UPDATE safety_weights SET weight_trucks = weight_trucks + 20
FROM traffic_counts WHERE vejid = road_id AND trucks <= 1200 and trucks > 500;

UPDATE safety_weights SET weight_trucks = weight_trucks + 25
FROM traffic_counts WHERE vejid = road_id AND trucks <= 1900 and trucks > 1200;

-- Update safety_weights based on proximity to parks or water
ALTER TABLE safety_weights ADD COLUMN weight_recreational NUMERIC;
UPDATE safety_weights SET weight_recreational = 0;
UPDATE safety_weights SET weight_recreational = weight_recreational -10
FROM bikelanes WHERE bikelanes.gid =  safety_weights.gid AND park = 'p';
UPDATE safety_weights SET weight_recreational = weight_recreational -10
FROM bikelanes WHERE bikelanes.gid =  safety_weights.gid AND water = 'w';

-- Update safety_weights based on Cycle Superhighways or Green Cycle Routes
ALTER TABLE safety_weights ADD COLUMN weight_route NUMERIC;
UPDATE safety_weights SET weight_route = 0;
UPDATE safety_weights SET weight_route = weight_route - 15
FROM bikelanes WHERE bikelanes.gid =  safety_weights.gid AND green = 'g';
UPDATE safety_weights SET weight_route = weight_route - 15
FROM bikelanes WHERE bikelanes.gid =  safety_weights.gid AND super = 'super';

-- Update safety_weights based on whether there is a physical bikelane
ALTER TABLE safety_weights ADD COLUMN weight_bikelane NUMERIC;
UPDATE safety_weights SET weight_bikelane = 0;
UPDATE safety_weights SET weight_bikelane = weight_bikelane - 20
FROM bikelanes WHERE bikelanes.gid = safety_weights.gid and bikelane = 'yes';

-- Summing total weight
ALTER TABLE safety_weights ADD COLUMN total_weight_safety NUMERIC;
UPDATE safety_weights SET total_weight_safety =
weight_class + weight_status + weight_speed + weight_cars + weight_trucks +
weight_recreational + weight_route;

-- Adding total weight to table bikelanes
UPDATE bikelanes SET total_weight_safety = safety_weights.total_weight_safety
FROM safety_weights where bikelanes.gid = safety_weights.gid;

-- Creating second table for weights
CREATE TABLE efficiency_weights AS
SELECT gid, vejid FROM bikelanes;
```

```sql
-- Update efficiency_weights based on class
ALTER TABLE efficiency_weights ADD COLUMN weight_class NUMERIC;
UPDATE efficiency_weights SET weight_class = 0;
UPDATE efficiency_weights SET weight_class = weight_class + 30 FROM all_roads
WHERE vejid = road_id AND class = 'Highway';

UPDATE efficiency_weights SET weight_class = weight_class + 10
FROM all_roads WHERE vejid = road_id AND class = 'Distribution street';

UPDATE efficiency_weights SET weight_class = weight_class + 20
FROM all_roads WHERE vejid = road_id AND class = 'Regional street';

-- Update efficiency_weights based on road status
ALTER TABLE efficiency_weights ADD COLUMN weight_status NUMERIC;
UPDATE efficiency_weights SET weight_status = 0;
UPDATE efficiency_weights SET weight_status = weight_status + 10
FROM all_roads WHERE vejid = road_id AND status = 'Main road';

-- Update efficiency_weights based on speed
ALTER TABLE efficiency_weights ADD COLUMN weight_speed NUMERIC;
UPDATE efficiency_weights SET weight_speed = 0;
UPDATE efficiency_weights SET weight_speed = weight_speed - 30
FROM all_roads WHERE vejid = road_id AND speed = 0;

UPDATE efficiency_weights SET weight_speed = weight_speed - 20
FROM all_roads WHERE vejid = road_id AND speed = 15;

UPDATE efficiency_weights SET weight_speed = weight_speed - 15
FROM all_roads WHERE vejid = road_id AND speed = 30;

UPDATE efficiency_weights SET weight_speed = weight_speed - 10
FROM all_roads WHERE vejid = road_id AND speed = 40;

UPDATE efficiency_weights SET weight_speed = weight_speed + 20
FROM all_roads WHERE vejid = road_id AND speed = 60;

UPDATE efficiency_weights SET weight_speed = weight_speed + 25
FROM all_roads WHERE vejid = road_id AND speed = 70;

UPDATE efficiency_weights SET weight_speed = weight_speed + 30
FROM all_roads WHERE vejid = road_id AND speed = 90;
```

```sql
-- Update efficiency_weights based on traffic counts (cars)
ALTER TABLE efficiency_weights ADD COLUMN weight_cars NUMERIC;
UPDATE efficiency_weights SET weight_cars = 0;
UPDATE efficiency_weights SET weight_cars = weight_cars + 5
FROM traffic_counts WHERE vejid = road_id AND cars_weekday <= 10000 and cars_weekday >= 3000;

UPDATE efficiency_weights SET weight_cars = weight_cars + 10
FROM traffic_counts WHERE vejid = road_id AND cars_weekday <= 17000 and cars_weekday > 10000;

UPDATE efficiency_weights SET weight_cars = weight_cars + 15
FROM traffic_counts WHERE vejid = road_id AND cars_weekday <= 27000 and cars_weekday > 17000;

UPDATE efficiency_weights SET weight_cars = weight_cars + 20
FROM traffic_counts WHERE vejid = road_id AND cars_weekday <= 43000 and cars_weekday > 27000;

UPDATE efficiency_weights SET weight_cars = weight_cars + 25
FROM traffic_counts WHERE vejid = road_id AND cars_weekday <= 67000 and cars_weekday > 43000;

UPDATE efficiency_weights SET weight_cars = weight_cars + 30
FROM traffic_counts WHERE vejid = road_id AND cars_weekday > 67000;


-- Update efficiency_weights based on traffic counts (trucks)
ALTER TABLE efficiency_weights ADD COLUMN weight_trucks NUMERIC;
UPDATE efficiency_weights SET weight_trucks = 0;
UPDATE efficiency_weights SET weight_trucks = weight_trucks + 5
FROM traffic_counts WHERE vejid = road_id AND trucks <= 100 and trucks >= 1;

UPDATE efficiency_weights SET weight_trucks = weight_trucks + 10
FROM traffic_counts WHERE vejid = road_id AND trucks <= 500 and trucks > 100;

UPDATE efficiency_weights SET weight_trucks = weight_trucks + 15
FROM traffic_counts WHERE vejid = road_id AND trucks <= 1200 and trucks > 500;

UPDATE efficiency_weights SET weight_trucks = weight_trucks + 20
FROM traffic_counts WHERE vejid = road_id AND trucks <= 1900 and trucks > 1200;

-- Update efficiency_weights based on proximity to parks or water
ALTER TABLE efficiency_weights ADD COLUMN weight_recreational NUMERIC;
UPDATE efficiency_weights SET weight_recreational = 0;
UPDATE efficiency_weights SET weight_recreational = weight_recreational -10
FROM bikelanes WHERE bikelanes.gid =  efficiency_weights.gid AND park = 'p';
UPDATE efficiency_weights SET weight_recreational = weight_recreational -10
```

```sql
FROM bikelanes WHERE bikelanes.gid =  efficiency_weights.gid AND water = 'w';

-- Update efficiency_weights based on uHighways or Green Cycle Routes
ALTER TABLE efficiency_weights ADD COLUMN weight_route NUMERIC;
UPDATE efficiency_weights SET weight_route = 0;
UPDATE efficiency_weights SET weight_route = weight_route -15
FROM bikelanes WHERE bikelanes.gid =  efficiency_weights.gid AND green = 'g';
UPDATE efficiency_weights SET weight_route = weight_route -25
FROM bikelanes WHERE bikelanes.gid =  efficiency_weights.gid AND super = 'super';

-- Update efficiency_weights based on whether there is a physical bikelane
ALTER TABLE efficiency_weights ADD COLUMN weight_bikelane NUMERIC;
UPDATE efficiency_weights SET weight_bikelane = 0;
UPDATE efficiency_weights SET weight_bikelane = weight_bikelane - 40
FROM bikelanes WHERE bikelanes.gid = efficiency_weights.gid and bikelane = 'yes';

-- Summing total weight
ALTER TABLE efficiency_weights ADD COLUMN total_weight_efficiency NUMERIC;
UPDATE efficiency_weights SET total_weight_efficiency =
weight_class + weight_status + weight_speed + weight_cars + weight_trucks +
weight_recreational + weight_route;

-- Adding total weight to table bikelanes
UPDATE bikelanes SET total_weight_efficiency =
efficiency_weights.total_weight_efficiency
FROM efficiency_weights where bikelanes.gid = efficiency_weights.gid;

-- Making sure that there are no negative safety_weights
SELECT MIN(total_weight_safety) FROM bikelanes;

-- The lowest weight is -90
UPDATE bikelanes SET total_weight_safety = total_weight_safety + 90;

-- Making sure that there are no negative efficiency_weights
SELECT MIN(total_weight_efficiency) FROM bikelanes;

-- The lowest weight is -90
UPDATE bikelanes SET total_weight_efficiency = total_weight_efficiency + 90;

-- Join edge_id and edge_length with weights in new table
CREATE TABLE edge_weights AS
(SELECT gid, edge_id, start_node, end_node, total_weight_safety,
total_weight_efficiency, e.length_
FROM bikelanes bl
INNER JOIN network_topology.relation AS r
ON (((bl.topogeom).id , (bl.topogeom).layer_id)) = (r.topogeo_id, r.layer_id)
INNER JOIN network_topology.edge_data AS e ON (r.element_id = e.edge_id)
ORDER BY edge_id);

ALTER TABLE edge_weights ADD COLUMN safety_weight_distance NUMERIC;
ALTER TABLE edge_weights ADD COLUMN efficiency_weight_distance NUMERIC;

--Multiplying the weights with the distance
UPDATE edge_weights SET safety_weight_distance = (total_weight_safety *(length_/100))::INT;
UPDATE edge_weights SET efficiency_weight_distance = (total_weight_efficiency*(length_/100))::INT;
--Adding distance as a weight
UPDATE edge_weights SET safety_weight_distance = safety_weight_distance + length_::INT;
UPDATE edge_weights SET efficiency_weight_distance = efficiency_weight_distance + length_::INT;

-- Export edge and node data
\COPY (SELECT edge_id, start_node, end_node FROM network_topology.edge_data ORDER BY start_node)
TO 'edge_data.csv' DELIMITER ',' CSV HEADER;

-- Retrieving data about the connection between edges and the org. data with weights
\COPY (SELECT gid, topology.GetTopoGeomElements(topogeom) FROM bikelanes
ORDER BY gid) TO 'edge_relation.csv' DELIMITER ',' CSV HEADER;

-- Export tables with edge weights
\COPY (SELECT * from edge_weights) TO 'edge_relation_weights.csv' DELIMITER ',' CSV HEADER;
```

Appendix VI: Visualised Edge Weights

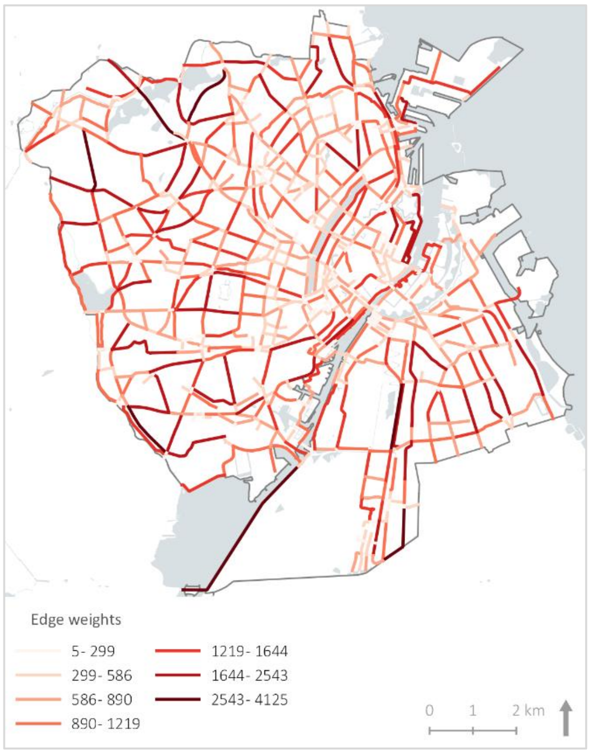

Figure 56. Edge weights based on $W_{efficiency}$

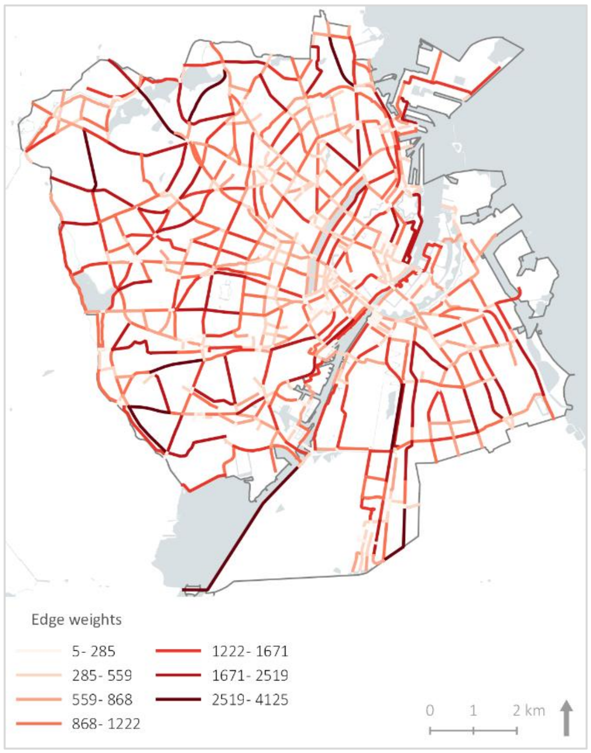

Figure 57. Edge weights based on W_{safety}

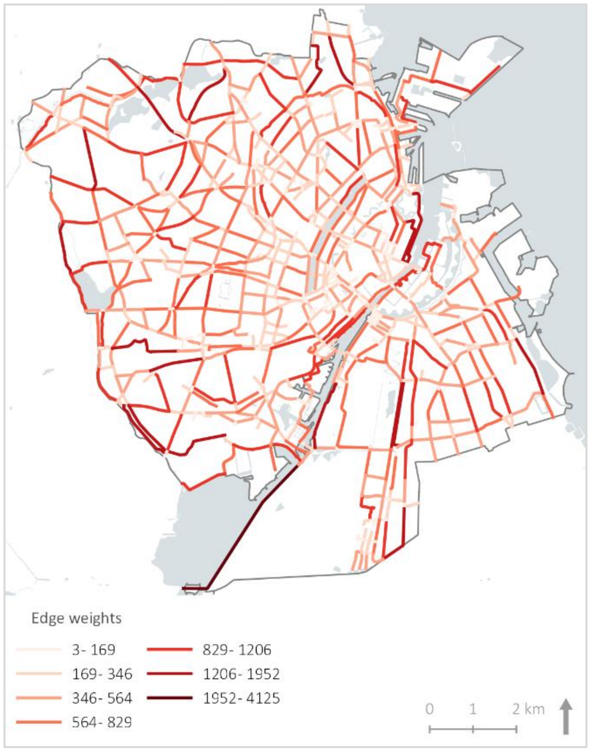

Figure 58. Edge weights based on $W_{distance}$

Appendix VII: Network Density with SQL

```sql
--Counting the number of vertices
SELECT COUNT(edge_id) FROM network_topology.edge_data;

CREATE TABLE vertices AS (SELECT start_node FROM network_topology.edge_data);
INSERT INTO vertices SELECT end_node FROM network_topology.edge_data;

ALTER TABLE vertices RENAME start_node TO vertex;

SELECT COUNT(DISTINCT vertex) FROM vertices;

-- Length of entire network
SELECT SUM(ST_Length(geom))/1000 FROM network_topology.edge_data;

--Length of network in different neighbourhoods
WITH inter AS (SELECT ST_Intersection(e.geom, n.geom) AS geom FROM network_topology.edge_data e, neighbourhoods n
 WHERE n.navn = 'Vanloese')
SELECT SUM(ST_Length(geom))/1000 FROM inter;

--Length of network per square kilometre in different neighbourhoods
WITH inter AS (SELECT ST_Intersection(e.geom, n.geom) AS geom FROM network_topology.edge_data e, neighbourhoods n
 WHERE n.navn LIKE 'Frederiksberg')
SELECT SUM(ST_Length(geom)/1000) /(SELECT area_km2 FROM neighbourhoods n WHERE n.navn LIKE 'Frederiksberg') FROM
inter;

-- Number of vertices with more than one edge connected (i.e. the vertex is present more than once)
CREATE VIEW intersections AS (SELECT vertex, COUNT(1) FROM vertices GROUP BY vertex HAVING COUNT(1) > 1 ORDER BY
vertex);
-- Join with original node geometry
CREATE VIEW inter_nodes AS (SELECT * FROM intersections INNER JOIN network_topology.node ON vertex = node_id);

-- Finding degree for each vertex
CREATE VIEW degree AS (SELECT vertex, COUNT(1) FROM vertices GROUP BY vertex ORDER BY vertex);
-- Joining degree with node geometry
CREATE VIEW nodes_degree AS (SELECT * FROM degree INNER JOIN network_topology.node ON vertex = node_id);

-- Find the number of intersections for each neighbourhood
SELECT COUNT(*) FROM inter_nodes i, neighbourhoods n WHERE ST_Intersects(i.geom, n.geom) AND n.navn LIKE 'Frederi
ksberg';
-- Find the number of intersections per square kilometres in different neighbourhoods
SELECT COUNT(*)/(SELECT area_km2 FROM neighbourhoods n WHERE n.navn LIKE 'Frederiksberg') FROM inter_nodes i, nei
ghbourhoods n WHERE ST_Intersects(i.geom, n.geom) AND n.navn LIKE 'Frederiksberg';
```

```sql
-- Retrieving the number of nodes and edges for each neighbourhood
SELECT COUNT(*) FROM network_topology.node v, neighbourhoods n
WHERE ST_Within(v.geom, n.geom) AND n.navn LIKE 'Bispebjerg%';

SELECT COUNT(*) FROM network_topology.edge_data e, neighbourhoods n
WHERE ST_Within(e.geom, n.geom) AND n.navn LIKE 'Frederiksberg%';

-- Finding stress centrality for neighbourhoods
WITH stress_centrality AS (SELECT index, edge_id, count, norm_stress, geom FROM stress_cent_dist s JOIN network_t
opology.edge_data e ON s.index = e.edge_id)
SELECT MIN(count)::INT FROM stress_centrality s, neighbourhoods n
WHERE ST_Intersects(s.geom, n.geom)
AND n.navn LIKE 'Bispebjerg%';

-- Finding closeness centrality for neighbourhoods
WITH closeness_centrality AS (SELECT index, node_id, sum_dist, norm_close, geom FROM closeness_eff c JOIN network
_topology.node n ON c.index = n.node_id)
SELECT ROUND(MIN(sum_dist)::NUMERIC,3) FROM closeness_centrality c, neighbourhoods n
WHERE ST_Intersects(c.geom, n.geom)
AND n.navn LIKE 'Amager V%';

-- Finding straightness centrality for neighbourhoods
WITH straightness_centrality AS (SELECT index, node_id, c_s, norm_straight, geom FROM straightness_eff s JOIN net
work_topology.node n ON s.index = n.node_id)
SELECT ROUND(MIN(c_s)::NUMERIC,3) FROM straightness_centrality s, neighbourhoods n
WHERE ST_Intersects(s.geom, n.geom)
AND n.navn LIKE 'Amager Ø%';

-- Comparing neighbourhoods
ALTER TABLE income_education ADD COLUMN income_norm NUMERIC;
UPDATE income_education
SET income_norm = ROUND((income/(SELECT MAX(income) FROM income_education)::NUMERIC);

ALTER TABLE income_education ADD COLUMN higher_ed_percent NUMERIC;
ALTER TABLE income_education ADD COLUMN highed_norm NUMERIC;
UPDATE income_education SET higher_ed_percent = percent_medium+percent_bachelor+percent_long
UPDATE income_education SET highed_norm = ROUND((higher_ed_percent/(SELECT MAX(higher_ed_percent)
FROM income_education)::NUMERIC),3);

ALTER TABLE age_density ADD COLUMN pop_dens_norm NUMERIC;
UPDATE age_density SET pop_dens_norm = ROUND((pop_dens/(SELECT MAX(pop_dens) FROM age_density))::NUMERIC,3);
```

```
ALTER TABLE age_density ADD COLUMN children_norm NUMERIC;
ALTER TABLE age_density ADD COLUMN young_norm NUMERIC;
ALTER TABLE age_density ADD COLUMN adults_norm NUMERIC;
ALTER TABLE age_density ADD COLUMN elderly_norm NUMERIC;

UPDATE age_density SET children_norm = ROUND((percent_0_17/(SELECT MAX(percent_0_17) FROM age_density))::NUMERIC,
3);
UPDATE age_density SET young_norm = ROUND((percent_18_24/(SELECT MAX(percent_18_24) FROM age_density))::NUMERIC,3
);
UPDATE age_density SET adults_norm = ROUND((percent_25_64/(SELECT MAX(percent_25_64) FROM age_density))::NUMERIC,
3);
UPDATE age_density SET elderly_norm = ROUND((percent_65_99/(SELECT MAX(percent_65_99) FROM age_density))::NUMERIC
,3);

CREATE TABLE neighbourhood_comparison_norm AS
(SELECT a.n_name, a.neigh_id, pop_dens_norm, children_norm, young_norm, adults_norm, elderly_norm, income_norm, h
ighed_norm
FROM age_density a INNER JOIN income_education i ON a.neigh_id = i.neigh_id);

CREATE TABLE neighbourhood_comparison AS
(SELECT i.n_name, i.neigh_id, income, pop_dens, higher_ed_percent, percent_0_17, percent_18_24, percent_25_64, pe
rcent_65_99
FROM income_education i JOIN age_density a ON i.neigh_id = a.neigh_id);
```

Appendix VIII: Adjacency/Incidence Matrix & Centrality Functions with Python

```python
'''
Function for creating and computing an adjacency matrix for an undirected graph
'''

def adj_matrix(input_data, start_column, end_column, edge_column):
    import pandas as pd

    '''
    The function requires one dataframe as input w. columns containing edge_id or edge weight, start vertex and
    end vertex
    Additionally, the column names for start_nodes, end_nodes and edge_id must be provided
    The function returns an adjacency matrix with the vertices as row indices and column names
    '''

    #Retrieving a list of all nodes and removing duplicates
    node_list = list(set(input_data[start_column].to_list() + input_data[end_column].to_list()))
    node_list.sort()

    #Creating an empty adjacency matrix
    output_matrix = pd.DataFrame(index=node_list, columns=node_list)

    for index, _ in input_data.iterrows():
        start_vertex = input_data.loc[index, start_column]
        end_vertex = input_data.loc[index, end_column]
        edge_id = input_data.loc[index, edge_column]
        output_matrix.at[start_vertex, end_vertex] = edge_id
        # Since the graph is not directed:
        output_matrix.at[end_vertex, start_vertex] = edge_id

    return output_matrix
```

```python
'''
Function for creating an incidence matrix for an undirected graph
'''

def inc_matrix(input_data, start_column, end_column, edge_column):
    '''
    The function requires one dataframe as input w. columns containing edge_id, start
    vertex and end vertex
    Additionally, the column names for start_nodes, end_nodes and edge_id
    must be provided
    The function returns an incidence matrix with the vertices as row indices and
    edges as column names
    '''
    import pandas as pd

    #Retrieving list of vertices, making sure that there are no duplicates
    vertices = list(set(input_data[start_column].to_list() +
    input_data[end_column].to_list()))
    vertices.sort()

    #Retrieving list of edges making sure that there are no duplicates
    edges = list(set(input_data[edge_column].to_list()))
    edges.sort()

    #Creating incidence matrix, initialised as 0
    inc_matrix = pd.DataFrame(data=0, index=vertices, columns=edges)

    #Looping through input data and filling out incidence matrix
    for _, row in input_data.iterrows():

        edge = row[edge_column]
        start_vertex = row[start_column]
        end_vertex = row[end_column]

        inc_matrix.at[start_vertex,edge] = 1
        inc_matrix.at[end_vertex,edge] = 1

    return inc_matrix
```

```
'''
Script with functions for computing shortest paths, lengths of shortest paths, stress centrality or euclidian dis
tance based on networkX
'''
import pandas as pd
import networkx as nx
import itertools

def construct_path_nodes(nodes, path_dictionary):
    '''
    Functions for recreating shortest paths computed with NetworkX
    The function returns the paths as a dataframe sort of like an adjacency matrix,
    but with lists of oath nodes as values
    Requires a list of nodes and dictionary with paths as input
    '''
    #Creating dataframe for resulting paths
    path_nodes = pd.DataFrame(index = nodes, columns= nodes)

    for i in nodes:
        for k in nodes:
            if k == i:
                continue
            path = nx.reconstruct_path(i, k, path_dictionary)
            path_nodes.at[i, k] = path



    return path_nodes

def construct_path_edges(nodes, path_dictionary, adjacency_matrix):
    '''
    Functions for recreating shortest paths computed with NetworkX
    The function returns the paths as a dataframe sort of like an adjacency matrix, but with lists of path edges
    as values
    Requires a list of nodes, dictionary with paths and adjacency matrix with edge ids as input
    '''
    #Creating dataframe for resulting paths
    path_edges = pd.DataFrame(index = nodes, columns= nodes)

    for i in nodes:
        for k in nodes:
            if k == i:
                continue
            path_nodes = nx.reconstruct_path(i, k, path_dictionary)
            #Empty list for edges
            edges = []
            for z in range(len(path_nodes)-1):
                #Finding edge id
                edge = adjacency_matrix.loc[path_nodes[z],path_nodes[z+1]]
                #Storing edges
                edges.append(edge)
            path_edges.at[i, k] = edges

    return path_edges
def construct_path_lengths(nodes, path_dictionary, adjacency_matrix):
    '''
    Functions for recreating shortest paths computed with NetworkX
    The function returns the paths as a dataframe sort of like an adjacency matrix, but with length between nodes
    as values
    Requires a list of nodes, dictionary with paths and adjacency matrix with edge lengths as input
    '''
    #Creating dataframe for path lengths
    path_lengths = pd.DataFrame(index = nodes, columns= nodes)

    for i in nodes:
        for k in nodes:
            if k == i:
                continue
            path_nodes = nx.reconstruct_path(i, k, path_dictionary)
            #Empty list for edges
            edges = []
            for z in range(len(path_nodes)-1):
                #Finding edge id
                edge = adjacency_matrix.loc[path_nodes[z],path_nodes[z+1]]
                #Storing edges
                edges.append(edge)
            length = sum(edges)
            path_lengths.at[i, k] = length

    return path_lengths


def compute_stress_centrality(edges, nodes, path_dictionary, adjacency_matrix):
    '''
    The function reconstructs paths based on a networkx path dictionary and counts how many paths an edge is part
    of
    Requires a list of all nodes and the path dictionary as input
    '''
    #Creating dataframe for stress centrality, initially all counts are set to zero
    zeros = [0]*len(edges)
    s_central = pd.DataFrame({'count': zeros}, index=edges)

    for a, b in itertools.combinations(nodes, 2):
        path_nodes = nx.reconstruct_path(a, b, path_dictionary)

        #Going through list path of nodes
        for z in range(len(path_nodes)-1):
            edge = adjacency_matrix.loc[path_nodes[z],path_nodes[z+1]]
            s_central.loc[edge, 'count'] = s_central.loc[edge, 'count'] + 1
    return s_central


def calculate_distance(point_geometries, node_id):
    '''
    Calculates the distance between Point geometries
    Needs a dataframe with the points and the point geometries and the name of the column with the node_id
    '''

    #Retrieving a list of all nodes and edges and removing duplicates
    node_ids = list(set(point_geometries['node_id']))
    node_ids.sort()

    distances = pd.DataFrame(index=node_ids, columns=node_ids)

    #Calculating the Euclidian distance between two points
    for i in node_ids:
        for k in node_ids:
            point_i = point_geometries.loc[point_geometries['node_id'] == i,'geom'].iloc[0]
            point_k = point_geometries.loc[point_geometries['node_id'] == k,'geom'].iloc[0]
            dist = point_i.distance(point_k)

            #Saving it to dataframe
            distances.at[i,k] = dist

    return distances


def distance_divided(euclidian_distance, path_length):
    #Functions divides values in dataframe w. euclidian distance with values from dataframe with path length
    #Dataframes should have same index and column names

    #Create new dataframe with same index and columns
    divided = pd.DataFrame(index=euclidian_distance.index, columns=euclidian_distance.index)

    #Fill any potential NaN-values
    euclidian_distance.fillna(0, inplace=True)
    path_length.fillna(0, inplace=True)

    #list of nodes
    nodes = euclidian_distance.index.to_list()
    for i in nodes:
        for k in nodes:
            divided.at[i,k] = euclidian_distance.loc[i,k]/path_length.loc[i,k]

    return divided
```

Appendix IX: Computing Vertex Degree & Clustering with Python

```
#Importing modules
import pandas as pd
from AdjacencyMatrix import adj_matrix
from sqlalchemy import create_engine

#Reading original file with edges and vertices
fp1 = r'C:\Users\viero\OneDrive\Documents\IGEON\THESIS\DATA\ANALYSIS_DATA\Code_files\edge_data.csv'
data = pd.read_csv(fp1)

#Using the function to fill the adjacency matrix
adj = adj_matrix(data, 'start_node', 'end_node', 'edge_id')

#Saving the adjacency matrix to csv
fp2 = r'C:\Users\viero\OneDrive\Documents\IGEON\THESIS\DATA\ANALYSIS_DATA\Code_files\adj_matrix.csv'
adj.to_csv(fp2)

#Counting number of connected edges to each vertex
count = list(adj.count())
intersections = [i for i in count if i != 1]
number_of_intersections = len(intersections)

#Loading adj_matrix into Postgres
engine = create_engine('postgresql://postgres:IGEON20@localhost:5432/network_analysis')

#Loading empty data frame to database
adj.to_sql("adj_mat", engine)

#Using psql to load data
\copy adj_mat FROM 'C:\Users\viero\OneDrive\Documents\IGEON\THESIS\DATA\ANALYSIS_DATA\Code_files\adj_matrix.csv'
DELIMITER ',' CSV HEADER;

#Retrieving a list of all nodes and removing duplicates
node_list = list(set(data['start_node'].to_list() + data['end_node'].to_list()))
node_list.sort()

#Empty dataframe to store clustering coefficient for each vertex
#Fill dataframe with zeros
zeros = [0.0] * len(node_list)
cluster = pd.DataFrame({'m_i':zeros, 'clustering':zeros, 'degree':zeros}, index=node_list)
```

```
#Replace NaN with 0
adj.fillna(0)

#Computing the clustering coefficients
for index, row in adj.iterrows():

    # Empty list for adjacenct vertices
    neighbours = []
    #Iterating through each columns
    for col in adj.columns:
        if row[col] > 0:
            neighbours.append(col)

    #Saving number of neighbours as degree
    d_i = len(neighbours)
    cluster.at[index,'degree'] = d_i

    #Number of links between adjacent vertices is initially set to 0
    m_i = 0

    #Iterate through neighbours to see if they are connected
    for i in neighbours:
        #Checking if adjacent vertices are connected
        for k in neighbours:
            if k == i:
                continue
            if adj.loc[i, k] > 0:
                m_i =+ 1

    #Update with number of links between adjacent vertices
    cluster.at[index,'m_i'] = m_i

    #Compute cluster coefficient
    if d_i > 1:
        cluster.at[index,'clustering'] = (2*m_i)/(d_i*(d_i-1))

#Find number of vertices with any clustering
is_clustered = cluster['clustering'] > 0
clustered_vertices = cluster[is_clustered]

#Find min, maximum and average clustering coefficient
min_cluster = clustered_vertices['clustering'].min()
max_cluster = cluster['clustering'].max()
```

```
ave_cluster = clustered_vertices['clustering'].mean()

#Find global clustering coefficient
global_cluster = cluster['clustering'].mean()

#Find min, max, average and median degree
min_degree = cluster['degree'].min()
max_degree = cluster['degree'].max()
mean_degree = cluster['degree'].mean()
median_degree = cluster['degree'].median()

#Exporting cluster to csv
fp3 = r'C:\Users\viero\OneDrive\Documents\IGEON\THESIS\DATA\ANALYSIS_DATA\Code_files\cluster.csv'
cluster.to_csv(fp3)

#Grouping by number of degrees
grouped_degree = cluster['degree'].value_counts()
fp4 = r'C:\Users\viero\OneDrive\Documents\IGEON\THESIS\DATA\ANALYSIS_DATA\Code_files\grouped_degree.csv'
grouped_degree.to_csv(fp4)

#Grouping by clustering
grouped_cluster = cluster['clustering'].value_counts()
fp5 = r'C:\Users\viero\OneDrive\Documents\IGEON\THESIS\DATA\ANALYSIS_DATA\Code_files\grouped_clustering.csv'
grouped_cluster.to_csv(fp5)
```

Appendix X: Shortest Path Analysis with Python

```
#Importing modules
from AdjacencyMatrix import adj_matrix
from centrality_functions import construct_path_edges, construct_path_nodes, construct_path_lengths, compute_stre
ss_centrality, calculate_distance, distance_divided
import pandas as pd
import geopandas as gpd
import networkx as nx
import numpy as np
from sqlalchemy import create_engine
import psycopg2

#Loading file with edge weights
fp1 = 'edge_relation_weights.csv'
edge_weights = pd.read_csv(fp1)

#Creating adjecency matrices with weights
adj_safety = adj_matrix(edge_weights, 'start_node', 'end_node', 'safety_weight_distance')
adj_eff = adj_matrix(edge_weights, 'start_node', 'end_node', 'efficiency_weight_distance')
adj_dist = adj_matrix(edge_weights, 'start_node', 'end_node', 'length_')
adj = adj_matrix(edge_weights, 'start_node', 'end_node', 'edge_id')

# NaN-values are replaced with zeros and Dataframe converted to numeric datatype
adj_eff.fillna(0, inplace = True)
adj_eff.apply(pd.to_numeric)

adj_safety.fillna(0, inplace= True)
adj_safety.apply(pd.to_numeric)

adj_dist.fillna(0, inplace=True)
adj_dist.apply(pd.to_numeric)
adj_dist.round(1)

#Creating the graphs using NetworkX
graph_e = nx.from_pandas_adjacency(adj_eff)
graph_s = nx.from_pandas_adjacency(adj_safety)
graph_d = nx.from_pandas_adjacency(adj_dist)

#Finding path and length between all vertex pairs, based on respectively different weights and geographical
distance
path_e, length_e = nx.floyd_warshall_predecessor_and_distance(graph_e)
```

```
path_s, length_s = nx.floyd_warshall_predecessor_and_distance(graph_s)
path_d, length_d = nx.floyd_warshall_predecessor_and_distance(graph_d)

#Retrieving a list of all nodes and edges and removing duplicates
node_list = list(set(edge_weights['start_node'].to_list() + edge_weights['end_node'].to_list()))
node_list.sort()

edge_list = list(set(edge_weights['edge_id'].to_list()))
edge_list.sort()

#Retrieving all paths as nodes
nodepaths_eff = construct_path_nodes(node_list, path_e)
nodepaths_safe = construct_path_nodes(node_list, path_s)
nodepaths_distance = construct_path_nodes(node_list, path_d)

#Retrieving all paths as edges
edgepaths_eff = construct_path_edges(node_list, path_e, adj)
edgepaths_safe = construct_path_edges(node_list, path_s, adj)
edgepaths_distance = construct_path_edges(node_list, path_d, adj)

#Computing lengths of shortest paths between all vertex pairs
lengths_eff = construct_path_lengths(node_list, path_e, adj_dist)
lengths_safe = construct_path_lengths(node_list, path_s, adj_dist)
lengths_dist = construct_path_lengths(node_list, path_d, adj_dist)

#Average path length
#Finding the sum of all path lengths (in km)
sum_path_eff = pd.DataFrame(lengths_eff.sum())
sum_path_eff.rename(columns ={0:'total_length'}, inplace = True)
total_length_e = int(sum_path_eff.sum()/1000)

sum_path_safe = pd.DataFrame(lengths_safe.sum())
sum_path_safe.rename(columns ={0:'total_length'}, inplace = True)
total_length_s = int(sum_path_safe.sum()/1000)

sum_path_dist = pd.DataFrame(lengths_dist.sum())
sum_path_dist.rename(columns ={0:'total_length'}, inplace = True)
total_length_d = int(sum_path_dist.sum()/1000)

#Normalised total path length
sum_path_eff['total_length_norm'] = sum_path_eff['total_length']/sum_path_eff['total_length'].max()
sum_path_safe['total_length_norm'] = sum_path_safe['total_length']/sum_path_safe['total_length'].max()
sum_path_dist['total_length_norm'] = sum_path_dist['total_length']/sum_path_dist['total_length'].max()
```

```
#Load summarised path lengths into Postgres
engine = create_engine('postgresql://postgres:IGEON20@localhost:5432/network_analysis')

sum_path_eff.to_sql('total_pathlength_eff', engine)
sum_path_safe.to_sql('total_pathlength_safe', engine)
sum_path_dist.to_sql('total_pathlength_dist', engine)

#Network efficiency
def divide_with_length(matrix):
    divided = matrix.copy(deep = True)
    divided.apply(pd.to_numeric)
    divided.fillna(0, inplace = True)
    for index, row in divided.iterrows():
        for col in divided.columns:
            if row[col] == 0:
                continue
            length = (row[col]/1000)
            divided.at[index,col] = 1/length

    return divided

efficiency_eff = divide_with_length(lengths_eff)
efficiency_safe = divide_with_length(lengths_safe)
efficiency_dist = divide_with_length(lengths_dist)

#Find total values for efficiency computation
sum_efficiency_e = sum(efficiency_eff.sum())
sum_efficiency_s = sum(efficiency_safe.sum())
sum_efficiency_d = sum(efficiency_dist.sum())

#Compute stress centrality
stress_cent_eff = compute_stress_centrality(edge_list, node_list, path_e, adj)
stress_cent_safe = compute_stress_centrality(edge_list, node_list, path_s, adj)
stress_cent_dist = compute_stress_centrality(edge_list, node_list, path_d, adj)

#Normalise values for stress centrality
#Find and divide with max value
max_stress_e = stress_cent_eff['count'].max()
max_stress_s = stress_cent_safe['count'].max()
max_stress_d = stress_cent_dist['count'].max()

stress_cent_eff['norm_stress'] = stress_cent_eff['count'] / max_stress_e
```

```
stress_cent_safe['norm_stress'] = stress_cent_safe['count'] / max_stress_s
stress_cent_dist['norm_stress'] = stress_cent_dist['count'] / max_stress_d

#Calculating betweenness centrality
path_count = 247456

stress_cent_eff['between'] = stress_cent_eff['count'] / path_count
stress_cent_safe['between'] = stress_cent_safe['count'] / path_count
stress_cent_dist['between'] = stress_cent_dist['count'] / path_count

#Normalise values for betweenness centrality
#Find and divide with max values
max_between_e = stress_cent_eff['between'].max()
max_between_s = stress_cent_safe['between'].max()
max_between_d = stress_cent_dist['between'].max()

stress_cent_eff['norm_between'] = stress_cent_eff['between'] / max_between_e
stress_cent_safe['norm_between'] = stress_cent_safe['between'] / max_between_s
stress_cent_dist['norm_between'] = stress_cent_dist['between'] / max_between_d

#Exporting stress centrality
fp2 = 'stress_centrality_eff.csv'
fp3 = 'stress_centrality_safe.csv'
fp4 = 'stress_centrality_dist.csv'

stress_cent_eff.to_csv(fp2)
stress_cent_safe.to_csv(fp3)
stress_cent_dist.to_csv(fp4)

#Load stress and betweenness centrality data into Postgres
stress_cent_eff.to_sql('stress_cent_eff', engine)
stress_cent_safe.to_sql('stress_cent_safe', engine)
stress_cent_dist.to_sql('stress_cent_dist', engine)



#Calculating Betweenness Centrality with NetworkX
between_eff = nx.edge_betweenness_centrality(graph_e, weight='weight', normalise = False)
between_safe = nx.edge_betweenness_centrality(graph_s, weight='weight', normalise = False)
between_dist = nx.edge_betweenness_centrality(graph_d, weight='weight', normalise = False)

#Closeness centrality
v_minus1 = 704 - 1
closeness_efficiency = pd.DataFrame()
```

```
closeness_efficiency['sum_dist'] = v_minus1/(lengths_eff.sum(axis = 0)/1000)
closeness_safety = pd.DataFrame()
closeness_safety['sum_dist'] = v_minus1/(lengths_safe.sum(axis = 0)/1000)
closeness_distance = pd.DataFrame()
closeness_distance['sum_dist'] = v_minus1/(lengths_dist.sum(axis = 0)/1000)

#Normalise values
closeness_efficiency['norm_close'] = closeness_efficiency['sum_dist'] / closeness_efficiency['sum_dist'].max()
closeness_safety['norm_close'] = closeness_safety['sum_dist'] / closeness_safety['sum_dist'].max()
closeness_distance['norm_close'] = closeness_distance['sum_dist'] / closeness_distance['sum_dist'].max()

#Export closeness centrality values
fp8 = 'closeness_eff.csv'
fp9 = 'closeness_safe.csv'
fp10 = 'closeness_dist.csv'

closeness_efficiency.to_csv(fp8)
closeness_safety.to_csv(fp9)
closeness_distance.to_csv(fp10)

#Load Closeness Centrality to postgres database
closeness_efficiency.to_sql('closeness_eff', engine)
closeness_safety.to_sql('closeness_safe', engine)
closeness_distance.to_sql('closeness_dist', engine)

#Straightness Centrality
#Connecting to database
connection = psycopg2.connect(database='network_analysis', user='postgres', password='IGEON20',
    host='localhost')

sql_query = 'SELECT node_id, geom FROM network_topology.node'

#Using geopandas to store geometries
nodes = gpd.GeoDataFrame.from_postgis(sql_query, connection, geom_col='geom' )

#Use distance function to compute straight line distance between vertices
euclidian_dist = calculate_distance(nodes, 'node_id')

#Find differences between Euclidian and path length with function for divided distances
divided_eff = distance_divided(euclidian_dist, lengths_eff)
divided_safe = distance_divided(euclidian_dist, lengths_safe)
divided_dist = distance_divided(euclidian_dist, lengths_dist)
```

```
#Find final values for Straightness Centrality computation
const = 1/(704-1)
straightness_eff = pd.DataFrame()
straightness_eff['c_s'] = const*divided_eff.sum(axis=0)
straightness_safe = pd.DataFrame()
straightness_safe['c_s'] = const*divided_safe.sum(axis=0)
straightness_dist = pd.DataFrame()
straightness_dist['c_s'] = const*divided_dist.sum(axis=0)

#Normalise values
straightness_eff['norm_straight'] = straightness_eff['c_s'] / straightness_eff['c_s'].max()
straightness_safe['norm_straight'] = straightness_safe['c_s'] / straightness_safe['c_s'].max()
straightness_dist['norm_straight'] = straightness_dist['c_s'] / straightness_dist['c_s'].max()

#Export Straightness Centrality values
fp11 = 'straightness_eff.csv'
fp12 = 'straightness_safe.csv'
fp13 = 'straightness_dist.csv'

straightness_eff.to_csv(fp11)
straightness_safe.to_csv(fp12)
straightness_dist.to_csv(fp13)

#Upload Straightness Centrality data to databas
straightness_eff.to_sql('straightness_eff', engine)
straightness_safe.to_sql('straightness_safe', engine)
straightness_dist.to_sql('straightness_dist', engine)
```

Appendix XI: Comparing Neighbourhood Indices with Python

```python
#Importing modules
import pandas as pd
from sqlalchemy import create_engine
import psycopg2 as pg
import pandas.io.sql as psql
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm

#Reading files with centrality and connectivity indicy values for each neighbourhood and weighting scheme
neigh_eff = pd.read_csv('Neighbourhood_efficiency.csv')
neigh_safe = pd.read_csv('Neighbourhood_safety.csv')
neigh_dist = pd.read_csv('Neighbourhood_distance.csv')

#Only keeping columns with average values
col_names = ['Neigh_id', 'Average Stress Centrality','Average Straightness Centrality', 'Average Closeness Centra
lity']
neigh_eff_ave = neigh_eff[col_names]
neigh_safe_ave = neigh_safe[col_names]
neigh_dist_ave = neigh_dist[col_names]

#Computing differences between weighting schemes as percentage
neigh_dist_eff_ave = 100 - (neigh_dist_ave.set_index('Neigh_id').div(neigh_eff_ave.set_index('Neigh_id')).mul(100
))
neigh_dist_safe_ave = 100 - (neigh_dist_ave.set_index('Neigh_id').div(neigh_safe_ave.set_index('Neigh_id')).mul(1
00))

neigh_dist_eff_ave = neigh_dist_eff_ave.round(3)
neigh_dist_safe_ave = neigh_dist_safe_ave.round(3)

#Load results into Postgres
engine = create_engine('postgresql://postgres:IGEON20@localhost:5432/network_analysis')

neigh_dist_eff_ave.to_sql('ave_diff_dist_eff', engine)
neigh_dist_safe_ave.to_sql('ave_diff_dist_safe', engine)

neigh_eff.to_sql('neighbourhood_eff', engine)
neigh_safe.to_sql('neighbourhood_safe', engine)
neigh_dist.to_sql('neighbourhood_dist',engine)
```

```
#Examining the relationship between socio-economic characteristics and centrality indices
neigh_eff_ave.set_index('Neigh_id', inplace = True)
neigh_eff_ave.sort_index(inplace = True)
neigh_safe_ave.set_index('Neigh_id', inplace = True)
neigh_safe_ave.sort_index(inplace = True)
neigh_dist_ave.set_index('Neigh_id', inplace = True)
neigh_dist_ave.sort_index(inplace = True)

#Retrieving data from Postgres database
connection = pg.connect(database='network_analysis', user='postgres', password='IGEON20', host='localhost')

sql_query1 = 'SELECT * FROM neighbourhood_comparison ORDER BY neigh_id'

neigh_data = psql.read_sql(sql_query1, connection)
neigh_data.set_index('neigh_id', inplace = True)
neigh_data.sort_index(inplace = True)

sql_query2 = 'SELECT * FROM neighbourhood_comparison_norm ORDER BY neigh_id'

neigh_data_norm = psql.read_sql(sql_query2, connection)
neigh_data_norm.set_index('neigh_id', inplace = True)
neigh_data_norm.sort_index(inplace = True)

sql_query3 = 'SELECT * FROM intersection_density ORDER BY neigh_id'
intersections = psql.read_sql(sql_query3, connection)
intersections.set_index('neigh_id', inplace = True)
intersections.drop(12, axis=0, inplace=True)

sql_query4 = 'SELECT * from street_density ORDER BY neigh_id'
street_dens = psql.read_sql(sql_query4, connection)
street_dens.set_index('neigh_id', inplace = True)
street_dens.drop(12, axis=0, inplace = True)

#Examining the correlation between network density and socio-economic and demographic variables

#Creating dataframes with all variables to be compared
density_merge = intersections.merge(street_dens, left_index=True, right_index=True)
density_variables = density_merge[['intersect_density', 'street_density']].copy()
variables_all = density_variables.merge(neigh_data, left_index=True, right_index=True)
variables = variables_all[['intersect_density','street_density', 'pop_dens', 'income', 'higher_ed_percent']].copy
()

variables_dist = variables.merge(neigh_dist_ave, left_index=True, right_index=True)
```

```python
variables_eff = variables.merge(neigh_eff_ave, left_index=True, right_index=True)
variables_eff.drop(['intersect_density', 'street_density'], axis=1, inplace=True)
variables_safe = variables.merge(neigh_safe_ave, left_index=True, right_index=True)
variables_safe.drop(['intersect_density', 'street_density'], axis=1, inplace=True)

#Export variables to csv
variables_dist.to_csv('variables.csv')

#Using pairplot for an initial screaning of correlation
pairplot_dist = sb.pairplot(variables_dist)
pairplot_efff = sb.pairplot(variables_eff)
pairplot_safe = sb.pairplot(variables_safe)

#Saving figures to file
pairplot_dist.savefig('pairplot_dist.png')
pairplot_efff.savefig('pairplot_eff.png')
pairplot_safe.savefig('pairplot_safe.png')

#Performing linear regression
X = variables_dist['pop_dens']
Y = variables_dist['Average Stress Centrality']

model = sm.OLS(X, Y).fit()
predictions = model.predict(Y)
model.summary()

#Plotting values
x = X.values.reshape(-1,1)
y = Y.values.reshape(-1,1)

linear_regres = LinearRegression()
linear_regres.fit(x,y)
y_pred = linear_regres.predict(x)

plt.scatter(X,Y)
plt.plot(x,y_pred, color='purple')
plt.title('Correlation')
plt.xlabel('Population Density')
plt.ylabel('Average Stress Centrality')

#Plotting residuals
plt.hist(model.resid_pearson)
```

Appendix XII: Neighbourhood Centralities with SQL

```sql
-- Finding stress centrality for neighbourhoods
WITH stress_centrality AS (SELECT index, edge_id, count, norm_stress, geom FROM stress_cent_dist s JOIN network_t
opology.edge_data e ON s.index = e.edge_id)
SELECT MIN(count)::INT FROM stress_centrality s, neighbourhoods n
WHERE ST_Intersects(s.geom, n.geom)
AND n.navn LIKE 'Bispebjerg%';

-- Finding closeness centrality for neighbourhoods
WITH closeness_centrality AS (SELECT index, node_id, sum_dist, norm_close, geom FROM closeness_eff c JOIN network
_topology.node n ON c.index = n.node_id)
SELECT ROUND(MIN(sum_dist)::NUMERIC,3) FROM closeness_centrality c, neighbourhoods n
WHERE ST_Intersects(c.geom, n.geom)
AND n.navn LIKE 'Amager V%';

-- Finding straightness centrality for neighbourhoods
WITH straightness_centrality AS (SELECT index, node_id, c_s, norm_straight, geom FROM straightness_eff s JOIN net
work_topology.node n ON s.index = n.node_id)
SELECT ROUND(MIN(c_s)::NUMERIC,3) FROM straightness_centrality s, neighbourhoods n
WHERE ST_Intersects(s.geom, n.geom)
AND n.navn LIKE 'Amager Ø%';

-- Comparing neighbourhoods
ALTER TABLE income_education ADD COLUMN income_norm NUMERIC;
UPDATE income_education
SET income_norm = ROUND((income/(SELECT MAX(income) FROM income_education)::NUMERIC);

ALTER TABLE income_education ADD COLUMN higher_ed_percent NUMERIC;
ALTER TABLE income_education ADD COLUMN highed_norm NUMERIC;
UPDATE income_education SET higher_ed_percent = percent_medium+percent_bachelor+percent_long
UPDATE income_education SET highed_norm = ROUND((higher_ed_percent/(SELECT MAX(higher_ed_percent)
FROM income_education)::NUMERIC),3);

ALTER TABLE age_density ADD COLUMN pop_dens_norm NUMERIC;
UPDATE age_density SET pop_dens_norm = ROUND((pop_dens/(SELECT MAX(pop_dens) FROM age_density))::NUMERIC,3);

ALTER TABLE age_density ADD COLUMN children_norm NUMERIC;
ALTER TABLE age_density ADD COLUMN young_norm NUMERIC;
ALTER TABLE age_density ADD COLUMN adults_norm NUMERIC;
ALTER TABLE age_density ADD COLUMN elderly_norm NUMERIC;
```

```
UPDATE age_density SET children_norm = ROUND((percent_0_17/(SELECT MAX(percent_0_17) FROM age_density))::NUMERIC,
3);
UPDATE age_density SET young_norm = ROUND((percent_18_24/(SELECT MAX(percent_18_24) FROM age_density))::NUMERIC,3
);
UPDATE age_density SET adults_norm = ROUND((percent_25_64/(SELECT MAX(percent_25_64) FROM age_density))::NUMERIC,
3);
UPDATE age_density SET elderly_norm = ROUND((percent_65_99/(SELECT MAX(percent_65_99) FROM age_density))::NUMERIC
,3);

CREATE TABLE neighbourhood_comparison_norm AS
(SELECT a.n_name, a.neigh_id, pop_dens_norm, children_norm, young_norm, adults_norm, elderly_norm, income_norm, h
ighed_norm
FROM age_density a INNER JOIN income_education i ON a.neigh_id = i.neigh_id);

CREATE TABLE neighbourhood_comparison AS
(SELECT i.n_name, i.neigh_id, income, pop_dens, higher_ed_percent, percent_0_17, percent_18_24, percent_25_64, pe
rcent_65_99
FROM income_education i JOIN age_density a ON i.neigh_id = a.neigh_id);
```

Appendix XIII: Implementation of the Floyd-Warshall Algorithm

```
'''
This script contains an implementation of the Floyd Warshall Algorithm for finding the shortest path between all
vertex pairs
The first function uses a pandas dataframe with an adjacency matrix as input and returns a distance_matrixance ma
trix and a dataframe with the last visited vertex as output
The second function returns the path between two vertices given the path matrix as input
'''

#Importing modules
import pandas as pd
import numpy as np
from random import randint
import math

def floyd_warshall(adjacency_matrix):
    #Creating distance_matrixance matrix with similar edge weights
    distance_matrix = adjacency_matrix.copy(deep=True)

    #Creating empty path matrix
    path_matrix = pd.DataFrame(data=None, index=distance_matrix.index, columns=distance_matrix.columns)

    vertices = list(adjacency_matrix.columns)

    #distance_matrixance is set to infinity if there is no edge between vertices
    for i in vertices:
        for j in vertices:
            if math.isnan(distance_matrix.loc[i,j]) == True:
                distance_matrix.at[i,j] = math.inf

    #Setting the distance_matrixance to zero where start and end vertex are identical
    #Setting the path matrix as v where where start vertex v and end vertex v are identical
    for i in vertices:
        for j in vertices:
            if i == j:
                distance_matrix.at[i,j] = 0
                path_matrix.at[i,j] = i

    #Filling path matrix with initial values
    for i in vertices:
        for j in vertices:
            if adjacency_matrix.loc[i,j]:
                path_matrix.at[i,j] = j
                path_matrix.at[j,i] = i

    #Finding the shortest path between vertices
    for k in range(1,11):
        for i in range(1,11):
            for j in range(1,11):
                if distance_matrix.loc[i,j] > distance_matrix.loc[i,k] + distance_matrix.loc[k,j]:
                    distance_matrix.at[i,j] = distance_matrix.loc[i,k] + distance_matrix.loc[k,j]
                    path_matrix.at[i,j] = path_matrix.loc[i,k]

    return distance_matrix, path_matrix

#Function for returning the path between two vertices
def return_path(u, v, path_matrix):
    if path_matrix.loc[u,v] == False:
        path = 'There is no path'
    path = [u]
    while u != v:
        u = path_matrix.loc[u,v]
        path.append(u)
    return path


#Testing

#Test data
vertices = list(range(1,11))
#Empty adjacency matrix
adj = pd.DataFrame(np.nan, index=vertices, columns=vertices)

#Filling adjacency matrix with random values
for i in vertices:
    for j in vertices:
        if i == j:
            continue
        if i % 2:
            edge_value = randint(3,20)
            adj.at[i,j] = edge_value
            adj.at[j,i] = edge_value

#Finding distances and last visited vertex
dist, path = floyd_warshall(adj)
```

```
start_node = 8
end_node = 1
test_path = return_path(start_node, end_node, path)
print('The path between {start} and {end} is:'.format(start=start_node,end=end_node),test_path)
```

Department of Physical Geography and Ecosystem Science

Master Thesis in Geographical Information Science

17. *Laura V. Drews:* Multi-criteria GIS analysis for siting of small wind power plants - A case study from Berlin (2012).
18. *Qaisar Nadeem:* Best living neighborhood in the city - A GIS based multi criteria evaluation of ArRiyadh City (2012).
19. *Ahmed Mohamed El Saeid Mustafa:* Development of a photo voltaic building rooftop integration analysis tool for GIS for Dokki District, Cairo, Egypt (2012).
20. *Daniel Patrick Taylor*: Eastern Oyster Aquaculture: Estuarine Remediation via Site Suitability and Spatially Explicit Carrying Capacity Modeling in Virginia's Chesapeake Bay (2013).
21. *Angeleta Oveta Wilson:* A Participatory GIS approach to *unearthing* Manchester's Cultural Heritage '*gold mine'* (2013).
22. *Ola Svensson:* Visibility and Tholos Tombs in the Messenian Landscape: A Comparative Case Study of the Pylian Hinterlands and the Soulima Valley (2013).
23. *Monika Ogden:* Land use impact on water quality in two river systems in South Africa (2013).
24. *Stefan Rova:* A GIS based approach assessing phosphorus load impact on Lake Flaten in Salem, Sweden (2013).
25. *Yann Buhot:* Analysis of the history of landscape changes over a period of 200 years. How can we predict past landscape pattern scenario and the impact on habitat diversity? (2013).
26. *Christina Fotiou:* Evaluating habitat suitability and spectral heterogeneity models to predict weed species presence (2014).
27. *Inese Linuza:* Accuracy Assessment in Glacier Change Analysis (2014).
28. *Agnieszka Griffin:* Domestic energy consumption and social living standards: a GIS analysis within the Greater London Authority area (2014).
29. *Brynja Guðmundsdóttir:* Detection of potential arable land with remote sensing and GIS - A Case Study for Kjósarhreppur (2014).
30. *Oleksandr Nekrasov:* Processing of MODIS Vegetation Indices for analysis of agricultural droughts in the southern Ukraine between the years 2000-2012 (2014).
31. *Sarah Tressel:* Recommendations for a polar Earth science portal in the context of Arctic Spatial Data Infrastructure (2014).
32. *Caroline Gevaert:* Combining Hyperspectral UAV and Multispectral Formosat-2 Imagery for Precision Agriculture Applications (2014).
33. *Salem Jamal-Uddeen:* Using GeoTools to implement the multi-criteria evaluation analysis - weighted linear combination model (2014).
34. *Samanah Seyedi-Shandiz:* Schematic representation of geographical railway network at the Swedish Transport Administration (2014).
35. *Kazi Masel Ullah:* Urban Land-use planning using Geographical Information System and analytical hierarchy process: case study Dhaka City (2014).
36. *Alexia Chang-Wailing Spitteler:* Development of a web application based on MCDA and GIS for the decision support of river and floodplain rehabilitation projects (2014).
37. *Alessandro De Martino:* Geographic accessibility analysis and evaluation of potential changes to the public transportation system in the City of Milan (2014).

38. *Alireza Mollasalehi:* GIS Based Modelling for Fuel Reduction Using Controlled Burn in Australia. Case Study: Logan City, QLD (2015).
39. *Negin A. Sanati:* Chronic Kidney Disease Mortality in Costa Rica; Geographical Distribution, Spatial Analysis and Non-traditional Risk Factors (2015).
40. *Karen McIntyre:* Benthic mapping of the Bluefields Bay fish sanctuary, Jamaica (2015).
41. *Kees van Duijvendijk:* Feasibility of a low-cost weather sensor network for agricultural purposes: A preliminary assessment (2015).
42. *Sebastian Andersson Hylander:* Evaluation of cultural ecosystem services using GIS (2015).
43. *Deborah Bowyer:* Measuring Urban Growth, Urban Form and Accessibility as Indicators of Urban Sprawl in Hamilton, New Zealand (2015).
44. *Stefan Arvidsson:* Relationship between tree species composition and phenology extracted from satellite data in Swedish forests (2015).
45. *Damián Giménez Cruz*: GIS-based optimal localisation of beekeeping in rural Kenya (2016).
46. *Alejandra Narváez Vallejo:* Can the introduction of the topographic indices in LPJ-GUESS improve the spatial representation of environmental variables? (2016).
47. *Anna Lundgren:* Development of a method for mapping the highest coastline in Sweden using breaklines extracted from high resolution digital elevation models (2016).
48. *Oluwatomi Esther Adejoro:* Does location also matter? A spatial analysis of social achievements of young South Australians (2016).
49. *Hristo Dobrev Tomov:* Automated temporal NDVI analysis over the Middle East for the period 1982 - 2010 (2016).
50. *Vincent Muller:* Impact of Security Context on Mobile Clinic Activities
    A GIS Multi Criteria Evaluation based on an MSF Humanitarian Mission in Cameroon (2016).
51. *Gezahagn Negash Seboka:* Spatial Assessment of NDVI as an Indicator of Desertification in Ethiopia using Remote Sensing and GIS (2016).
52. *Holly Buhler:* Evaluation of Interfacility Medical Transport Journey Times in Southeastern British Columbia. (2016).
53. *Lars Ole Grottenberg*: Assessing the ability to share spatial data between emergency management organisations in the High North (2016).
54. *Sean Grant:* The Right Tree in the Right Place: Using GIS to Maximize the Net Benefits from Urban Forests (2016).
55. *Irshad Jamal:* Multi-Criteria GIS Analysis for School Site Selection in Gorno-Badakhshan Autonomous Oblast, Tajikistan (2016).
56. *Fulgencio Sanmartín:* Wisdom-volkano: A novel tool based on open GIS and time-series visualization to analyse and share volcanic data (2016).
57. *Nezha Acil:* Remote sensing-based monitoring of snow cover dynamics and its influence on vegetation growth in the Middle Atlas Mountains (2016).
58. *Julia Hjalmarsson:* A Weighty Issue: Estimation of Fire Size with Geographically Weighted Logistic Regression (2016).

59. *Mathewos Tamiru Amato:* Using multi-criteria evaluation and GIS for chronic food and nutrition insecurity indicators analysis in Ethiopia (2016).
60. *Karim Alaa El Din Mohamed Soliman El Attar:* Bicycling Suitability in Downtown, Cairo, Egypt (2016).
61. *Gilbert Akol Echelai:* Asset Management: Integrating GIS as a Decision Support Tool in Meter Management in National Water and Sewerage Corporation (2016).
62. *Terje Slinning:* Analytic comparison of multibeam echo soundings (2016).
63. *Gréta Hlín Sveinsdóttir:* GIS-based MCDA for decision support: A framework for wind farm siting in Iceland (2017).
64. *Jonas Sjögren:* Consequences of a flood in Kristianstad, Sweden: A GIS-based analysis of impacts on important societal functions (2017).
65. *Nadine Raska:* 3D geologic subsurface modelling within the Mackenzie Plain, Northwest Territories, Canada (2017).
66. *Panagiotis Symeonidis:* Study of spatial and temporal variation of atmospheric optical parameters and their relation with PM 2.5 concentration over Europe using GIS technologies (2017).
67. *Michaela Bobeck:* A GIS-based Multi-Criteria Decision Analysis of Wind Farm Site Suitability in New South Wales, Australia, from a Sustainable Development Perspective (2017).
68. *Raghdaa Eissa:* Developing a GIS Model for the Assessment of Outdoor Recreational Facilities in New Cities Case Study: Tenth of Ramadan City, Egypt (2017).
69. *Zahra Khais Shahid:* Biofuel plantations and isoprene emissions in Svea and Götaland (2017).
70. *Mirza Amir Liaquat Baig:* Using geographical information systems in epidemiology: Mapping and analyzing occurrence of diarrhea in urban - residential area of Islamabad, Pakistan (2017).
71. *Joakim Jörwall:* Quantitative model of Present and Future well-being in the EU-28: A spatial Multi-Criteria Evaluation of socioeconomic and climatic comfort factors (2017).
72. *Elin Haettner:* Energy Poverty in the Dublin Region: Modelling Geographies of Risk (2017).
73. *Harry Eriksson:* Geochemistry of stream plants and its statistical relations to soil- and bedrock geology, slope directions and till geochemistry. A GIS-analysis of small catchments in northern Sweden (2017).
74. *Daniel Gardevärn:* PPGIS and Public meetings – An evaluation of public participation methods for urban planning (2017).
75. *Kim Friberg:* Sensitivity Analysis and Calibration of Multi Energy Balance Land Surface Model Parameters (2017).
76. *Viktor Svanerud:* Taking the bus to the park? A study of accessibility to green areas in Gothenburg through different modes of transport (2017).
77. *Lisa-Gaye Greene:* Deadly Designs: The Impact of Road Design on Road Crash Patterns along Jamaica’s North Coast Highway (2017).
78. *Katarina Jemec Parker:* Spatial and temporal analysis of fecal indicator bacteria concentrations in beach water in San Diego, California (2017).

79. *Angela Kabiru:* An Exploratory Study of Middle Stone Age and Later Stone Age Site Locations in Kenya's Central Rift Valley Using Landscape Analysis: A GIS Approach (2017).
80. *Kristean Björkmann:* Subjective Well-Being and Environment: A GIS-Based Analysis (2018).
81. *Williams Erhunmonmen Ojo:* Measuring spatial accessibility to healthcare for people living with HIV-AIDS in southern Nigeria (2018).
82. *Daniel Assefa:* Developing Data Extraction and Dynamic Data Visualization (Styling) Modules for Web GIS Risk Assessment System (WGRAS). (2018).
83. *Adela Nistora:* Inundation scenarios in a changing climate: assessing potential impacts of sea-level rise on the coast of South-East England (2018).
84. *Marc Seliger:* Thirsty landscapes - Investigating growing irrigation water consumption and potential conservation measures within Utah's largest master-planned community: Daybreak (2018).
85. *Luka Jovičić:* Spatial Data Harmonisation in Regional Context in Accordance with INSPIRE Implementing Rules (2018).
86. *Christina Kourdounouli:* Analysis of Urban Ecosystem Condition Indicators for the Large Urban Zones and City Cores in EU (2018).
87. *Jeremy Azzopardi:* Effect of distance measures and feature representations on distance-based accessibility measures (2018).
88. *Patrick Kabatha:* An open source web GIS tool for analysis and visualization of elephant GPS telemetry data, alongside environmental and anthropogenic variables (2018).
89. *Richard Alphonce Giliba:* Effects of Climate Change on Potential Geographical Distribution of Prunus africana (African cherry) in the Eastern Arc Mountain Forests of Tanzania (2018).
90. *Eiður Kristinn Eiðsson:* Transformation and linking of authoritative multi-scale geodata for the Semantic Web: A case study of Swedish national building data sets (2018).
91. *Niamh Harty:* HOP!: a PGIS and citizen science approach to monitoring the condition of upland paths (2018).
92. *José Estuardo Jara Alvear:* Solar photovoltaic potential to complement hydropower in Ecuador: A GIS-based framework of analysis (2018).
93. *Brendan O'Neill:* Multicriteria Site Suitability for Algal Biofuel Production Facilities (2018).
94. *Roman Spataru:* Spatial-temporal GIS analysis in public health – a case study of polio disease (2018).
95. *Alicja Miodońska:* Assessing evolution of ice caps in Suðurland, Iceland, in years 1986 - 2014, using multispectral satellite imagery (2019).
96. *Dennis Lindell Schettini:* A Spatial Analysis of Homicide Crime's Distribution and Association with Deprivation in Stockholm Between 2010-2017 (2019).
97. *Damiano Vesentini:* The Po Delta Biosphere Reserve: Management challenges and priorities deriving from anthropogenic pressure and sea level rise (2019).

98. *Emilie Arnesten:* Impacts of future sea level rise and high water on roads, railways and environmental objects: a GIS analysis of the potential effects of increasing sea levels and highest projected high water in Scania, Sweden (2019).
99. *Syed Muhammad Amir Raza:* Comparison of geospatial support in RDF stores: Evaluation for ICOS Carbon Portal metadata (2019).
100. *Hemin Tofiq:* Investigating the accuracy of Digital Elevation Models from UAV images in areas with low contrast: A sandy beach as a case study (2019).
101. *Evangelos Vafeiadis:* Exploring the distribution of accessibility by public transport using spatial analysis. A case study for retail concentrations and public hospitals in Athens (2019).
102. *Milan Sekulic:* Multi-Criteria GIS modelling for optimal alignment of roadway by-passes in the Tlokweng Planning Area, Botswana (2019).
103. *Ingrid Piirisaar:* A multi-criteria GIS analysis for siting of utility-scale photovoltaic solar plants in county Kilkenny, Ireland (2019).
104. *Nigel Fox:* Plant phenology and climate change: possible effect on the onset of various wild plant species' first flowering day in the UK (2019).
105. *Gunnar Hesch:* Linking conflict events and cropland development in Afghanistan, 2001 to 2011, using MODIS land cover data and Uppsala Conflict Data Programme (2019).
106. *Elijah Njoku:* Analysis of spatial-temporal pattern of Land Surface Temperature (LST) due to NDVI and elevation in Ilorin, Nigeria (2019).
107. *Katalin Bunyevácz:* Development of a GIS methodology to evaluate informal urban green areas for inclusion in a community governance program (2019).
108. *Paul dos Santos:* Automating synthetic trip data generation for an agent-based simulation of urban mobility (2019).
109. *Robert O' Dwyer:* Land cover changes in Southern Sweden from the mid-Holocene to present day: Insights for ecosystem service assessments (2019).
110. *Daniel Klingmyr:* Global scale patterns and trends in tropospheric NO2 concentrations (2019).
111. *Marwa Farouk Elkabbany:* Sea Level Rise Vulnerability Assessment for Abu Dhabi, United Arab Emirates (2019).
112. *Jip Jan van Zoonen:* Aspects of Error Quantification and Evaluation in Digital Elevation Models for Glacier Surfaces (2020).
113. *Georgios Efthymiou:* The use of bicycles in a mid-sized city – benefits and obstacles identified using a questionnaire and GIS (2020).
114. *Haruna Olayiwola Jimoh:* Assessment of Urban Sprawl in MOWE/IBAFO Axis of Ogun State using GIS Capabilities (2020).
115. *Nikolaos Barmpas Zachariadis:* Development of an iOS, Augmented Reality for disaster management (2020).
116. *Ida Storm:* ICOS Atmospheric Stations: Spatial Characterization of CO2 Footprint Areas and Evaluating the Uncertainties of Modelled CO2 Concentrations (2020).
117. *Alon Zuta:* Evaluation of water stress mapping methods in vineyards using airborne thermal imaging (2020).
118. *Marcus Eriksson:* Evaluating structural landscape development in the municipality Upplands-Bro, using landscape metrics indices (2020).

119. *Ane Rahbek Vierø:* Connectivity for Cyclists? A Network Analysis of Copenhagen’s Bike Lanes (2020).